AFRL-RY-WP-TR-2020-0013

CLEARSCOPE: FULL STACK PROVENANCE GRAPH GENERATION FOR TRANSPARENT COMPUTING ON MOBILE DEVICES

Michaell Gordon, Jordan Eikenberry, Anthony Eden, Jeffrey Perkins, Malavika Samak, Henny Sipma, and Martin Rinard

Massachusetts Institute of Technology

JULY 2020
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with The Under Secretary of Defense memorandum dated 24 May 2010 and AFRL/DSO policy clarification email dated 13 January 2020. This report is available to the general public, including foreign nationals.

Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2020-0013 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

*//Signature//

CHARLES P. SATTERTHWAITE
PM, Resilient & Agile Avionics Branch
Spectrum Warfare Division

//Signature//

BENJAMIN J. BRUCKMAN, Maj, USAF
Chief, Resilient & Agile Avionics Branch
Spectrum Warfare Division

//Signature//

JOHN F. CARR, Chief
Spectrum Warfare Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
July 2020	Final	30 June 2015 – 30 November 2019

4. TITLE AND SUBTITLE
CLEARSCOPE: FULL STACK PROVENANCE GRAPH GENERATION FOR TRANSPARENT COMPUTING ON MOBILE DEVICES

5a. CONTRACT NUMBER
FA8650-15-C-7564

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)
Michaell Gordon, Jordan Eikenberry, Anthony Eden, Jeffrey Perkins, Malavika Samak, Henny Sipma, and Martin Rinard

5d. PROJECT NUMBER
1000

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
Y1B5

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH
45433-7320
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency
DARPA/I2O
675 North Randolph Street
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYZC

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2020-0013

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with The Under Secretary of Defense memorandum dated 24 May 2010 and AFRL/DSO policy clarification email dated 13 January 2020. This material is based on research sponsored by Air Force Research laboratory (AFRL) and the Defense Advanced Research Agency (DARPA) under agreement number FA8650-15-C-7564. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation herein. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies of endorsements, either expressed or implied, of AFRL and DARPA or the U.S. Government. Report contains color.

14. ABSTRACT
The ClearScope project associates a provenance history graph for each value of Android application via a custom build of the Android operating system. Provenance provides a history of the sensitive sources and sinks that influenced a value, including the temporal order of the operations, and details of the operations (e.g., file names, IP addresses, data values, the calling program and user, etc.). This information can be employed to improve the accuracy and efficiency of malware and APT detection, forensics, and policy enforcement. The ClearScope project combines multiple instrumentation systems to provide unprecedented coverage for an Android system at low overhead. Performance experiments with the Caffeine Mark benchmarks demonstrate 14% overhead. Additionally, we demonstrate only a 1% overhead for Firefox browser benchmarks. For the TC engagements, we captured all in-bounds malicious actions performed by TA4 (the red team). For TC, we are the only system to track and report fine-grained and value-precise data-provenance. We have robust ClearScope builds for Android 5, 6, 7, and 8 for multiple devices. We also published our work in major conferences and technical reports.

15. SUBJECT TERMS
provenance, dynamic instrumentation, security, information leaks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Charles Satterthwaite
Unclassified	Unclassified	Unclassified	SAR	178	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Table of Contents

Section **Page**

List of Figures

List of Tables

1.0 SUMMARY

Detailed information about the paths that data take through a system is invaluable for understanding sources and behaviors of complex exfiltration malware. We present a new system, ClearScope, associates a provenance history graph for each value of program. Provenance provides a history of the sensitive sources and sinks that influenced a value, including the temporal order of the operations, and details of the operations (e.g., file names, IP addresses, data values, the calling program and user, etc.).

The ClearScope project included three main instrumentation and monitoring mechanisms:

1. Off-line static instrumentation of all Java code of the Android Open Source Platform (AOSP). This static instrumentation operates on the compiled DEX code for ALL of the Java code of a new Android system. It also includes aggressive and sophisticated static optimizations to reduce overhead of provenance tracking.
2. Dynamic monitoring of native code executing outside the context of the Android Java runtime. This monitoring is achieved via modifications to the kernel to provide callbacks into reporting and tracking code at system calls of interest. The monitoring is extremely low-overhead, and conservatively maintains provenance histories for values.
3. On-line static instrumentation of downloaded 3rd party applications. A stripped down version of the DEX static instrumentation is included on a ClearScope device, and modifies all downloaded applications at install time.

These three mechanisms combine to give us unprecedented coverage of the execution of apps and processes on an Android device. We have also protected the tracking and reporting with novel kernel protections.

The ClearScope system design enables this unprecedented level of provenance tracking detail by 1) structuring the provenance representation as references, via *provenance tags*, to *provenance events* that record the movement of data between system components and into or out of the device and 2) adopting a split design in which provenance events are streamed to a remote server for storage, with only the minimal information required to generate the tagged stream of events retained on the device. ClearScope also includes compiler optimizations that enable efficient provenance tracking within applications by eliminating unnecessary provenance tracking computations and adopting and efficient aggregate provenance representation for arrays when all array elements have the same provenance.

Experience using ClearScope to analyze the notorious Adups FOTA malware highlights the significant benefits that this level of comprehensive detail can bring. Performance experiments with the Caffeine Mark benchmarks show that the overall ClearScope provenance tracking overhead on this benchmark suite is 14%. Additionally, we demonstrate only a 1% overhead for Firefox browser benchmarks.

For Transparent Computing (TC)'s engagements, we captured all in-bounds malicious actions performed by TA4 (the red team). Across TC, we are the only system to track and report fine-grained and value-precise data-provenance. We have robust ClearScope builds for Android 5, 6, 7, and 8 for multiple devices. We also published our work in major conferences and technical reports [1, 2, 3, 4, 5, 6].

2.0 INTRODUCTION

Understanding the flow of information through a device can be critical for finding and understanding information and privacy leaks. A standard approach is to instrument the software running on the device to tag data with information about its source [7, 8, 9]. The information can then be propagated through the device and read at specified points to enforce privacy policies.

For the Transparent Computing (TC) program, we developed a novel and robust system, ClearScope, for precise and comprehensive provenance tracking of information that flows through Android devices. In contrast to previous systems, ClearScope tracks the complete path that data takes through the device, from its initial entry into the device through to its exit point, including applications, files, binders, and pipes that the data traverses along this path, as well as tracking flows in both the Dalvik Android Runtime (ART) and code executing outside of the runtime (native code). ClearScope can also track up to 2^{32} combinations of information sources and intermediate information traversal points. Previous systems, in contrast, can track only a small fixed number of information sources (typically between 1 to 32 sources). And the information that ClearScope delivers has unprecedented precision, including the time of data traversal events, the precise location in the application where data traversal events take place, and the initial source or sources of relevant data at the level of individual bytes.

The ClearScope project included three main instrumentation and monitoring mechanisms:

1. Off-line static instrumentation of all Java code of the Android Open Source Platform (AOSP). This static instrumentation operates on the compiled DEX code for ALL of the Java code of a new Android system. It also includes aggressive and sophisticated static optimizations to reduce overhead of provenance tracking (see Section 3.1.3).
2. Dynamic monitoring of native code executing outside the context of the Android Java runtime. This monitoring is achieved via modifications to the kernel to provide callbacks into reporting and tracking code at system calls of interest. The monitoring is extremely low-overhead, and conservatively maintains provenance histories for values (see Section 3.1.17).
3. On-line static instrumentation of downloaded 3rd party applications. A stripped down version of the DEX static instrumentation is included on a ClearScope device, and modifies all downloaded applications at install time (see Section 3.1.4).

These three mechanisms combine to give us unprecedented coverage of the execution of apps and processes on an Android device. We have also protected the tracking and reporting with novel kernel protections (see Section 3.1.17.5).

ClearScope includes several implementation techniques that enable this level of information to be productively collected from a running Android device. First, its system architecture includes a remote server that maintains the majority of the detailed information (Section 3.1.18). This system design effectively partitions the maintained provenance information between the device and the server, maintaining the majority of the detailed information on the server and only the minimal amount of information required for efficient operation on the device. With this design, the device streams collected provenance information to the server as it executes. The device itself maintains only the tables that it needs to generate the stream of provenance events. The server retains the full

provenance tracking information, including all information required to create a provenance web that captures the movement of data through the device.

ClearScope also includes several program optimizations. These include optimizations that maintain a single provenance tag for an array of values if all values in the array have the same provenance (without these optimizations the device does not even boot) and optimizations that remove provenance propagation calculations for values that do not escape the application. Together, these optimizations can reduce the provenance tracking overhead from a factor of two or more to 14% (as measured in the standard Caffeine Mark benchmark set).

2.1 Complementary Provenance Tracking Mechanisms

For ClearScope, we developed complementary provenance tracking systems motivated by the common execution pattern of Android applications. Android apps are typically implemented mostly in Java (or Katlin) with included native libraries (written in C/C++). Execution of the Java component of apps happens in the context of the ART, while binary libraries execute natively (outside the context of ART). Most of the dynamic execution of a (non-game) application is in the ART, and binary libraries are called like stateless accelerators for performance sensitive operations. Our complete system tracks provenance through ART execution with complete precision with deep application modification.

For the native execution context, our experiments showed the accelerator model computation does not require such deep and invasive application modification and monitoring. Our native tracking component thus keeps a single tag of provenance for each thread executing outside the context of ART. This extremely low-overhead mechanism is suited towards capturing flows in the native code where syscalls are rare, and most tainted data comes from (and is sinked in) the ART context.

When ART code calls native code (in a binary library) there is well-defined interface that is termed the Java Native Interface (JNI). We have modified this interface to switch between the tracing mechanisms and pass provenance information between the ART context and the native content.

The single provenance tag of the thread (which we term the “provenance union”, see Section 3.1.17.2) is reset when the thread switches from the ART context to the native context via the call. Furthermore, the tags on the arguments of the JNI call that initiated the context switch are joined and the provenance union is set to this join. In the native context, when provenance is injected into the thread via the execution of a syscall source, the new provenance is joined with the provenance union. When the native context is finished execution, on switching back to the ART context, the provenance union tag is joined with any return values or memory returned to the ART via the JNI call.

This combination of mechanisms enabled ClearScope to maintain acceptable overhead, and unprecedented accuracy, since to our knowledge it is the only non-emulation system to track provenance through both ART and non-ART contexts.

2.2 Evaluation and Results

During the life of the project, we have the following results and evaluations:

1. For each of the engagements, our system captured all in-bounds[1] malice performed by TA4 (the red team). For the engagements, our system executed in a robust manner, e.g., for the last engagement we supported 3 devices over 2 weeks.
2. We have used our implemented ClearScope system to analyze the notorious Adups FOTA malware [10] shipped with over 700 million Android devices. This malware implements a persistent, hidden information exfiltration algorithm that exfiltrates SMS messages, histories, call logs, and contacts to an external Chinese web site, with both 24 and 72 hour exfiltration cycles. Understanding this malware took Kryptowire months of analysis effort [11]. With ClearScope, we were able to analyze the exact flow, pinpoint the source of the information leak, and characterize the behavior of the malware with several hours examining the provenance logs.
3. Recently, we employed ClearScope on Android 8.1 to evaluate 50 applications from the Google Play top 100 apps list (as of August 2019). We collected over 1TB of data over 3 weeks of execution of the applications over 8 devices. Currently, we are analyzing this data to find interesting security and privacy violations.

2.3 Contributions

ClearScope makes the following contributions:

- The ClearScope project maintained industry-strict coding practices to develop multiple robust versions of our custom builds of the AOSP. We have robust builds for Android versions 5, 6, 7, and 8. Each of the versions pass all expected tests in the Android Compatibility Test Suite (CTS) which includes hundreds of thousands of tests (see Section 3.1.16). Our system are ready for many transition opportunities including application vetting, honey pots, malware reverse engineering, and end-user deployment.
- The system was developed over 4 years and includes over 7.34M LOC of the Android Open Source Project added or modified. Four full-time developers worked on the project, and the project graduated three Masters students and two PhD students.
- Unlike previous systems, ClearScope records the complete path that data takes as it traverses the system, including data entry and exit points, and application, file, pipe, binder, and socket traversals. The recorded provenance information includes detail such as times when provenance actions occur and provenance information that the level of individual bytes of data.
- Unlike previous systems, ClearScope tracks and reports flows through both the ART and through code executing outside the context of the ART (native execution). Each of the two

[1] We considered malice “in-bounds” if our system was scheduled, based on our statement of work at the time of the engagement, to capture and report the underlying mechanism used for the malice.

execution contexts includes a separate mechanism for tracking. ART tracking is accomplished via DEX instruction-level instrumentation (see Section 3.1.3, while native tracking is accomplished via syscall monitoring via a novel kernel modifications (see Section 3.1.17).

- In our system, provenance reporting is extremely low overhead and naturally multi-threaded. Constructing a provenance message and sending the message to the on-device provenance service is performed in the execution context of the user thread that caused the message to be constructed. We call this mechanism “Same-Thread Tracing” (see Section 3.1.17.1).
- ClearScope is extremely high-precision, with no false positives for tracking in the ART. For ART, our system maintains a provenance tag per array element. For both systems, a tag per byte is associated with each byte transferred over IPC/RPC, and for each byte of the filesystem written by tracked processes.
- ClearScope collects an unprecedented amount of information about the flow of data through the device. It is infeasible to maintain this information only the device itself — the amount of information would exceed the storage capacity of typically Android devices. ClearScope therefore adopts a new design that streams information off to a remote server, maintaining only the information required to efficiently generate the stream locally. This novel split design is one of the key prerequisites to the effective collection and maintenance of this level of provenance information.
- ClearScope implements several optimizations that enable it to operate with acceptably low overhead (14% on Caffeine Mark benchmarks). These optimizations include using a single provenance tag to represent the provenance for all array elements when the array elements all have the same provenance and eliminating provenance calculations for provenance that does not escape the application.
- The native tracking component of ClearScope is extremely low-overhead, with only 1% overhead added for popular browser benchmarks executing in Firefox on the device.
- We have used our implemented ClearScope system to analyze the Adups FOTA malware as well as 35 top Android applications from the Google Play Store. These results highlight the effectiveness of ClearScope in collecting detailed and comprehensive provenance information for these applications.
- Our Android 8 system includes protections against insider attacks on our tracking and reporting. The provenance messages inbound to the reporting service are authenticated such that the service can determine if the app has the ability to tamper with messages, e.g., its privileges have been maliciously escalated. The kernel instructions added for enabling and disabling tracing are guarded to make sure they are called from trusted code. Furthermore, our off-device provenance translator can detect tampering of the provenance shadow filesystem.

2.4 Conclusion and Next Steps

Accurate provenance information is critical for understanding device behavior and how information flows through the device. This information can be particularly critical for understanding persistent and stealthy information exfiltration malware. In comparison with

previous taint tracking systems, ClearScope provides comprehensive provenance tracking that it unprecedented in the quality and detail of the information that it can provide.

ClearScope is robust and ready for transition opportunities. Some of the appropriate missions include:

- Malware and Advanced Persistent Threat (APT) detection,
- Malware and APT forensics,
- Real-time, fine-grained, and advanced policy enforcement,
- Pre-install application vetting for malware, and
- App understanding for bug finding and policy adherence.

Furthermore, the techniques that we have developed in the context of Android will easily transfer to other systems including Linux desktop / server, IoT and routers.

We sincerely thank DARPA for giving us the opportunity to develop ClearScope and hope that it finds use in the DoD.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

For all the research we performed in this program, we adopted an experimental approach driven by the evaluation scenarios.

3.1 ClearScope

We next present the ClearScope implementation, including the representation of provenance throughout the system, when provenance events are generated, the overall system design, and the different ClearScope optimizations.

3.1.1 System Design

This section provides a formal presentation of our novel notion of provenance; how provenance is tracked and reported.

3.1.1.1 Provenance Events

ClearScope instruments the Android system and the DEX executables to emit provenance events at program points where data enters or exits the device, is stored or retrieved from files on the device, or enters or exits Android applications. These provenance events are then streamed off to a remote server, which maintains the streamed provenance information. Each event has the following fields:

- **Flow:** Tells whether the event is a source event (when data enters an application), a sink event (when information leaves an application), or other event. Examples of other events include file events (such as file creation, deletion, or open), binder events (such as open or close), and pipe events (such as open or close).
- **Event Type:** Information about the type of the event. Many events are triggered by system calls; this field typically records the system call that triggered the event.
- **Application Information:** The application id, thread id, and program point (summarized as debugging information that identifies the specific point in the program where the event occurred) for the event.
- **Time:** The time when the event occurred.
- **Event Data:** Provenance data for the event. This data typically includes the provenance tags for each byte of transferred data (run-length encoded for events that transfer multiple bytes). It can also optionally include all of the transferred data.

3.1.1.2 Provenance Tags

ClearScope maintains a 32 bit provenance tag for every byte of primitive data (characters, integers, floating point numbers, booleans, etc.) accessed by Android applications, stored in the local file system, or transferred between Android applications. For data in Android applications, these tags are stored in shadow fields that ClearScope adds to the Java data structures for this purpose. For data stored in file systems, each file has a shadow file that stores this provenance information. For data between Android applications, we have modified the communication mechanism (such as Binder) to include additional metadata that carries these provenance tags.

Each 32 bit tag indexes data structures that maintain detailed provenance information about the tagged data. Conceptually, these data structures maintain information about the last provenance event for the data, with the data structures linked together to enable the reconstruction of the complete provenance web of events for each byte of primitive data in the system. This provenance web captures the complete path through the device for that byte. We next detail the information that the provenance tags index.

Provenance Sets: Some values are derived from multiple pieces of data. For example, a value may be computed by adding values read from a file to values read from the GPS on the device. Provenance sets record the sets of provenance tags that capture this value derivation information.

Previous Provenance Tag: This tag links provenance data structures together to enable the server to reconstruct the complete provenance web for each byte of information. The nodes in this web are the provenance events that record the movement and computation of data through the system. The edges record relationships between these events. For example, if an application reads data from a file, the provenance tags for the data inside the application will index a data structure that stores information about the corresponding file read event. The previous provenance tag in this data structure will index a data structure that stores information about the file write event that wrote the data into the file. The previous provenance tag for this file write data structure will, in turn, index a data structure that stores the provenance information for the provenance event that injected the data into the application that wrote the file. In this way the previous provenance tags enable the reconstruction of the complete provenance provenance web that captures the detailed flow of information through the device.

File Provenance: This data structure maintains information about provenance events on files. There are several cases:

- **File Write:** Each file has a shadow file that stores the provenance information for the data in that file. Each of the tags in this shadow file references provenance information that summarize the file write events that stored the data in that file. The recorded information includes the application that wrote the data and the statement in the application that wrote the data. The previous tag enables ClearScope to trace the provenance of the data back through the application that write the data.
- **File Read:** Data that was obtained by reading a file has a provenance tag that indexes a data structure that records information about the file read events that generated the data. The recorded information includes the file, the offset within the file for the data, and the time of the read. The previous tag indexes the corresponding file write data structures that summarize the events that wrote the data into the file.
- **File Open, Close, Delete:** The indexed data structure records information about the file open, close, or delete operation. This information includes the application and statement within the application that performed the operation.

Binder Provenance: Android uses the Binder mechanism [12] to communicate information between Android software components. ClearScope maintains detailed provenance information for information communicated via the Binder, including byte-level provenance for all communicated data. Supporting this detailed provenance information required extensive changes to the Binder implementation to support passing additional provenance information through the Binder interface.

Android applications also access Android services via the Binder. To support these services, we developed 81 provenance types to identify the specific service that generated each byte of data. Examples include the camera, the microphone, the GPS, and a wide variety of sensors. So, for example, if an application reads data from the camera, the provenance tags for the camera data inside the application will index data structures that identify the data as coming from the camera along with metadata such as the time when the data was read from the camera.

Binder performs file descriptor translation across binder calls — a file descriptor in one application can be transferred via the Binder to another application, which can then use the translated file descriptor to read the referenced file. ClearScope augments the Binder implementation to appropriately configure the file descriptors for the shadow file in the application receiving the information from the Binder.

Network Provenance: Network provenance data structures record information about provenance events for the network. The recorded information includes the IP address and port and the time of the network read or write. It is also optionally possible to record the transmitted or received information. ClearScope also records provenance events that open or close network connections.

Pipe Provenance: Pipe provenance data structures record information about provenance events involving pipes. The recorded information includes the two communicating applications and the time of the communication. ClearScope also records provenance events that open or close pipes.

3.1.1.3 System Design

By streaming much of the provenance information off the device to a remote server, ClearScope avoids the need to accumulate all of the provenance information on the device. This design decision is critical to enabling ClearScope to function on Android devices such as smartphones, which typically have limited storage capacity in comparison with a remote server.

The decision to structure the provenance system as events referenced by provenance tags enables this productive division of responsibility between the device and the server. With this design, the events, which contain the vast majority of the information, are stored on the server and available for analysis. The device stores the shadow files for the file system on the device and per-application provenance tag mappings that store just enough data to enable the device to memoize provenance lookups and generate the stream of provenance events. This approach enables ClearScope to deliver unprecedented levels of provenance detail, including the construction of a complete byte-level provenance web, while still operating on devices with limited resources.

With this system design, each application has its own provenance table, stored in application memory in the (modified) Dalvik runtime. This table enables the application to perform the required quick memoized provenance tag lookups. Provenance tags are unique across applications and allocated to applications in blocks by a tag system service built for this purpose.

3.1.1.4 Provenance Propagation

We next present an overview of the provenance propagation algorithm in ClearScope. We start with the basic algorithm, then discuss several optimizations: aggregate array provenance, loop specialization, method specialization, and dead provenance elimination.

Basic Provenance Propagation: The basic provenance propagation algorithm instruments the DEX code to appropriately propagate the provenance across individual computing instructions within the Android application. The instrumentation augments each primitive Java value with a

shadow provenance tag. Provenance information for composite values such as Java objects are comprised of the union of the provenance information for the primitive values contained in the object.

The ClearScope compiler instruments the Android DEX code to include additional instructions that propagate the provenance tags. For each load or store instruction, the compiler adds a corresponding load or store that propagates the provenance tags to the corresponding shadow fields. For compute instructions (such as instructions that add two values), the ClearScope compiler inserts a provenance join operation. This operation takes the provenance tags for the operands of the compute instruction and returns a new provenance tag for the join of the two operand provenance tags. This returned join value typically indexes a provenance set containing a list of the two operand provenance tags.

The ClearScope instrumentation memoizes calls to the provenance join operation. If the two operand provenance tags have been previously joined, the instrumentation simply returns the provenance tag from the previous join operation. This memoization improves performance and eliminates the excessive creation of new operand tags that would otherwise take place.

The instrumentation also augments procedure calls with shadow parameters to hold the provenance information for any primitive parameters. There is a single global object that holds the provenance information for the return value. Binder and pipe calls are also augmented to pass provenance information in addition to the values. This provenance information is maintained at the level of the individual bytes of transferred data.

The DEX instrumentation can be added either offline or on the device. ClearScope implements a mechanism that intercepts the call to the DEX compiler, adds the instrumentation, then proceeds on to invoke the DEX compiler on the instrumented DEX code.

Array Aggregation Optimization: Many arrays store data with homogeneous provenance information, i.e., all array elements have the same provenance tags. ClearScope optimizes for this common case by storing a single provenance tag for all array elements when these elements have the same tag. This optimization substantially reduces the ClearScope memory footprint and makes the difference between a feasible and infeasible system — without this optimization the device will not boot.

Because of this optimization, the ClearScope DEX instrumentation has to check several cases on each array access (in the absence of the loop specialization optimization described below). Each array can be in one of two states: *aggregated* (in which there is a single provenance tag for all array elements) or *expanded* (in which there is a shadow array that holds the provenance information, with each element of the shadow array holding the provenance for the corresponding array element). Array reads check array the state to determine whether to fetch the provenance tag from the aggregate tag or the shadow array. Array writes check the array state along with the provenance tag for the written value to determine if the instrumentation should 1) leave the aggregate provenance tag intact (if the array is in the aggregate state and the written array element has the same state as the array's aggregate state), 2) write a shadow array element (if the array is in expanded state), or 3) expand the array and write a shadow array element (if the array is in aggregate state and the provenance tag for the written element does not match the aggregate tag).

Loop Specialization: The loop specialization optimization is designed to work with the array aggregation optimization. This optimization adds a loop header to loops that access arrays. The loop header checks for common optimizable cases, then jumps to specialized code generated for

each such case. The most common optimizable case occurs when the provenance can be precomputed in the loop header for all accesses in the loop. To apply this optimization, the ClearScope compiler:

- **Array Extraction:** The ClearScope compiler analyzes the loop body to find all arrays accessed in the body.
- **Aggregate Checks:** For each extracted array, the ClearScope compiler checks to see if the array is in aggregate state. If so, it retrieves the provenance tags for each array.
- **Write Checks:** For each array written in the loop, the ClearScope compiler checks that all of the writes will write values into the array whose provenance information matches the aggregate provenance tag.

If the aggregate and write checks succeed, the loop will not change the provenance information and the ClearScope compiler generates specialized loop code that completely omits the provenance tracking code.

The ClearScope compiler also implements more sophisticated checks that, for example, check that the loop writes every element of an array and that all written elements have the same provenance. In this case the generated code inserts a single provenance assignment operation into the header that sets the provenance tag to the new value and again generates specialized code that completely omits the provenance tracking code.

Method Specialization: In some cases, depending on the calling context, ClearScope can detect that method calls will leave the provenance information unchanged. In such cases ClearScope generates and invokes a specialized version of the method that omits provenance tracking instrumentation.

Dead Provenance Information Elimination: ClearScope does not need to maintain provenance information for computed values that do not escape the application in which they are located. Such values often occur, for example, in conditionals or loop bounds. ClearScope implements a program analysis that detects such values and eliminates all provenance instrumentation for these values.

3.1.2 Provenance Formalization

In this section, we formally define provenance tracking for a sequentially consistent multi-processing system. Section 3.1.2.1 formalizes the system state. Section 3.1.2.2 extends the system state definition and defines the provenance tracking.

3.1.2.1 System definition

Val represents the set of values that can be defined in the system. It consists of all memory addresses and integer values. *Var* represents all plausible variable names that may be defined by a process in the system. *Process* represents the set of all processes in the system.

$$\begin{aligned} c, a \in \text{Val} &: \text{Int} \cup \text{Addr} \\ v_1, v_2, v_3 \in \text{Var} &\subset \text{String} \\ p_1, p_2 \in \text{Process} &\subset \text{Int} \end{aligned}$$

The system state is defined as a function, that maps every process in the system to a tuple (ϱ,μ). Where, ϱ represents the process environment and μ represents the process memory. ϱ maps every process variable to a value and μ maps every memory address to a value.

$$\begin{aligned} \text{State} &: \text{Process} \mapsto \text{Mem} \times \text{Env} \\ \mu \in \text{Mem} &: \text{Addr} \mapsto \text{Val} \\ \rho \in \text{Env} &: \text{Var} \mapsto \text{Val} \end{aligned}$$

The definition assumes the operating system as yet another process with special privileges. It is represented as process po, that has its own process environment ϱo and memory μo. The memory μo represents all the system resources such as file system, network ports etc., managed by the operating system.

$$\begin{aligned} \texttt{stmt} \mapsto\ & \texttt{start p} \mid \texttt{end p} \mid \texttt{switch } \texttt{p}_1\ \texttt{p}_2 \mid \texttt{read } \texttt{v}_\texttt{d}\ \texttt{v}_\texttt{s} \mid \\ & \texttt{write } \texttt{v}_\texttt{s}\ \texttt{v}_\texttt{d} \mid \texttt{sys-read } \texttt{v}_\texttt{d}\ \texttt{v}_\texttt{s} \mid \texttt{sys-write } \texttt{v}_\texttt{s}\ \texttt{v}_\texttt{d} \mid \\ & \texttt{v}_1 = \texttt{c} \mid \texttt{v}_1 = \texttt{v}_2 \mid \texttt{v}_1 = \texttt{v}_2 \oplus \texttt{v}_2 \\ \oplus \mapsto\ & + \mid - \mid \div \mid \times \mid \neq \mid = \mid < \mid > \mid \text{mod} \end{aligned}$$

Figure 3.1: Execution language of the system

Figure 3.1 presents a simple language that presents the set of instructions that may be executed by the processes in the system. The first three statements represent the starting a new process p, terminating a process p and context switching from process p_1 to p_2 respectively. These statements can only be executed by the operating system, represented as process po. The statement ($read\ v_d\ v_s$) retrieves a value stored by the process memory location specified by variable v_s and assigns it to variable v_d. Here, the variable v_s can only contain a memory location owned by the current process. Similarly, the statement ($write\ v_s\ v_d$) writes the value in variable v_s to a memory location specified by variable v_d. Here too, the target memory location must be owned by the current process.

The statements ($sys\text{-}read\ v_d\ v_s$) and ($sys\text{-}write\ v_s\ v_d$) are system calls, that can read/write the system resources maintained by the operating system. Statement ($sys\text{-}read\ v_d\ v_s$) reads a value from a memory location maintained by the operating system (location specified by variable v_s) and assigns it to variable v_d. Similarly, the statement ($sys\text{-}write\ v_s\ v_d$) writes a value contained in variable v_s into a memory location specified by variable v_d. Here too, the target address specified by variable v_d is managed by the operating system. Statement (v_1=v_2) assigns value stored in variable v_2 to variable v_1. (v_1=$v_2 \oplus v_3$) performs the binary operation specified by the operator $\oplus$ on the contents of variables v_2, v_3, and stores the result in variable v_1. (v_1=c) assigns a constant value c to variable v_1.

$$\texttt{sys-boot} : (\texttt{s},\texttt{d}) \leftarrow \texttt{ReadDB}() \qquad \texttt{sys-shutdown} : \texttt{WriteDB}(\texttt{s},\texttt{d})$$

$$\frac{\texttt{s}' \leftarrow \texttt{s}[\texttt{p}_\texttt{i} \leftarrow \texttt{init}]}{(\texttt{s},\texttt{d},\texttt{p}) \xrightarrow{\texttt{l:(start } \texttt{p}_\texttt{i}\texttt{)}} (\texttt{s}',\texttt{d},\texttt{p})}[\texttt{strt}] \qquad \frac{\texttt{s}' \leftarrow \texttt{s}[\texttt{p}_\texttt{i} \leftarrow \bot]}{(\texttt{s},\texttt{d},\texttt{p}) \xrightarrow{\texttt{l:(end } \texttt{p}_\texttt{i}\texttt{)}} (\texttt{s}',\texttt{d},\texttt{p})}[\texttt{end}] \qquad \frac{}{(\texttt{s},\texttt{d},\texttt{p}_\texttt{i}) \xrightarrow{\texttt{l:(switch } \texttt{p}_\texttt{i}\ \texttt{p}_\texttt{j}\texttt{)}} (\texttt{s},\texttt{d},\texttt{p}_\texttt{j})}$$

$$\frac{(\mu_p,\rho_p) \leftarrow s[p]\ \ (a,i) \leftarrow \rho_p[v_s]\ \ \rho'_p \leftarrow \rho_p[v_d \leftarrow \mu_p[a]]}{(s,d,p) \xrightarrow{\text{l: read } v_d\ v_s} (s[p \leftarrow (\mu_p,\rho'_p)],d,p)}[\text{rd}] \qquad \frac{(\mu_p,\rho_p) \leftarrow s[p]\ \ (a,i) \leftarrow \rho_p[v_d]\ \ \mu'_p \leftarrow \mu_p[a \leftarrow}{(s,d,p) \xrightarrow{\text{l:write } v_s\ v_d} (s[p \leftarrow (\mu'_p,\rho_p)],d,}$$

$$\frac{\begin{array}{c}(\mu_o,\rho_o) \leftarrow s[p_o]\ \ (\mu_p,\rho_p) \leftarrow s[p]\ \ (a,i) \leftarrow \rho_p[v_s]\ \ (c,j) \leftarrow \mu_o[a]\ \ k \leftarrow \text{newId}() \\ \rho'_p \leftarrow \rho_p[v_d \leftarrow (c,k)]\ \ h' \leftarrow h[k \leftarrow (p,l,c,\{j\},i)]\end{array}}{(s,d,p) \xrightarrow{\text{l: sys-read } v_d\ v_s} (s[p \leftarrow (\mu_p,\rho'_p)],h',p)}[\text{sys-rd}]$$

$$\frac{\begin{array}{c}(\mu_o,\rho_o) \leftarrow s[p_o]\ \ (\mu_p,\rho_p) \leftarrow s[p]\ \ (a,i) \leftarrow \rho_p[v_d]\ \ (c,j) \leftarrow \rho_p[v_s]\ \ k \leftarrow \text{newId}() \\ \mu'_o \leftarrow \mu_o[a \leftarrow (c,k)]\ \ h' \leftarrow h[k \leftarrow (p,l,c,\{j\},i)]\end{array}}{(s,d,p) \xrightarrow{\text{l: sys-write } v_s\ v_d} (s[p_o \leftarrow (\mu'_o,\rho_o)],h',p)}[\text{sys-wr}]$$

$$\frac{\begin{array}{c}(\mu_p,\rho_p) \leftarrow s[p]\ \ (c_i,i) \leftarrow \rho_p[v_2]\ \ (c_j,j) \leftarrow \rho_p[v_3]\ \ k \leftarrow \text{newId}() \\ h' \leftarrow h[k \leftarrow (p,l,c_i \oplus c_j,\{i,j\},\perp)]\ \ \rho'_p \leftarrow \rho_p[v_1 \leftarrow (c_i \oplus c_j,k)]\end{array}}{(s,d,p) \xrightarrow{\text{l: } v_1 = v_2 \oplus v_3} (s[p \leftarrow (\mu_p,\rho'_p)],h',p)}[\text{bi-op}]$$

$$\frac{(\mu_p,\rho_p) \leftarrow s[p]\ \ \rho'_p \leftarrow \rho_p[v_1 \leftarrow \rho_p[v_2]]}{(s,d,p) \xrightarrow{\text{l:}v_1=v_2} (s[p \leftarrow (\mu_p,\rho'_p)],d,p)}[\text{assign}] \qquad \frac{(\mu_p,\rho_p) \leftarrow s[p]\ \ \rho'_p \leftarrow \rho_p[v_1 \leftarrow (c,\perp)]}{(s,d,p) \xrightarrow{\text{l:}v_1 = c} (s[p \leftarrow (\mu_p,\rho'_p)],d,p)}[\text{c-assign}]$$

Figure 3.2: Inference rules

3.1.2.2 Provenance tracking

We now define the provenance tracking for the above system. Provenance tracking associates a provenance identifier to every variable and memory location owned by all processes in the system. It also maintains a database that stores important data pertaining to every provenance identifier created in the system. Using the database and the provenance identifier associated with a memory location, the precise provenance data of a value can be derived.

We extend the system state as follows to enable provenance tracking.

$$\begin{aligned}\mu \in \text{Mem} &: \text{Addr} \mapsto \text{Val} \times \text{Int} \\ \rho \in \text{Env} &: \text{Var} \mapsto \text{Val} \times \text{Int}\end{aligned}$$

Here, every process memory maps its address to a pair instead, comprising a value and a provenance identifier. Similarly, every process environment maps variables to a value-provenance identifier pair.

The provenance tracking maintains a tracking state as a tuple (s,d,p). Where, s is the current system state, d is the database instance that contains the provenance information of the system and p is the process that is currently executing in the system.

$$\text{State} \times \text{Database} \times \text{Process}$$
$$\mathtt{d} \in \text{Database} : \text{Int} \mapsto \text{Process} \times \text{Src} \times \text{Val} \times \mathscr{P}(\text{Int}) \times \text{Int}$$

The database instance d maps every provenance identifier created by the system to a tuple (p, l,c, $\{i_1 \ldots i_m\}$,i_n) , where,

- p is the process that created the provenance entry.
- l is the static source code location of the instruction executed by p, responsible for creating the provenance entry.
- c is the constant value that was read, written, or computed by the instruction.
- $i_1 \ldots i_m$ are the provenance identifiers of values, that influenced the value c.
- If the instruction at l accessed a memory address a maintained by the operating system, i_n represents the provenance identifier associated with computing the address a. Otherwise, it is set to $\perp$.

Figure 3.2 presents the rules for tracking provenance flow within a system. At system boot, the provenance tracking initializes the system state s and the database instance d appropriately from a file. The file contains the existing provenance data of the system resources from previous executions. When the system shuts down, provenance tracking updates the file with the new provenance data associated with the system resources and the updated provenance database. This ensures the provenance information is persistently maintained across boots.

Provenance tracking for creating a new process p is handled by rule [$strt$]. It initializes the process memory and environment. Every memory location and environment variable allocated to p is mapped to a special provenance id $\perp$, that indicates the absence of provenance data. Rule [end] is applied when, a process p_i terminates. The rule modifies the system state s and deallocates the resources allocated to the process. The rule [swp] is applied when the operating system context switches from process p_i to p_j. This rule updates the tracking state by updating the current process to p_j.

A process reading and writing its own memory is handled by [rd] and [wr] rules respectively. If a process executes ($read\ v_s\ v_d$) statement, the [rd] rule reads the process environment and retrieves the address a referenced by variable v_s. It then reads the process memory and maps the value-provenance id pair associated with address a to variable v_d. Similarly, executing ($write\ v_d\ v_s$) statement invokes [wr] rule, which maps address a referenced by variable v_d, to the value-provenance id pair mapped to variable v_s. These statements do not create any new provenance identifier or require updates to the database instance d.

Every system call issued by a process creates a new provenance identifier and a suitable update to the provenance database. Executing system read and write calls are handled by rules [$sys{-}rd$] and [$sys{-}wr$] respectively. Executing ($sys{-}read\ v_s\ v_d$) system call, triggers [$sys{-}rd$] rule. This rule creates a new provenance identifier k and maps it to a new entry in the database d. The rule first identifies value-id pair (a,i) mapped to variable v_s, where address a is maintained by the the operating system. Next, the rule identifies the value-id pair (c,j) mapped to address a. The new provenance entry created for the new provenance identifier k, encodes both provenance identifiers i,j. In addition to this, the entry also encodes the source location l in the program

which triggered the system call, the process id of the executing process, and value c at address a. The variable v_d is re-mapped to a new value-id pair (c,k). Similarly, executing $(sys\text{–}write\ v_d\ v_s)$ creates a new provenance identity and a corresponding entry in the database. The target memory location a written by the process, is owned by the operating system and is re-mapped to a newly created value-id pair. We do not elaborate this rule for brevity.

The tracking also creates a new provenance identifier when a binary assignment statement $(v_1=$ EQ v\s\do6(2)⊕v\s\do6(3)) is executed by the process. Rule [$bi\text{–}op$] handles this case, and it creates a new id k and adds a corresponding provenance entry to the database. The new entry encodes the provenance identifiers associated with both the operands v_2, v_3. The variable v_1 is re-mapped to a new value-id pair. Assignment statement $(v_1=v_2)$ is handled by [$assign$] rule which maps variable v_1 to the same value-id pair as v_2. Finally, executing $(v_1=c)$ maps the variable v_1 to $(c, \perp)$ pair, indicating the absence of provenance data for variable v_1.

3.1.3 Java-Based Instrumentation Implementation

We use Soot as a framework for instrumenting Dalvik bytecode for both the system and the application. It works by first translating the Dalvik instructions into Jimple IR. From the Jimple, we generate new (semantically equivalent) taint-tracked Jimple based on the following grammatical rules. In the following, ⇒ denotes instruction insertion *after* the current rule, and ⇐ denotes instruction insertion *before* the current rule.

□*stmt*□ □ □*assignStmt*□ | □*identityStmt*□ | □*gotoStmt*□ | □*ifStmt*□
| □*invokeStmt*□ | □*switchStmt*□ | □*monitorStmt*□
| □*returnStmt*□ | □*throwStmt*□ | □*breakpointStmt*□
| □*nopStmt*□

□*assignStmt*□ □ LOCAL = □*rvalue*□ ; {
if (LOCAL.getType() instanceof PrimType) ⇒LOCAL$_t$= □*rvalue*□ $_t$
} | FIELD = □*imm*□ ; {
if (FIELD.getType() instanceof PrimType) ⇒FIELD$_t$= □*imm*□ $_t$
} | LOCAL . FIELD = □*imm*□ ; {
if (FIELD.getType() instanceof PrimType) ⇒LOCAL.FIELD$_t$= □*imm*□ $_t$
} | LOCAL [□*imm*$_1$□] = □*imm*$_2$□ ;
if (LOCAL.getType() instanceof PrimType) {
⇒LOCAL$_a$ = TC.getAggregateTaint(LOCAL)
⇒if (LOCAL$_a$!= -1) goto AGGR
⇒ LOCAL$_t$ = TC.getArrayTaint(LOCAL)
⇒ goto NONAGGR
⇒AGGR:
⇒ if (LOCAL$_a$ == □*imm*$_2$□ $_t$) goto DONE

⇒ $LOCAL_t$ = TC.unaggregate(LOCAL)
⇒NONAGGR:
⇒ $LOCAL_t$ [□*imm*$_1$□] = □*imm*$_2$□ $_t$
⇒DONE:
}

□*identityStmt*□ □ LOCAL **:=** **@this** **:** TYPE **;**
| LOCAL **:=** **@parameter** $_n$ **:** TYPE **;** {
if (TYPE instanceof PrimType) {
/*
Note that *n* in the original statement will be shifted by the number of tag parameters seen (up to *n*)
*/
⇒$LOCAL_t$:= **@parameter** $_{n+1}$: int
}
if (*n* == numOfParameters) {
⇒TCRET := **@parameter** $_{n+1}$: java.lang.TCReturn
}
} | LOCAL **:=** **@exception** **;**

□*gotoStmt*□ □ **goto** LABEL **;**

□*ifStmt*□ □ **if** □*conditionExpr*□ **goto** LABEL **;**

□*invokeStmt*□ □ **invoke** □*invokeExpr*□ **;**

□*switchStmt*□ □ **lookupswitch** □*imm*□ **{**
case VALUE $_1$ **:** **goto** LABEL $_1$ **;**
⋮
case VALUE $_n$ **:** **goto** LABEL $_n$ **;**
default **:** **goto** DEFAULTLABEL **;** **}**
| **tableswitch** □*imm*□ **{**
case LOW **:** **goto** LOWLABEL **;**
⋮
case HIGH **:** **goto** HIGHLABEL **;**
default **:** **goto** DEFAULTLABEL **;** **}**

□*monitorStmt*□ □ **entermonitor** □*imm*□ **;** | **exitmonitor** □*imm*□ **;**

□*returnStmt*□ □ **return** □*imm*□ **;** {
if (returnType instanceof PrimType) ⇐ TCRET.taint = □*imm*□ $_t$
} | **return** **;**

□*throwStmt*□ □ **throw** □*imm*□ **;**

□ **breakpoint** **;**
□*breakpointStmt*
□

□*nopStmt*□ □ **nop** **;**

□*imm*□ □ LOCAL {
□*imm*□ $_t$ → $LOCAL_t$
} | CONSTANT {

if (CONSTANT instanceof PrimType) □*imm*□ $_t$ → TRUSTED

}

□ □*imm*□ $_1$ □*condop*□ □*imm*□ $_2$

□*conditionExpr*□

□*condop*□ □ > | < | = | ≠ | ≤ | ≥

□*rvalue*□ □ □*concreteRef*□ {

□*rvalue*□ $_t$ → □*concreteRef*□ $_t$;

} | □*imm*□ {

□*rvalue*□ $_t$ → □*imm*□ $_t$;

} | □*expr*□ {

□*rvalue*□ $_t$ → □*expr*□ $_t$;

}

□*concreteRef*□ □ FIELD {

if (FIELD.getType() instanceof PrimType) □*concreteRef*□ $_t$ → FIELD$_t$;

} | LOCAL **.** FIELD {

if (FIELD.getType() instanceof PrimType) □*concreteRef*□ $_t$ → LOCAL.FIELD$_t$;

} | LOCAL **[** □*imm*□ **]**

if (LOCAL.getType() instanceof PrimType) {

⇒tagTmp = TC.getAggregateTaint(LOCAL)

⇒if (tagTmp != -1) goto AGGR

⇒ tagArrayTmp = TC.getArrayTaint(LOCAL)

⇒ tagTmp = tagArrayTmp[□*imm*□]

⇒AGGR:

□*concreteRef*□ $_t$ → tagTmp

}

□*invokeExpr*□ □ **specialinvoke** LOCAL **.** M **(** □*imm*$_1$□ **,** ⋯ **,** □*imm*$_n$□ **)**

| **interfaceinvoke** LOCAL **.** M **(** □*imm*$_1$□ **,** ⋯ **,** □*imm*$_n$□ **)**

| **virtualinvoke** LOCAL **.** M **(** □*imm*$_1$□ **,** ⋯ **,** □*imm*$_n$□ **)**

| **staticinvoke** M **(** □*imm*$_1$□ **,** ⋯ **,** □*imm*$_n$□ **)**

/*

Parameters □*imm*$_1$□ , ⋯ , □*imm*$_n$□

will (potentially) become □*imm*$_1$□ , □*imm*$_1$□ $_t$, ⋯ , □*imm*$_n$□ , □*imm*$_n$□ $_t$, TCRET

*/

if (M.getReturnType() instanceof PrimType) □*invokeExpr*□ $_t$ → TCRET.taint

□*expr*□ □ □*imm*$_1$□ □*binop*□ □*imm*$_2$□ {

if (□*imm*$_1$□ .getType() instanceof PrimType && □*imm*$_2$□ .getType() instanceof PrimType) {

```
                    if ( □imm1□  instanceof Constant) {
                      if ( □imm2□  instanceof Constant) {
                        □expr□ t → TRUSTED
                      } else {
                        □expr□ t → □imm2□ t
                      }
                    } else if ( □imm2□  instanceof Constant) {
                      □expr□ t → □imm1□ t
                    } else {
                      □expr□ t → TC.join(□imm1□ t, □imm2□ t)
                    }
                  }
                } | ( TYPE ) □imm□
                | □imm□ instanceof TYPE
                | □invokeExpr□
                | new REFTYPE
                | newarray ( TYPE ) [ □imm□ ]
                | newmultiarray ( TYPE ) [ □imm□ 1 ] ⋯ [ □imm□ n ]
                | length □imm□
                | neg □imm□
□binop□      □  + | - | > | < | = | ≠ | ≤ | ≥ | * | / | << | >>
                | <<< | % | rem | &
```

Methods are copied and instrumented on the method copy. If instrumenting for application code, the original method body is additionally replaced with a call to the instrumented copy. We originally did this to ensure that we can never fully deviate on an uninstrumented execution path (in the event that we somehow ended up in an uninstrumented call). We think this is probably largely unnecessary now, given that the Android Runtime Android Runtime (ART) runtime will always call the instrumented on a Java callback from native land.

3.1.4 Dynamic instrumentation

We added a hook into ART's class linker for checking to see if the DEX being loaded into the class linker was uninstrumented. We check this simply by verifying that the constant table does not contain an entry for the `"java.lang.TCReturn"` class. If no entry was found, a message is sent via Binder to the instd (instrumentation daemon) process requesting for this DEX to be instrumented. The instd process must operate under root, and fork itself under the uid/gid of the caller to ensure that the requesting application has the correct permissions to access the instrumented DEX that instd will generate. Once instrumented on the device, instd will replace the DEX with the instrumented DEX on the device. Unfortunately, since the FD that we're operating on is already closed at this point, we cannot simply swap out the FD in memory with the new FD. Instead, we must return a status back to the process that forked and execed the dex2oat process, which in this case is installed. The installd process will check this value to determine if the DEX had to be instrumented. If it didn't, then we are good, and we proceed to install the already

instrumented APK. If it did, then we must re-fork and exec the dex2oat process on the newly instrumented DEX/APK. On this second pass the instrumented APK will be installed presuming no error was encountered along the way.

Since Android 8 now has an in-memory DEX class loader, instd also needed to support this. We do so by adding another binder call to this service that passes the bytes of the DEX to instd. It will then write out these bytes to a temporary to instrument to a file readable by the calling pid. Since the original Java native call this was called from already returns some sort of cookie, the most obvious approach here was to pack the bytes from the instrumented DEX into a Java `byte[]`, and put this inside of an exception that was be thrown back into Java. On the Java side, we catch this exception, and replace the contents of the `ByteBuffer` with the bytes obtained from the exception. We then call `DexFile.openInMemoryDexFile()` again on this new `ByteBuffer` to reload the instrumented DEX.

3.1.5 Sources and sinks generation

Sources and sinks are specified and formatted using Google protocol buffers (see **Error! Reference source not found.**). These messages can be programmatically inserted into a call by creating a summary for the call that you wish to treat as a source, sink, or other event. They are also automatically injected into Binder system calls when translating the AIDL into java source code, which we will discuss in the next sub-section.

3.1.5.1 Binder

Unfortunately, we do a really poor job of reporting in Binder. We can lose accuracy and precision in various ways, simply because we've assumed each Parcel is written in a linear fashion, and that's not quite true. Parcel data can be written in a random-access fashion. So, we will eventually have to address these issues.

3.1.5.1.1 How do we support these calls?

- `Parcel.setDataPosition()`: This allows us to set the position anywhere in the parcel, and arbitrarily write over (potentially) another value. We suppose this isn't super important. Worst case is we've overwritten some value with another value, both of which we'll have an EventData object for. Basically, We would just consider this a case of over-reporting.
- `Parcel.marshall()`, `Parcel.unmarshall()`, `Parcel.appendFrom()`: These are much more worrisome, since you can essentially pass off data from one parcel to another parcel, and you would miss that hand-off here, when this maybe should be treated as another link in our prov graph. At the very least, We think we should see if there is an easy fix for these calls. We would essentially need to keep track of the parcel position at each EventData object that we generate for each of the corresponding reads/writes. That will allow us to know which EventData objects will need to be copied over, in the case of `appendFrom()`. For marshalling and unmarshalling, I'm not sure if it's quite so simple. We suppose the Parcel could keep track of a weak reference map of `byte[]` objects corresponding with the event data that was marshalled. Then simply recover this information when you unmarshall by performing a lookup in the weak map.

It turns out there are a lot of core framework classes that require these pieces to work, to precisely "transact" what is going on.

3.1.5.1.2 Special Classes to support

- Binder/BinderProxy
- Bundle
- BaseBundle
- PersistableBundle
- Parcel
- Parcelable (and inner classes, i.e. Creator, ClassLoaderCreator)
- ParcelableSpan

Aside from Parcel, the source code alterations required is pretty straight forward, and minimal.

3.1.5.1.3 Parcel

The motivation here is two-fold. First, we would like to speed up binder reporting a bit by essentially minimizing the number of report calls made on any given Parcel. Also, we would like the events reported on each binder to be an accurate hierarchical representation of each of the calls that we are treating as a source/sink. In addition to the hierarchies, we would like to (ideally) report each EventData to be named/labeled according to the Parcelable's field name, or corresponding AIDL argument name. This way, to an analyst, it is (or should be) much more obvious what the corresponding values mean when you have this hierarchical context attached to the protobuf message.

We therefore try to remedy this problem by making a copy of all Parcel read/write calls (or at least the ones we are interested in reporting on). The copy will contain an additional argument for the label (which will correspond to the name of the EventData generated). For example, some Parcel read/write calls look like this:

```
class Parcel {

    final static String DEFAULT_READ_LABEL = "binder_read";
    final static String DEFAULT_WRITE_LABEL = "binder_write";

    @TCBinder(source = true)
    int readInt(TCReturn ret) {
        return readInt(DEFAULT_READ_LABEL, ret);
    }

    int readInt(String label, TCReturn ret) {
        int val = nativeReadInt(mNativePtr, 0, ret);
        ret.taint = TC.defineProv(ret.getAppPpt(), mSysCall, mProvKind, true, ret.taint);
        mSrcData.peek().add(TC.reportInt(label, val, ret.taint, EventData.SRC));
        return val;
    }

    @TCBinder(source = true)
    String readString(TCReturn ret) {
        return readString(DEFAULT_READ_LABEL, ret);
    }
```

```
   String readString(String label, TCReturn ret) {
      String val = nativeReadString(mNativePtr, 0, ret);
      mSrcData.peek().add(TC.reportNonArrayObject(label, val, String.class, EventData.SRC,
            TCDefiner.joiner(ret.getAppPpt(), mSysCall, mProvKind, true)));
      return val;
   }

   @TCBinder(source = true)
   String[] readStringArray() {
      return readStringArray(DEFAULT_READ_LABEL);
   }

   String[] readStringArray(String label) {
      String[] array = null;

      int length = readIntNR();  // Do not report at all, since we probably don't care
about this.
      if (length >= 0)
      {
         array = new String[length];
         pushObject(array, true /* isSource */, label);
         try {
            for (int i = 0 ; i < length ; i++)
            {
               array[i] = readString(TC.arrayOf(label, i));  // labeled as: label[i], at
each ith EventData object
            }
         } finally {
            popObject(true /* isSource */);
         }
      }

      return array;
   }

   @TCBinder(source = true)
   void readMap(Map outVal, ClassLoader loader) {
      return readMap(outVal, loader, DEFAULT_READ_LABEL);
   }

   void readMap(Map outVal, ClassLoader loader, String label) {
      pushObject(outVal, true /* isSource */, label);
      try {
         int N = readIntNR();  // Do not report at all, since we probably don't care about
this.
         while (N > 0) {
            Object key = readValue(loader, "key");     // labeled as: "key"
            Object value = readValue(loader, "value");  // labeled as: "value"
            outVal.put(key, value);
            N--;
         }
      } finally {
         popObject(true /* isSource */);
      }
   }

   @TCBinder(source = false)
   void writeInt(int val, int val_t, TCReturn ret) {
      writeInt(val, val_t, DEFAULT_WRITE_LABEL);
   }

   void writeInt(int val, int val_t, String label, TCReturn ret) {
      mSinkData.peek().add(TC.reportInt(label, val, val_t, EventData.SINK));
      val_t = TC.defineProv(ret.getAppPpt(), mSysCall, mProvKind, false, val_t);
      nativeWriteInt(mNativePtr, 0, val, val_t, ret);
   }
```

```
   @TCBinder(source = false)
   void writeString(String val, TCReturn ret) {
      writeString(val, DEFAULT_WRITE_LABEL);
   }

   void writeString(String val, String label, TCReturn ret) {
      mSinkData.peek().add(TC.reportNonArrayObject(label, val, String.class,
EventData.SINK));
      nativeWriteString(mNativePtr, val, mProvKind, mSysCall, ret);
   }

   @TCBinder(source = false)
   void writeStringArray(String[] val) {
      writeStringArray(val, DEFAULT_WRITE_LABEL);
   }

   void writeStringArray(String[] val, String label) {
      if (val != null) {
         pushObject(val, false /* isSource */, label);
         try {
            int N = val.length;
            writeIntNR(N);  // Do not report at all, since we probably don't care about
this.
            for (int i=0; i<N; i++) {
               writeString(val[i], TC.arrayOf(label, i));  // labeled as: label[i], at
each ith EventData object
            }
         } finally {
            popObject(false /* isSource */);
         }
      } else {
         writeIntNR(-1);  // Do not report at all, since we probably don't care about this.
      }
   }

   @TCBinder(source = false)
   void writeMap(Map val) {
      writeMap(val, DEFAULT_WRITE_LABEL);
   }

   void writeMap(Map val, String label) {
      if (val == null) {
         writeIntNR(-1); // Do not report at all, since we probably don't care about this.
         return;
      }
      pushObject(val, false /* isSource */, label);
      try {
         Set<Map.Entry<String,Object>> entries = val.entrySet();
         writeIntNR(entries.size());  // Do not report at all, since we probably don't care
about this.
         for (Map.Entry<String,Object> e : entries) {
            writeValue(e.getKey(), "key");     // labeled as: "key"
            writeValue(e.getValue(), "value");  // labeled as: "value"
         }
      } finally {
         popObject(false /* isSource */);
      }
   }
}
```

For anything we wish to not report on at all, we create a copy of the corresponding read/write Parcel call, and make an `"NR"` copy (which essentially propagates the values and tags without reporting them). This is useful for things that are really only used for information flow, and the tag is therefore not necessary, for example. These are things like representing null objects with a

boolean value prior to the object write/read, or for things like array/map sizes used only for the purpose of allocating the correctly sized objects on the other end of the call.

You'll notice that for more complicated Objects such as `java.util.Map`, we essentially represent this as an ObjectValue on the map, containing separate protobuf "value" objects for each of the respective keys and values of the map. In a similar fashion, this enables us to create hierarchies of objects for each parcelable written/read to/from that respective parcel, and each of those can be writing/reading to/from yet another parcelable on that parcel, and so on, and so forth (which we talk about in the next section). Parcel needs to support the following new operations for handling the reports:

- `pushObject(Object val, boolean isSource, String label)`, `pushObject(Object[] val, boolean isSource, String label)`, `pushObject(LongArray val, boolean isSource, String label)`, `pushObject(Collection val, boolean isSource, String label)`, `pushObject(Class val, boolean isSource, String label)`: Simply appends a new EventData with a given label for val (and depending on the type of value, all of it's respective parts), and then pushes an empty EventDataList onto the srcData/sinkData stack (depending on value of isSource).
- `popObject(isSource)`: Simply pops the top EventDataList from the srcData/sinkData stack (depending on the value of isSource), and binds the EventData from the list to the last EventData object written/read from the parent.
- `recycle()`: This is where we will move the reporting on all sources/sinks. If we presume no resource leak in the system, then we can be certain this is the absolute latest we can issue the java report on the sources/sinks attached to this Parcel. This is both simpler than our current approach, and also drastically minimizes the number of native report calls we have to make for each Parcel (in a lot of cases).

3.1.5.1.4 How tags are passed

In general, tags are interleaved with their corresponding data value. The events are generated and reported for each read/write to the Parcel like we discuss in sections 3.1.9.3 and 3.1.9.4. Primitive arrays are optimized for C++ (i.e. so we can perform a memcpy), hence the elements of the array are not interleaved. Instead, they are structured as follows:

1. The length of the array (or -1 if null)
2. The aggregate tag (or -1 if not aggregate)
3. The elements of the array
4. The elements of the corresponding non-aggregate tag array (if aggregate tag == -1)

3.1.5.1.5 User-defined Parcelables

We can take full advantage of Java 8 and create default interface methods to create default implementations for the write/read calls required to fully take advantage of this framework.

```
interface android.os.Parcelable {

    ...

    /**
     * Flatten this object in to a Parcel.
     *
     * @param dest The Parcel in which the object should be written.
     * @param flags Additional flags about how the object should be written.
     * May be 0 or {@link #PARCELABLE_WRITE_RETURN_VALUE}.
     */
    @TCBinder(type = Type.ROOT, source = false)
    public void writeToParcel(Parcel dest, @WriteFlags int flags);

    /**
     * Flatten this object in to a Parcel.
     *
     * @param dest The Parcel in which the object should be written.
     * @param flags Additional flags about how the object should be written.
     * May be 0 or {@link #PARCELABLE_WRITE_RETURN_VALUE}.
     */
    public default void writeToParcel(Parcel dest, @WriteFlags int flags, String label) {
        dest.pushObject(this, false /* isSource */, label);
        try {
            writeToParcel(dest, flags);
        } finally {
            dest.popObject(false /* isSource */);
        }
    }

    /**
     * Interface that must be implemented and provided as a public CREATOR
     * field that generates instances of your Parcelable class from a Parcel.
     */
    public interface Creator<T> {
        /**
         * Create a new instance of the Parcelable class, instantiating it
         * from the given Parcel whose data had previously been written by
         * {@link Parcelable#writeToParcel Parcelable.writeToParcel()}.
         *
         * @param source The Parcel to read the object's data from.
         * @return Returns a new instance of the Parcelable class.
         */
        @TCBinder(type = Type.ROOT, source = true)
        public T createFromParcel(Parcel source);

        /**
         * Create a new instance of the Parcelable class, instantiating it
         * from the given Parcel whose data had previously been written by
         * {@link Parcelable#writeToParcel Parcelable.writeToParcel()}.
         *
         * @param source The Parcel to read the object's data from.
         * @return Returns a new instance of the Parcelable class.
         */
        public default T createFromParcel(Parcel source, String label) {
            // TODO: Ideally we would like to operate on the T returned here
            //      and not 'this', but it would require pulling the parcel
            //      out of T's constructor in a lot of cases, which would
            //      require some analysis here.
            dest.pushObject(this, true /* isSource */, label);
            try {
                return createFromParcel(source);
            } finally {
                dest.popObject(true /* isSource */);
            }
        }
```

```
        ...
    }

    /**
     * Specialization of {@link Creator} that allows you to receive the
     * ClassLoader the object is being created in.
     */
    public interface ClassLoaderCreator<T> extends Creator<T> {
        /**
         * Create a new instance of the Parcelable class, instantiating it
         * from the given Parcel whose data had previously been written by
         * {@link Parcelable#writeToParcel Parcelable.writeToParcel()} and
         * using the given ClassLoader.
         *
         * @param source The Parcel to read the object's data from.
         * @param loader The ClassLoader that this object is being created in.
         * @return Returns a new instance of the Parcelable class.
         */
        @TCBinder(type = Type.ROOT, source = true)
        public T createFromParcel(Parcel source, ClassLoader loader);

        /**
         * Create a new instance of the Parcelable class, instantiating it
         * from the given Parcel whose data had previously been written by
         * {@link Parcelable#writeToParcel Parcelable.writeToParcel()} and
         * using the given ClassLoader.
         *
         * @param source The Parcel to read the object's data from.
         * @param loader The ClassLoader that this object is being created in.
         * @return Returns a new instance of the Parcelable class.
         */
        public default T createFromParcel(Parcel source, ClassLoader loader, String label) {
            // TODO: Ideally we would like to operate on the T returned here
            //      and not 'this', but it would require pulling the parcel
            //      out of T's constructor in a lot of cases, which would
            //      require some analysis here.
            dest.pushObject(this, false /* isSource */, label);
            try {
                return createFromParcel(source, loader);
            } finally {
                dest.popObject(false /* isSource */);
            }
        }
    }

}
```

This is a excellent way to preserve a natural hierarchy of Parcelables in your binder calls without having to add any additional code for all of the child classes! This works great so long as the application is AIDL generated, and the SDK is one that supports this default interface method implementation (i.e. SDK version >= 24). For everything else, sadly we must go in and physically hand-code this on System applications that are built below this SDK requirement. Having said this, there were not too many to need to go through by hand. We suspect you could write a Java AST parser fairly easily to do an adequate enough job, if there were quite a number of these. Alternatively, I've marked "root" @TCBinder calls, as well as @TCBinder report calls. Perhaps we can use these to construct call paths that would automatically insert these methods and ensure calls with the "label" argument exist for any method call that leaves with that Parcel object.

3.1.5.1.6 AIDL Translation

Since the AIDL files contain the names of the data items that are read/written to the parcel for each call, we can automatically insert the correct Parcel calls (with the data and label) for each of the AIDL calls specified in the file. This allows us to construct these more detailed Binder prov messages for free, without having to manually write it out by hand.

EXAMPLE:

Consider setting the test provider location with the location manager.

Listing: Location.aidl

```
package android.location;

parcelable Location;
```

Listing: Location.java

```
public class Location implements Parcelable {

   ...

   public static final Parcelable.Creator<Location> CREATOR = new
Parcelable.Creator<Location>() {
      @Override
      @TCBinder(source = false)
      public Location createFromParcel(Parcel in) {
         return createFromParcel(in, Parcel.DEFAULT_READ_LABEL);
      }

      @Override
      public Location createFromParcel(Parcel in, String label) {
         Location l = new Location();
         in.pushObject(l, true /* isSource */, label);
         try {
            l.mProvider = in.readString("mProvider");
            l.mTime = in.readLong("mTime");
            l.mElapsedRealtimeNanos = in.readLong("mElapsedRealtimeNanos");
            l.mFieldsMask = in.readByte("mFieldsMask");
            l.mLatitude = in.readDouble("mLatitude");
            l.mLongitude = in.readDouble("mLongitude");
            l.mAltitude = in.readDouble("mAltitude");
            l.mSpeed = in.readFloat("mSpeed");
            l.mBearing = in.readFloat("mBearing");
            l.mHorizontalAccuracyMeters = in.readFloat("mHorizontalAccuracyMeters");
            l.mVerticalAccuracyMeters = in.readFloat("mVerticalAccuracyMeters");
            l.mSpeedAccuracyMetersPerSecond =
in.readFloat("mSpeedAccuracyMetersPerSecond");
            l.mBearingAccuracyDegrees = in.readFloat("mBearingAccuracyDegrees");
            l.mExtras = Bundle.setDefusable(in.readBundle("mExtras"), true);
         } finally {
            in.popObject(true /* isSource */);
         }
         return l;
      }
```

```
        @Override
        public Location[] newArray(int size) {
            return new Location[size];
        }
    };

    @Override
    public void writeToParcel(Parcel out, int flags) {
        out.pushObject(this, false /* isSource */, label);
        try {
            out.writeString(mProvider, "mProvider");
            out.writeLong(mTime, "mTime");
            out.writeLong(mElapsedRealtimeNanos, "mElapsedRealtimeNanos");
            out.writeByte(mFieldsMask, "mFieldsMask");
            out.writeDouble(mLatitude, "mLatitude");
            out.writeDouble(mLongitude, "mLongitude");
            out.writeDouble(mAltitude, "mAltitude");
            out.writeFloat(mSpeed, "mSpeed");
            out.writeFloat(mBearing, "mBearing");
            out.writeFloat(mHorizontalAccuracyMeters, "mHorizontalAccuracyMeters");
            out.writeFloat(mVerticalAccuracyMeters, "mVerticalAccuracyMeters");
            out.writeFloat(mSpeedAccuracyMetersPerSecond, "mSpeedAccuracyMetersPerSecond");
            out.writeFloat(mBearingAccuracyDegrees, "mBearingAccuracyDegrees");
            out.writeBundle(mExtras, "mExtras");
        } finally {
            out.popObject(false /* isSource */);
    }

}
```

Listing: ILocationManager.aidl

```
interface ILocationManager
{
    ...

    void setTestProviderLocation(String provider, in Location loc);

    ...
}
```

AIDL OUTPUT:

Considering the AIDL for the example of ILocationManager.setTestProviderLocation(), the aidl tool will generate code that resembles the following:

Listing: AIDL-translated ILocationManager.java

```
package android.location;

/**
 * System private API for talking with the location service.
 *
 * @hide
 */
public interface ILocationManager extends android.os.IInterface {

    ...
```

```
   /** Local-side IPC implementation stub class. */
   public static abstract class Stub extends android.os.Binder implements
android.location.ILocationManager {

       ...

       @Override
       public boolean onTransact(int code, android.os.Parcel data, android.os.Parcel reply,
int flags)
               throws android.os.RemoteException {
           switch (code) {

           ...

           case TRANSACTION_setTestProviderLocation: {
               data.setSysCall("void
               android.location.ILocationManager$Stub.
setTestProviderLocation(java.lang.String provider, android.location.Location loc) [boolean
android.location.ILocationManager$Stub.onTransact(int code, android.os.Parcel data,
android.os.Parcel reply, int flags)]");
               data.enforceInterface(DESCRIPTOR);
               java.lang.String _arg0;
               _arg0 = data.readString("provider");
               android.location.Location _arg1;
               if ((0 != data.readIntNR())) {
                   _arg1 = android.location.Location.CREATOR.createFromParcel(data, "loc");
               } else {
                   _arg1 = null;
               }
               this.setTestProviderLocation(_arg0, _arg1);
               reply.setProvKind(TCReport.BinderObject.LOCATION);
               reply.setSysCall("void
               android.location.ILocationManager$Stub.
setTestProviderLocation(java.lang.String provider, android.location.Location loc) [boolean
android.location.ILocationManager$Stub.onTransact(int code, android.os.Parcel data,
android.os.Parcel reply, int flags)]");
               reply.writeNoException();
               return true;
           }

           ...

           }
           return super.onTransact(code, data, reply, flags);
       }

       private static class Proxy implements android.location.ILocationManager {

           ...

           @Override
           public void setTestProviderLocation(java.lang.String provider,
android.location.Location loc)
                   throws android.os.RemoteException {
               android.os.Parcel _data = android.os.Parcel
                       .obtain(TCReport.BinderObject.LOCATION,
                               "void
               android.location.ILocationManager$Stub$Proxy.
setTestProviderLocation(java.lang.String
               provider, android.location.Location loc)");
               android.os.Parcel _reply = android.os.Parcel
                       .obtain(TCReport.BinderObject.BINDER,
                               "void
               android.location.ILocationManager$Stub$Proxy.
setTestProviderLocation(java.lang.String
               provider, android.location.Location loc)");
```

```
            try {
                _data.writeInterfaceToken(DESCRIPTOR);
                _data.writeString(provider, "provider");
                if ((loc != null)) {
                    _data.writeIntNR(1);
                    loc.writeToParcel(_data, 0, "loc");
                } else {
                    _data.writeIntNR(0);
                }
                mRemote.transact(Stub.TRANSACTION_setTestProviderLocation, _data, _reply,
0);
                _reply.readException();
            } finally {
                _reply.recycle();
                _data.recycle();
            }
        }
    }

    ...

    public void setTestProviderLocation(java.lang.String provider, android.location.Location
loc)
          throws android.os.RemoteException;

    ...
}
```

Listing: Prov Output

```
define_sys_call <
  id: 7
  prog_id: 4
  value: "**BINDER** void
android.location.ILocationManager$Stub$Proxy.setTestProviderLocation(java.lang.String
provider, android.location.Location loc)"
>
define_app_ppt <
  id: 8
  prog_id: 4
  value: ...
>
define_prov <
  flow: 1
  id: 9
  prog_id: 4
  type: 178
  app_ppt: 8
  sys_call: 7
  prev_id: 0
>
event <
  flow: SINK
  prog_id: 4
  app_ppt: 8
  sys_call: 7
  tid: 856
  time: 1556735811185
  event_data <
    name: "provider"
    value_type: SINK
    is_array: false
```

```
    tag <
      ...
    >
    is_null: false
    string_value <
      ...
    >
  >
  event_data <
    name: "loc"
    value_type: SINK
    is_array: false
    tag <
      ...
    >
    is_null: false
    object_value <
      type: "android.location.Location"
      ...
      event_data <
        name: "mProvider"
        ...
        string_value <
          ...
        >
      >
      event_data <
        name: "mTime"
        long_value <
          ...
        >
      >
      ...
    >
  >
>


.....  Some time passes  .....


define_sys_call <
  id: 57
  prog_id: 54
  value: "**BINDER** void
android.location.ILocationManager$Stub.setTestProviderLocation(java.lang.String provider,
android.location.Location loc) [boolean
android.location.ILocationManager$Stub.onTransact(int code, android.os.Parcel data,
android.os.Parcel reply, int flags)]"
>
define_app_ppt <
  id: 58
  prog_id: 54
  value: "boolean android.os.Binder.execTransact(int code, long dataObj, long replyObj, int
flags) [line: 443]"
>
define_prov <
  flow: 0
  id: 59
  prog_id: 84
  type: 178
  app_ppt: 58
  sys_call: 57
  prev_id: 0
>
event <
  flow: SRC
```

```
  prog_id: 12
  app_ppt: 8
  sys_call: 6
  tid: 912
  time: 1556735811186
  event_data <
    name: "provider"
    value_type: SRC
    is_array: false
    tag <
      ...
    >
    is_null: false
    string_value <
      ...
    >
  >
  event_data <
    name: "loc"
    value_type: SRC
    is_array: false
    tag <
      ...
    >
    is_null: false
    object_value <
      type: "android.location.Location"
      ...
      event_data <
        name: "mProvider"
        ...
        string_value <
          ...
        >
      >
      event_data <
        name: "mTime"
        long_value <
          ...
        >
      >
      ...
    >
  >
>
```

Obviously, this was at least partially hand generated, so I've elided over some of the details in the output.

3.1.5.2 Linux.java / Posix.java

Events and trivial sources (i.e. Single source events, where the returned value is the source) are automatically injected at instrumentation time. They are specified by calls in the provenance.stubs file (see **Error! Reference source not found.**).

Listing: provenance.stubs

```
libcore.io.Linux {
  boolean @src access(java.lang.String @apred,int);
  void chmod(java.lang.String @apred,int);
  void chown(java.lang.String @apred,int,int);
  void execve(java.lang.String,java.lang.String[],java.lang.String[]);
```

```
  void execv(java.lang.String,java.lang.String[]);
  void fchmod(java.io.FileDescriptor @bpred,int);
  void fchown(java.io.FileDescriptor @bpred,int,int);
  int @src getgid();
  int @src getpid();
  ...
}
```

The provenance.stubs files support the following type of "annotations" on the parameters and return type:

- `@src` - Specifies the value that should be marked as the source to this event (default Binder type is POSIX).
- `@apred` - Marks which parameter is the file predicate (to be computed after the call).
- `@apred_cached` - Like `@apred`, but the prov type for this predicate is cached.
- `@bpred` - Marks which parameter is the file predicate (to be computed before the call).
- `@bpred_cached` - Like `@bpred`, but the prov type for this predicate is cached.

For all other non-trivial sources, sinks, and events, we must handle this in a summarized call in `libcore.io.Linux`. For example...

```
public final class Linux implements Os {

   @TCDontShadow
   public native long lseek(FileDescriptor fd, long offset, int whence) throws
ErrnoException;

   @TCDontShadow
   @TCTransparent
   public long lseek(FileDescriptor fd, long offset, int offset_t, int whence, int
whence_t, TCReturn ret) throws ErrnoException {
       long nBytes = lseek(fd, offset, whence);
       if (!TC.programStarted()) {
          ret.taint = 0;
          return nBytes;
       }

       int sysCall = TC.defineSysCall("long libcore.io.Linux.lseek(FileDescriptor fd, long
offset, int whence) [line: 168]");

       FileDescriptor taintFd = fd.getTaintFd();
       if (taintFd != null) {
          // Seek the tag file at four times the offset (to account for tag size).
          lseek(taintFd, offset*4, whence);
       }

       EventData[] data = {
          TC.reportNonArrayObject("fd", fd, FileDescriptor.class, EventData.PARAM),
          TC.reportLong("offset", offset, offset_t, EventData.PARAM),
          TC.reportInt("whence", whence, whence_t, EventData.PARAM),
          TC.reportLong("r", nBytes, 0, EventData.RET)
       };
       int pred = TC.toPredicate(fd, true);
       TC.event(ret.getAppPpt(), sysCall, data, pred);

       ret.taint = 0;
       return nBytes;
   }
```

```
   @TCDontShadow
   public native FileDescriptor dup(FileDescriptor oldFd) throws ErrnoException;

   @TCDontShadow
   @TCTransparent
   public FileDescriptor dup(FileDescriptor oldFd, TCReturn ret) throws ErrnoException {
      // Generate the event...
      int idx = 0;
      EventData[] data = new EventData[2];
      int sysCall = TC.defineSysCall("FileDescriptor libcore.io.Linux.dup(FileDescriptor
oldFd) [line: 217]");
      data[idx++] = TC.reportNonArrayObject("oldFd", oldFd, FileDescriptor.class,
EventData.PARAM);
      int oldPathPred = TC.toPredicate(oldFd, true);

      FileDescriptor newFd = dup(oldFd);

      int newPathPred = TC.toPredicate(newFd, false);
      data[idx++] = TC.reportNonArrayObject("r", newFd, FileDescriptor.class,
EventData.RET);
      TC.event(ret.getAppPpt(), sysCall, data, oldPathPred, newPathPred);

      FileDescriptor fd_t = oldFd.getTaintFd();
      if (fd_t != null)
         newFd.setTaintFd(dup(fd_t));
      newFd.setRemoteId(oldFd.getRemoteId());

      return newFd;
   }

}
```

We make use of the `@TCDontShadow` annotation to indicate that the instrumenter should not touch this method. This is done on both the original and instrumented copy, since we provide our own instrumentation for this call. Note that additionally the instrumented copy should also be annotated by `@TCTransparent` to ensure that this method is invisible to reflection.

3.1.6 JNI interface

We modified the interface itself by adding new function calls to the end of the JNINativeInterface struct.

```
struct JNINativeInterface {
   ...

   //
   // Clearscope/TC functions
   //

   // Tag array access functions:
   jtag*       (*GetTagArrayElements)(JNIEnv *, jtagArray, jboolean *);
   void        (*ReleaseTagArrayElements)(JNIEnv *, jtagArray, jtag *, jint);
   void        (*GetTagArrayRegion)(JNIEnv *, jtagArray, jsize, jsize, jtag *);
   void        (*SetTagArrayRegion)(JNIEnv *, jtagArray, jsize, jsize, const jtag *);

   // Array/String taint access functions:
   jtag        (*GetAggregateTaint)(JNIEnv*, jarray);
   jtagArray    (*GetArrayTaint)(JNIEnv*, jarray);
   void        (*SetAggregateTaint)(JNIEnv*, jarray, jtag);
   jtagArray    (*Unaggregate)(JNIEnv*, jarray);
   jtag        (*GetStringAggregateTaint)(JNIEnv*, jstring);
```

```
   jtagArray   (*GetStringArrayTaint)(JNIEnv*, jstring);
   void       (*SetStringAggregateTaint)(JNIEnv*, jstring, jtag);
   jtagArray   (*UnaggregateString)(JNIEnv*, jstring);

   // java.lang.TCReturn field access functions:
   void       (*SetReturnTaint)(JNIEnv*, jreturn, jtag);
   jstring     (*GetPpt)(JNIEnv*, jreturn);

   // ProvMsgr/binder functions:
   jtag       (*DefineAppPpt)(JNIEnv*, const char*);
   jtag       (*DefineSysCall)(JNIEnv*, const char*);
   jtag       (*DefineProvType)(JNIEnv *, jsize);
   jtag       (*DefineProvPipeType)(JNIEnv *, const char*);
   jtag       (*DefineProvFileType)(JNIEnv *, const char*, jchar, jint, jint);
   jtag       (*DefineProvNetworkType)(JNIEnv *, jint, const char*, jint, const char*,
jint, jint);

   jtag       (*DefineProv)(JNIEnv*, const char*, const char*, jsize, jboolean, jtag);
   jtag       (*DefineProvUsingTag)(JNIEnv*, jtag, jtag, jtag, jboolean, jtag);
   jtag       (*DefineProvSet)(JNIEnv*, jtag);
   void       (*ReportMsg)(JNIEnv*, uint8_t*, jint);

   // Tag generation functions:
   jtag       (*NewTag)(JNIEnv*);
   jtag       (*JoinTaint)(JNIEnv*, jtag, jtag);
   jtag       (*JoinTaintUnion)(JNIEnv*, jtag, jtag);

   // Taint bridge functions:
   void       (*PushTaint)(JNIEnv*, const jtag*, const char*, jint);
   jtag       (*PopTaint)(JNIEnv*);
   jtaintbridge (*CurrentTaintBridge)(JNIEnv*);

   // System call functions for accessing the metadata file descriptor:
   void (*SyscallSetFdTaintFd)(JNIEnv*, jint, jint);
   void (*SyscallSetFdRemoteID)(JNIEnv*, jint, const char*);
   void (*SyscallSetFdProvTypeID)(JNIEnv *, jint, jtag);

   jint (*SyscallGetFdTaintFd)(JNIEnv*, jint);
   const char* (*SyscallGetFdRemoteID)(JNIEnv*, jint);
   jtag (*SyscallGetFdProvTypeID)(JNIEnv *, jint, jboolean);
   void (*SyscallFdCleanup)(JNIEnv*, jint);
}
```

These functions are utilized on the native side to both summarize Java native calls that are easy to summarize, and also as entry points for LAM (lean and mean). In addition to this we add another struct for communicating the tags, system call id, and application program point id, returned tag, etc...

```
typedef struct {
   uint8_t is_java;
   jtag app_ppt;
   const char* shorty;
   struct {
      jtag* taint;
   } params;
   struct {
      jtag taint;
   } ret;
   long scratch; /* XXX */
} taint_bridge_t;

typedef taint_bridge_t* jtaintbridge;
```

3.1.6.1 Native methods

Java native methods when instrumented are represented as a native method in Java, though it's not actually backed by any native code. Instead, we correct the call at runtime, such that the native code can be located and executed for each native call. We accomplish this by first skipping Dalvik-to-native compilation in dex2oat for all Java native methods. This is necessary because otherwise a bridge to the native routine will be compiled into the binary. So, by skipping this we divert the call through the Java Native Interface (JNI) trampoline. Inside of the JNI trampoline we do the following:

1. Pop arguments and respective tags from the current stack frame, pushing the arguments back onto the stack (since this is what the original method will be expecting on the stack).
2. Replace the `*sp` that points to the current `ArtMethod*` with the original (uninstrumented) `ArtMethod*` (since the original is bound to the actual native routine).
3. Push a new `taint_bridge_t` struct containing the tags we popped from Step 1, as well as the other information LAM is expecting for the call.
4. Make the native call.
5. Pop the current taint bridge, and record the returned tag from this struct onto this thread's TCReturn object.

Note that for any call that provides a native summary we must actually call the original native routine since a) we've obviously provided a native implementation in this case, and b) this call should not be running under LAM, since this summary will handle the tag propagation to and from Java. Also, there are certain system calls that we wish not to intervene on in LAM. To accommodate for this, we have added two special method modifiers (as bitwise-flags to the methods access value). The first is used to mark system methods that will not run under LAM. The second is for specifying whether the tags should be stripped or not.

3.1.7 Reflection

There are two things to consider for supporting reflection. First, it needs to be transparent to the application. For example, all of the fields, constructors, methods that we add to the system/application needs to be invisible to reflection. We support this by marking each of these new elements we add with the `TCTransparent` annotation. Second, methods invoked (or objects constructed) at runtime need to operate on the instrumented method. Likewise, fields set reflectively also need to set the tag field as well. We need to therefore consider each of the following cases:

- `java.lang.Class`
  - `forName()` - Throw a ClassNotFoundException() if the class returned is @TCTransparent.
  - `getConstructors()`, `getDeclaredConstructors()` - Return all constructors not annotated by @TCTransparent.
  - `getAnnotations()`, `getDeclaredAnnotations()` - Return all annotations not annotated by @TCTransparent (including @TCTransparent itself).

    - `getField()`, `getDeclaredField()` - Throw a NoSuchFieldException if the field returned is @TCTransparent.
    - `getFields()`, `getDecalredFields()` - Return all fields not annotated by @TCTransparent.
    - `getMethods()`, `getDeclaredMethods()` - Return all fields not annotated by @TCTransparent.
    - `getConstructor()`, `getDeclaredConstructor()`, `getMethod()`, `getDeclaredMethod()` - Throw a NoSuchMethodException() if the constructor/method returned is @TCTransparent.
    - `newInstance()` - See `Constructor.newInstance()`.
- `java.lang.reflect.Field`
    - `get()` - If the underlying field is a primitive, then box the value of the field with the value of the corresponding tag field, and return the boxed primitive.
    - `getBoolean()`, `getByte()`, `getChar()`, `getShort()`, `getInt()`, `getFloat()`, `getLong()`, `getDouble()` - Retrieves the tag for the corresponding tag field, sets `TCReturn.taint` with this value, and returns the value of the underlying primitive field.
    - `set()` - If the underlying field is a primitive, then unbox the value and tag, and set both the primitive field and corresponding tag field with these values.
    - `setBoolean()`, `setByte()`, `setChar()`, `setShort()`, `setInt()`, `setFloat()`, `setLong()`, `setDouble()` - Sets both the primitive field and corresponding tag field from the value and tags (respectively) passed into the call.
- `java.lang.reflect.Method`
    - `invoke()` - This is handled natively to support native "reflective" calls made via JNI. The primitive arguments passed to a call need to be unboxed (both the value, and tag). While unwinding the arguments in this fashion, we construct the new arguments to the instrumented call. We then invoke the instrumented method on the new set of arguments. If this was called from the JNI while tracing with LAM, then we additionally need to get the current taint bridge and propagate the tag return from the method invocation. We get this by obtaining this thread TCReturn object, and accessing the taint field.
- `java.lang.reflect.Constructor`
    - `newInstanceof()` - This is essentially turns into the same underlying native call as `Method.invoke()`.

3.1.8 Proxy Classes

While like reflection, proxy classes are actually a bit different, since a method invoked on a proxy class appears like any other method call. What this means is that unlike reflection, the primitive

arguments to the call are not boxed. We therefore need to correct for this in the native entry point for `InvocationHandler.invoke()` as follows:

1. Strip all of the tag parameters from the proxy method call, and box all of primitive arguments from their (value,tag) pair, then reconstruct a new Java object array with the arguments (as they would appear in an uninstrumented call).
2. Call `InvocationHandler.invoke()` passing it this new object array with the corrected arguments, as well as the Method object of the original call.
3. If the underlying method returns a primitive, we need to additionally unbox the primitive return value and corresponding tag. `TCReturn.taint` will be set to the tag we unboxed from this current threads TCReturn object.

Now that the implementation of `InvocationHandler.invoke()` will receive the original method and boxed arguments with taint to the call, this is exactly what we want since there will almost certainly be a call to `Method.invoke()` here on this Method and arguments. This means that reflection will actually operate on the instrumented method copy, and will unbox the (value,tag) pair that we had previously boxed for the proxy call, and everything should just work as expected on the actual instrumented call.

3.1.9 Reporting

Event data can be natively reported on all of the Java primitive values, arrays, strings, as well as other Java objects. These calls can be found in `java.lang.TC` in each of the static `report*()` methods, and each return an EventData object. In `java.lang.Object`, we added the following method to handle the reporting for general objects:

```
public EventData report(String name, int dataType, TCDefiner definer) {
    EventData data = new EventData();
    ObjectValue objValue = new ObjectValue();
    objValue.type = this.getClass().getQuickName();
    objValue.hashCode = System.identityHashCode(this);
    data.name = name;
    data.valueType = dataType;
    data.setObjectValue(objValue);
    return data;
}
```

This can be overridden to include fields that you wish to report on, but above is the default implementation. Alternatively, you can alter the `classes.defs` file, and this will tell the instrumenter how these Objects will be reported. The instrumenter will then automatically generate the `report()` method for these classes at instrumentation time.

Listing: Example of classes.def

```
java.io.File               : (path)
java.io.FileDescriptor       : all
java.lang.Class             : (getName())
java.net.InetAddress         : (family, ipaddress, hostName)
```

In this example, `File` only reports on the `path` field, `FileDescriptor` reports on all fields, `Class` reports on the String returned from the call to `Class.getName()`, and `InetAddress` reports on `family`, `ipaddress`, and `hostName` fields.

3.1.9.1 Protocol Buffers

As mentioned earlier in the paper, we rely on Google protocol buffers to generate report for all sources/sinks and other events that incur in the system. Below is the specification for how provenance messages are defined in ClearScope.

Listing: Protocol Buffers provenance message declarations

```
package art;
option java_package = "java.lang";
option optimize_for = SPEED;
option java_outer_classname = "TCReport";

message RLETag {
    // no length means aggregate
    optional int32 length = 1 [default = -1];
    required uint32 tag = 2;
}

/*************************
 * Data types for events *
 *************************/

message PointerValue {
    required uint64 value = 1;
}

message BoolValue {
    repeated bool value = 1;
}

message ByteValue {
    required bytes value = 1;
}

message CharValue {
    required string value = 1;
}

message ShortValue {
    repeated int32 value = 1;
}

message IntValue {
    repeated int32 value = 1;
}

message FloatValue {
    repeated float value = 1;
}

message LongValue {
    repeated int64 value = 1;
}

message DoubleValue {
    repeated double value = 1;
```

```
}

message StringValue {
   repeated string value = 1;
}

message ObjectValue {
   required string type = 1;
   optional int32 hash_code = 2 [default = 0];  // Could be a null ref.
   repeated EventData value = 3;
}

message EventData {
   enum ValueType {
      PARAM = 0;
      SRC = 1;
      SINK = 2;
      RET = 3;
   }
   required string name = 1;
   required ValueType value_type = 2;
   optional bool is_array = 3 [default = false];
   oneof data {
      BoolValue bool_value = 4;
      ByteValue byte_value = 5;
      CharValue char_value = 6;
      ShortValue short_value = 7;
      IntValue int_value = 8;
      FloatValue float_value = 9;
      LongValue long_value = 10;
      DoubleValue double_value = 11;
      StringValue string_value = 12;
      ObjectValue object_value = 13;
      PointerValue pointer_value = 16;
   }
   repeated RLETag tag = 14;
   optional bool is_null = 15 [default = false];
}

message GeneralObject {
   required string type = 1;
   repeated StringPair properties = 2;
}

message BinderObject {
   enum ProvKind {
      ACCESSIBILITY_SERVICE = 0;
      ACTIVITY_MANAGEMENT = 1;
      ALARM_SERVICE = 2;
      ANDROID_AUTO = 93;
      ANDROID_RADIO = 97;
      ANDROID_TV = 3;
      ANDROID_VR = 99;
      AUDIO_IO = 4;
      AUTOFILL = 98;
      BACKUP_MANAGER = 5;
      BINDER = 6;
      BLUETOOTH = 7;
      BOOT_EVENT = 8;
      BROADCAST_RECEIVER_MANAGEMENT = 9;
      CAMERA = 10;
      CLIPBOARD = 11;
      COMPANION_DEVICE = 100;
      COMPONENT_MANAGEMENT = 12;
      CONTENT_PROVIDER = 13;
      CONTENT_PROVIDER_MANAGEMENT = 14;
      DATABASE = 15;
```

```
DEVICE_ADMIN = 16;
DEVICE_SEARCH = 17;
DEVICE_USER = 18;
DISPLAY = 19;
DROPBOX = 20;
EMAIL = 21;
EXPERIMENTAL = 22;
FILE = 23;
FILE_SYSTEM = 24;
FILE_SYSTEM_MANAGEMENT = 25;
FINGERPRINT = 26;
FLASHLIGHT = 27;
GATEKEEPER = 28;
HDMI = 29;
IDLE_DOCK_SCREEN = 30;
IMS = 31;
INFRARED = 32;
INSTALLED_PACKAGES = 33;
JSSE_TRUST_MANAGER = 34;
KEYCHAIN = 35;
KEYGUARD = 36;
LOCATION = 37;
LOWPAN = 92;
MACHINE_LEARNING = 38;
MBMS = 89;
MEDIA = 39;
MEDIA_CAPTURE = 40;
MEDIA_LOCAL_MANAGEMENT = 41;
MEDIA_LOCAL_PLAYBACK = 42;
MEDIA_NETWORK_CONNECTION = 43;
MEDIA_REMOTE_PLAYBACK = 44;
MIDI = 45;
NATIVE = 46;
NETWORK = 47;
NETWORK_MANAGEMENT = 48;
NFC = 49;
NOTIFICATION = 50;
OVERLAY_MANAGER = 96;
PAC_PROXY = 51;
PERMISSIONS = 52;
PERSISTANT_DATA = 53;
POSIX = 54;
POWER_MANAGEMENT = 55;
PRINT_SERVICE = 56;
PROCESS_MANAGEMENT = 57;
QUICK_SETTINGS = 90;
RCS = 94;
RECEIVER_MANAGEMENT = 58;
RPC = 59;
SCREEN_AUDIO_CAPTURE = 60;
SERIAL_PORT = 61;
SERVICE_CONNECTION = 62;
SERVICE_MANAGEMENT = 63;
SHORTCUTS = 88;
SMS_MMS = 64;
SPEECH_INTERACTION = 65;
STATUS_BAR = 66;
SYNC_FRAMEWORK = 67;
SYSTEM_UPDATE = 91;
TASK_STACK = 95;
TELEPHONY = 68;
TEST = 69;
TEXT_SERVICES = 70;
THREADING = 71;
TIME_EVENT = 72;
UI = 73;
UID_EVENT = 74;
```

```
        UI_AUTOMATION = 75;
        UI_MODE = 76;
        UI_RPC = 77;
        USAGE_STATS = 78;
        USB = 79;
        USER_ACCOUNTS_MANAGEMENT = 80;
        USER_INPUT = 81;
        VIBRATOR = 82;
        WAKE_LOCK = 83;
        WALLPAPER_MANAGER = 84;
        WAP = 85;
        WEB_BROWSER = 86;
        WIDGETS = 87;
    }
    required ProvKind kind = 1;
    repeated StringPair properties = 2;
}

message FileObject {
    required string path = 1;
    required int32 permissions = 2;
    required string type = 3;
    optional int64 sizeInBytes = 4 [default = 0];    //at open
}

message NetworkObject {
    required string localAddress = 1;
    required int32 localPort = 2;
    optional string remoteAddress = 3 [default = ""];
    optional int32 remotePort = 4 [default = -1];
    required int32 protocol = 5;
    optional int32 initTcpSeqNum = 6;
}

message IPCObject {
    enum IPCObjectType {
        IPC_OBJECT_PIPE_NAMED = 0;
        IPC_OBJECT_PIPE_UNNAMED = 1;
        IPC_OBJECT_SOCKET_ABSTRACT = 2;
        IPC_OBJECT_SOCKET_PATHNAME = 3;
        IPC_OBJECT_SOCKET_UNNAMED = 4;
        IPC_OBJECT_SOCKET_NETLINK = 5;
    }

    required IPCObjectType type = 10;
    required string uniqueID = 11;
}

//deprecated
message PipeObject {
    required string uniqueID = 3;
}

message PacketSocketObject {
    required int32 protocol = 1;
    required int32 ifindex = 2;
    required int32 hatype = 3;
    required int32 pkttype = 4;
    required int32 halen = 5;
    required bytes addr = 6;
}

message Event {
    enum Flow {
        EVENT = 0;
        SRC = 1;
        SINK = 2;
```

```
    }
    required Flow flow = 1;
    required uint32 prog_id = 2;
    //java user program point
    required uint32 app_ppt = 3;
    //string signature of java call or native sys-call
    required uint32 sys_call = 4;
    required int64 tid = 6;
    required int64 time = 7;
    repeated EventData eventData = 8;
    optional uint32 predicate1_id = 9;    // prov type id
    optional uint32 predicate2_id = 10;   // prov type id
    //native library and offset
    optional uint32 native_ppt = 11;
    //string signature of java native method
    optional uint32 java_native_call = 12;

    optional uint32 taint_union_entr_id = 13;
    optional uint32 taint_union_exit_id = 14;
}

message DefineProgram {
    required uint32 id = 1;
    required uint32 host_id = 2;
    required string pname = 3;
    required int32 pid = 4;
    required int32 ppid = 5;
    required int32 uid = 6;              //user id assigned by OS
    required int64 start_time = 7;
}

message DefineProvType {
    required uint32 id = 2;
    required uint32 prog_id = 3;
    //required string value = 4;
    oneof object {
        GeneralObject generalObj = 5;
        FileObject fileObj = 6;
        NetworkObject networkObj = 7;
        PacketSocketObject packetSockObj = 8;
        // PipeObject should no longer be used, keeping for backwards
        // compatability, see IPCObject
        PipeObject pipeObj = 9 [deprecated=true];
        BinderObject binderObj = 10;
        IPCObject ipcObject = 11;
    }
}

message DefineAppPpt {
    required uint32 id = 1;
    required uint32 prog_id = 2;
    required string value = 3;
}

message DefineSysCall {
    required uint32 id = 1;
    required uint32 prog_id = 2;
    required string value = 3;
}

message DefineUnknownProv {
   required uint32 id = 1;
   required uint32 prog_id = 2;
}

message DefineProv {
    enum Flow {
```

```
        SRC = 0;
        SINK = 1;
    }

    required Flow flow = 1;
    required uint32 id = 2;
    required uint32 prog_id = 3;
    required int32 type = 4;
    required uint32 app_ppt = 5;
    required uint32 sys_call = 6;
    required uint32 prev_id = 7;
    optional string prev_device_id = 8;
}


message DefineProvSet {
    required uint32 id = 1;
    required uint32 prog_id = 2;
    repeated uint32 child = 3;
}

message HostInfo {
    required uint32 id = 1;
    required string hostname = 2;
    repeated StringPair hostIds = 3;
    required string osDetails = 4;
    repeated InterfaceInfo interfaces = 5;
}

message StringPair {
    required string key = 1;
    required string value = 2;
}

message InterfaceInfo {
    required string name = 1;
    required string macAddress = 2;
    repeated string ipAddresses = 3;
}

message User {
    required int32 userId = 1;  //user id assigned by OS
    required string name = 2;
    repeated string groups = 3;
}

message ProvMessage {
    oneof type {
        DefineProgram define_program = 1;
        DefineAppPpt define_app_ppt = 2;
        DefineSysCall define_sys_call = 3;
        DefineProv define_prov = 4;
        DefineProvSet define_prov_set = 5;
        DefineProvType define_prov_type = 6;
        Event event = 7;
        User user = 9;
        HostInfo host_info = 10;
        DefineUnknownProv unknown_prov = 11;
    }
}
```

3.1.9.2 How tags are defined

The `java.lang.TCDefiner` class tells each of the `TC.report*()` calls whether and how to tag the data for each report call. Primitive arrays are tagged via the `java.lang.RLE`, and also

generate the appropriate run-length encoding protobuf message. There are three definers that ClearScope uses:

1. `TCNullDefiner` - The call to `TCDefiner.define()` simply returns the original tag. This definer is only ever used for debugging, and testing purposes.
2. `TCProvDefiner` - This is backed by a call to `TC.defineProvSet()` on the provided tag. I.e., for each tag called on `TCDefiner.define()`, it will ensure that a `DefineProvSet` message is created for the tag passed to the call (only if the prov set was not already previously defined).
3. `TCProvJoiner` - This is backed by a call to `TC.defineProv()`. In other words, the tag returned by `TCDefiner.define()` is the join of the tag passed to the call, application program point, and sys call ids. A `DefineProv` protobuf message is generated for each call.

3.1.9.3 Sources (with read example)

If there is no tag FD associated with the FD that we are reading from, then we mark the read bytes with the tag associated with the FileDescriptor number itself. This allows us to do things like tag the FileDescriptor objects used for `stdout`, `stderr`, and `stdin` of an execed process, for example. Otherwise, we read four times as many bytes from the tag file, as we did the original FD. The byte array is now tagged with the tags read in from the tag file. We then make the appropriate call to `TC.reportByteArray()` that joins with the application program point, and system call. A new `EventData[]` object of the remaining parameters, and the call to `TC.sink()` is made. This outlines the basic principle behind all reads/sources in ClearScope.

```
   public int read(FileDescriptor fd, byte[] bytes, int byteOffset, int byteOffset_t, int
byteCount, int byteCount_t,
         TCReturn ret) throws ErrnoException, InterruptedIOException {
      // This indirection isn't strictly necessary, but ensures that our public interface
is type safe.
      int bytesRead = readBytes(fd, bytes, byteOffset, byteCount);
      if (!TC.programStarted() || bytesRead <= 0) {
         ret.taint = 0;
         return bytesRead;
      }

      int sysCall = TC.defineSysCall(
         "int libcore.io.Linux.read(java.io.FileDescriptor fd, byte[] bytes, int
byteOffset, int byteCount) [line: 459]");
      int appPpt = ret.getAppPpt();
      int provType = fd.getProvType(false);
      if (provType == -1) {
         // Could only be true if fd was opened in pre-zygote init code.
         ret.taint = 0;
         return bytesRead;
      }

      try {
         readTaint(fd, bytes, byteOffset, bytesRead);
      } catch (ErrnoException err) {
         // Taint FD cannot be null here, since an errno exception cannot be raised in the
event that
         // there is no taint FD.  So no null check is needed in this case.
         TC.logWarning("Unable to read taint from fd(%d) due to errno(%d): %s",
fd.getTaintFd().getInt$(), err.errno,
                     strerror(err.errno));
```

```
      } catch (Throwable misc) {
         FileDescriptor taintFd = fd.getTaintFd();
         TC.logWarning("Unable to read taint from fd(%d) due to %s: %s", taintFd == null ?
-1 : taintFd.getInt$(),
                     misc.getClass().getName(), misc.getMessage());
      }

      // Generate the source event...
      ret.taint = TC.defineProv(appPpt, sysCall, provType, true, 0);
      EventData[] data = {
         TC.reportNonArrayObject("fd", fd, FileDescriptor.class, EventData.PARAM),
         TC.reportByteArray("bytes", bytes, bytesRead, EventData.SRC,
                TCDefiner.joiner(appPpt, sysCall, provType, true)),
         TC.reportInt("byteOffset", byteOffset, byteOffset_t, EventData.PARAM),
         TC.reportInt("byteCount", byteCount, byteCount_t, EventData.PARAM),
         TC.reportInt("r", bytesRead, ret.taint, EventData.SRC)
      };
      TC.source(appPpt, sysCall, provType, data);

      return bytesRead;
   }
```

3.1.9.4 Sinks (with write example)

In a write call, the sink event is generated prior to writing out the tags to the tag file. We construct an `EventData[]` of the parameters to the call, and pass that to the call to `TC.sink()` like we did in the previous read example. After the sink message is generated and reported, we write write the tags to the corresponding tag file. If the `byte[]` had aggregate taint, we must write out this tag (joined with the application program point, and system call id) for each byte written to the original FD. Otherwise, we write out the tags (joined with the application program point, and system call id) corresponding to each byte written to the original FD. This outlines the basic principle behind all writes/sinks in ClearScope.

```
   public int write(FileDescriptor fd, byte[] bytes, int byteOffset, int byteOffset_t, int
byteCount,
         int byteCount_t, TCReturn ret) throws ErrnoException, InterruptedIOException {
      // This indirection isn't strictly necessary, but ensures that our public interface
is type safe.
      int bytesWritten = writeBytes(fd, bytes, byteOffset, byteCount);
      if (!TC.programStarted() || bytesWritten <= 0) {
         ret.taint = 0;
         return bytesWritten;
      }

      int sysCall = TC.defineSysCall(
         "int libcore.io.Linux.write(java.io.FileDescriptor fd, byte[] bytes, int
byteOffset, int byteCount) [line: 751]");
      int appPpt = ret.getAppPpt();
      int provType = fd.getProvType(false);
      if (provType == -1) {
         // Could only be true if fd was opened in pre-zygote init code.
         ret.taint = 0;
         return bytesWritten;
      }

      // Generate the sink event...
      EventData[] data = {
         TC.reportNonArrayObject("fd", fd, FileDescriptor.class, EventData.PARAM),
         TC.reportByteArray("bytes", bytes, bytesWritten, EventData.SINK),
         TC.reportInt("byteOffset", byteOffset, byteOffset_t, EventData.PARAM),
         TC.reportInt("byteCount", byteCount, byteCount_t, EventData.PARAM),
         TC.reportInt("r", bytesWritten, 0, EventData.RET)
```

```
      };
      TC.sink(appPpt, sysCall, provType, data);

      try {
         writeTaint(fd, bytes, byteOffset, bytesWritten, appPpt, sysCall, provType);
      } catch (ErrnoException err) {
         // Taint FD cannot be null here, since an errno exception cannot be raised in the
event that
         // there is no taint FD.  So no null check is needed in this case.
         TC.logWarning("Unable to write taint from fd(%d) due to errno(%d): %s",
fd.getTaintFd().getInt$(), err.errno,
                       strerror(err.errno));
      } catch (Throwable misc) {
         FileDescriptor taintFd = fd.getTaintFd();
         TC.logWarning("Unable to write taint from fd(%d) due to %s: %s", taintFd == null ?
-1 : taintFd.getInt$(),
                       misc.getClass().getName(), misc.getMessage());
      }

      ret.taint = 0;
      return bytesWritten;
   }
```

3.1.10 Build environment

This was a bit of a pain due to the overall complexity of the android build system. Most of this complexity stems from the fact that components of the system are modularized but interdependent, and also some things are built for both the host and device. Additional build steps were required for building the following:

- `dex-instrumentation` - This is built for both the host and the device. This has to build very early for the host, since the **`dx`** tool relies on it to perform the dex translation, as well as perform the instrumentation, on Java bytecode. Additionally, we modified the build rules for the Jack compiler to perform instrumentation on modules built with Jack.
- `provmsgr` - Device only provenance messenger service for serializing prov message data.
- `instd` - Device only instrumentation daemon for providing dynamic instrumentation support.
- `clearscope` - Device and host executables for accessing/analyzing metadata from files.

Fortunately, since system code is built in a directory separate from application code, we can actually pass the right instrumentation flag (-aosp or -app) that will instrument the bytecode as SYSTEM or APPLICATION code.

3.1.10.1 Protocol Buffer Integration

Due to unforeseen build issues relating to how android modules utilize the protocol buffer libraries, and the craziness of the dependencies involved here, the protobuf Java library needed to be integrated into the core java libraries. This is because system calls need to be able to send protocol buffer messages for generating sources, sinks, and other events. This took some clever retrofitting of the android build system for including certain classes in with the System API. This is because Doclava required the javadocs to be formatted in a particular way (provided that the method was not hidden - via "@hide").

3.1.11 Array aggregation / deaggregation

See section 3.1.3.

3.1.12 ART modifications

The only thing super critical for having a working instrumentation required both changes to the garbage collector, and also changes to the mirror classes (which we will talk about in separate subsections). Other less critical changes involved modifying threads to additionally allocate a TCReturn Java object once per thread creation. Also, since class initializers now take an argument (namely, the TCReturn object of the current thread), we modified the class linker to allocate classes by passing the TCReturn reference for that thread to it's initializer. Additionally, to speed up the lookup time needed to access the shadowed version of the reflected Object (Field, Method, or Constructor), the class linker finds these exactly one time at the time the class is loaded.

3.1.12.1 Garbage collector

Because we altered the Java array header to contain a heap reference to a Java `int[]` allocation. We have to compensate for this new allocation during the Garbage Collection mark phase. They have abstracted this concept of visiting a reference tree not only for the purpose of Garbage Collection, but also to handle things like cloning. The ART developers make use of C++ functions, and depending on the use case it passes a different Visitor object to the `Object::VisitReferences()` member function. This is where we must consider this new allocation, for when the Object type is a primitive array.

3.1.12.2 Mirror classes

Since we are modifying the entire system API, we must also modify (by hand) all of the mirror classes in the ART runtime. These classes are core classes (like String, Class, Object, etc...) that are used internally by the runtime. Since they are essentially a "mirror"-image of the Java allocation, the fields in the C++ mirror class must be exactly aligned with the Java class. Fields in Dalvik are ordered with Object fields appearing first, followed by primitive types in descending precision (i.e. 64-bit, then 32-bit, etc...). Because these fields are also packed, it is slightly less trivial to know the exact byte offset of each field. So, generally We will print out the expected field offsets in the first pass, and then correct the order in the second pass.

3.1.12.3 Optimizations

The ART runtime contains many compiler intrinsics for replacing common method calls with inlined assembly that can be generated by the compiler a priori. This replacement procedure is performed on the LIR (low-level IR) produced by the dex2oat compiler. Some examples of calls with intrinsics are String.charAt(), Math.max(), and Thread.currentThread(). In cases where tags do not have to propagate through the call, we essentially mirror the same intrinsic (ensuring that registers line up correctly for the parameters to the instrumented intrinsic). Additionally, we added intrinsics for accessing both aggregate and element-by-element taint on arrays, since it would otherwise turn into an expensive JNI call, and these accesses occur all of the time (including tight loops). Optionally, we can inline the entire array load and store instruction (only in the case where the element is not wide), which is a bit of a smaller performance improvement over the standard array intrinsics. We do this since array accesses actually turn into about 30-40 Dalvik instructions (depending on if it's a load or a store).

3.1.13 Upgrades to new Android versions

3.1.13.1 Describe process and pain points

The issue with migrating to newer versions mostly is due to the fact that it requires complicated 3-way merging between divergent repositories (plural). It's not usually as simple as merging, since typically we need to make a physical copy of the existing method when making method summaries. Therefore, if something inside of a method that we summarized had changed, the "git merge" will be oblivious to the change that would need to happen in the instrumented summary. So, this would introduce a (potentially bad) bug.

We therefore take a very systemic approach here, and manually pull in changes into the Android Open Source Platform (AOSP) iteratively, starting with the most elementary changes required to have a complete instrumentable system. Generally we will start with binder first, since that typically is the hardest of the modified components to work with. Once we have this working on a slightly modified vanilla system that just throws away the tags, we can start to pull in the ART changes required to have a basic complete instrumentation (i.e. see the necessary changes discussed in section 3.1.12). You will also need to address the changes that are divergent from the newer android version. This will typically require understanding new functionality introduced in the more modern version, and hopefully the number of instances is small.

3.1.13.2 Java 8

This required supporting two additional Dalvik instructions (invokedynamic, and invokepolymorphic). Unfortunately this required some amount of work in Soot to be able to support these instructions, since Soot (at that time) did not support these features. Later on there was enough support for this that we were able to go in and fix whatever bugs there were for this, and so now our instrumentation supports Java 8.

3.1.13.3 Strings

What was once implemented as a `char[]` allocation in `java.lang.String` became part of the actual String allocation (where the chars are inlined into the object's allocation). For this reason, Strings cannot be constructed by making the usual constructor call (since the allocation size is now dynamically determined. This is therefore implemented by replacing all constructor calls with calls to a static method inside `java.lang.StringFactory`, as shown in the table below:

`new String()`	`StringFactory.newEmptyString()`
`new String(String)`	`StringFactory.newStringFromString(String)`
`new String(char[])`	`StringFactory.newStringFromChars(char[])`
`new String(char[], int, int)`	`StringFactory.newStringFromChars(char[], int, int)`
`new String(int, int, char[])`	`StringFactory.newStringFromChars(int, int, char[])`
`new String(int[], int, int)`	`StringFactory.newStringFromChars(int[], int, int)`

new String(byte[], int, int, int)	StringFactory.newStringFromBytes(byte[], int, int, int)
new String(byte[], int, int)	StringFactory.newStringFromBytes(byte[], int, int)
new String(byte[], int, int, String)	StringFactory.newStringFromBytes(byte[], int, int, String)
new String(byte[], int, int, Charset)	StringFactory.newStringFromBytes(byte[], int, int, CharsetName)
new String(byte[], int)	StringFactory.newStringFromBytes(byte[], int)
new String(byte[], int, int, int)	StringFactory.newStringFromBytes(byte[], int, int, int)
new String(byte[])	StringFactory.newStringFromBytes(byte[])
new String(byte[], String)	StringFactory.newStringFromBytes(byte[], String)
new String(byte[], Charset)	StringFactory.newStringFromBytes(byte[], Charset)
new String(StringBuffer)	StringFactory.newStringFromStringBuffer(StringBuffer)
new String(StringBuilder)	StringFactory.newStringFromStringBuilder(StringBuilder)

The only modifications required to support this was to simply find the place in the compiler where this translation happened, then add translations for each of their respective instrumented constructor calls. Then our DEX instrumentation will get us most of the way there. The only thing left at that point would be to hand instrument all of the native calls by utilizing the JNI taint access routines that we added for primitive arrays.

3.1.14 String (and Primitive Wrapper) interning

We support string interning by replacing the `==` operator with a call to `TC.acmp(Object, Object)` when either operand can be either a String or one of the (non-float) primitive wrapper class types: `Boolean`, `Byte`, `Character`, `Short`, `Integer`, and `Long`. This simply calls `Object.transparentEqualEquals()` on the two operands, which is a method we added to `java.lang.Object` for comparing the interned/cached references. This method is overridden in each of the aforementioned classes of the types that will support interning/caching. For example, for supporting `String.intern()`, consider the following changes to `java.lang.String`:

```
public final class String
   implements java.io.Serializable, Comparable<String>, CharSequence {

   ...
```

```
    /** @hide */
    @TCTransparent
    protected transient String tc_interned;

    ...

    /** @hide */
    @Override
    @TCDontShadow
    @TCTransparent
    public boolean transparentEqualEquals(Object o) {
        if (!(o instanceof String))
            return false;
        String otherString = (String) o;
        Object val1 = this.tc_interned;
        if (val1 == null)
            val1 = this;
        Object val2 = otherString.tc_interned;
        if (val2 == null)
            val2 = otherString;
        return val1 == val2;
    }

    @FastNative
    @TCDontShadow
    public native String intern();

    /** @hide */
    @TCDontShadow
    @TCTransparent
    public String intern(TCReturn ret) {
        // Create copy of String and set the interned string
        // to be a field in the copied String.  The String we
        // return here will be the one we copied.  Any comparisons
        // will have to compare interned references.
        String si = intern();
        String newString = StringFactory.newStringFromString(this);
        newString.tc_interned = si;
        return newString;
    }

    ...

}
```

We perform a similar technique for supporting comparisons on Objects returned from calls to the boxing routines that would otherwise return a cached value. For example, consider the following changes to `java.lang.Byte` (which essentially caches byte value with a unique wrapped Byte object):

```
public final class Byte extends Number implements Comparable<Byte> {

    ...

    @TCDontShadow
    private final byte value;

    /** @hide */
    @TCDontShadow
    @TCTransparent
    public int value$t;

    /** @hide */
    @TCDontShadow
```

```
    @TCTransparent
    public transient boolean cached;

    ...

    private static class ByteCache {
        private ByteCache(){}

        static final Byte cache[] = new Byte[-(-128) + 127 + 1];

        static {
            for(int i = 0; i < cache.length; i++)
                cache[i] = new Byte((byte)(i - 128), true);
        }
    }

    @TCDontShadow
    public Byte(byte value, boolean cached) {
        this.value = value;
        this.cached = cached;
    }

    /** @hide */
    @TCDontShadow
    @TCTransparent
    public Byte(byte value, int value_t, boolean cached, int cached_t, TCReturn ret) {
        this.value = value;
        this.value$t = value_t;
        this.cached = cached;
    }

    @TCDontShadow
    public static Byte valueOf(byte b) {
        final int offset = 128;
        return ByteCache.cache[(int)b + offset];
    }

    /** @hide **/
    @TCDontShadow
    @TCTransparent
    public static Byte valueOf(byte b, int b_t, TCReturn ret) {
        // Return new instance instead of caching!
        return new Byte(b, b_t, true, 0, ret);
    }

    /** @hide */
    @TCDontShadow
    @TCTransparent
    public boolean transparentEqualEquals(Object object) {
        if (!(object instanceof Byte))
            return false;
        Byte other = (Byte) object;
        if (!cached || !other.cached)
            return this == object;
        return other.value == value;
    }

    ...

}
```

3.1.15 SELinux

Since we are adding system services to the device, it was critical to modify the `init.rc` file such that the system is initialized properly to allow for tagging inside system processes/services.

1. `provmsgr` - This service needs to start as early as possible, since we are also tracking native processes via LAM tracing. It's marked a critical service, since it's required for the device to function properly.
2. `instd` - The instrumentation daemon is also a critical services that must start under root user, because it needs to be able to fork/exec under essentially any trusted/untrusted app.

In addition to this, both the `provmsgr` and `instd` service need directories created under `/data` for storing provenance data (when writing prov data to a file), and also directories for storing the dynamically instrumented DEX files (in the case of `instd`). It also required setting up the tag file system, since the tag files are actually written to a separate file system. We also needed to set up shared memory, since LAM tracing requires this.

SELinux policies needed to be created for handling these additional system services. Since the base policies were a bit too restrictive, we also needed to allow for extra sets of permissions for these services. Also, since all trusted/untrusted apps needed to (for example) communicate over binder, we had to allow access through the service manager.

1. `provmsgr` - Allow for communicating over binder, creating FIFO file for writing prov messages, as well as working with UDP sockets (for accessing network information).
2. `instd` - Allow for communicating over binder, accessing Dalvik cache, execing the shell and zygote (for invoking our Java soot-based instrumentation), setting the UID/GID (needed to ensure correct permissions of the instrumented Dalvik files), and writing files to the system directory (for storing Dalvik).
3. `lam_tracer` - Allow access to shared memory, access to procfs, rootfs, tmpfs, fifos, and datagram sockets.

We also added properties for both debugging purposes, and also for the purpose of doing things like turning off dynamic instrumentation, for example. These additional properties needed to be added so that the system server could access these properties.

3.1.16 Compatibility Test Suite (CTS) modifications and results

Most all of the modifications were simply to increase the timeout needed to run certain tests, since in certain cases the instrumented tests exceeded this timeout bound. We wrote scripts to triage the failures so that we could easily compare runs between the vanilla system, system instrumented with soot-identity, as well as the fully-instrumented system. This allowed us to go in and more easily fix any soot related bug that was breaking certain fundamental (but more esoteric) parts of the system that was apparently hard for Soot to get right (i.e. system features that rely on certain system annotations). In fact, in the early phases of the project, the phone wasn't even robust enough to fully boot without crashing somewhere in the system API on critical Java services. For example, there were many floating-point bugs that prevented the phone from displaying graphics properly. So, we actually fixed a fair number of soot bugs (15-20, or so).

3.1.17 Binary Tracking and Reporting Implementation

The binary tracking and reporting implementation is based on kernel modifications that enable low-overhead system call monitoring and reporting callbacks. We call the system lean-and-mean (LAM) tracking and reporting. The main goal of LAM was to achieve a very low overhead, and compete with existing kernel-side system call tracing systems. With a previous system based on

ptrace, the cost of context-switching was too high; Firefox (which creates at least 100 threads) did not scale well. Like many others have before, we discovered the performance drawbacks of ptrace. Thus LAM was designed with a "same-thread" model in mind.

Existing Linux interfaces such as seccomp allow the user to provide a BPF program that can be used to efficiently trace system calls, but BPF's cannot have unbounded loops, something we needed for the maintaining provenance metadata. So we opted for something more general: signal delivery.

Every time a system call is made, we interrupt the current thread before and after the system call has returned. The remainder of this section discuss the kernel mechanism design and modifications and the user-level design of LAM.

3.1.17.1 Kernel Modifications

3.1.17.1.1 Controlling tracing of system calls

In `arch/arm64/kernel/entry.S`, the kernel-side entry point for system calls, there is a slow path and a fast path. While the fast path transfers control straight to the relevant system call procedure, the slow path takes a detour into `syscall_trace_enter` located in `arch/arm64/kernel/ptrace.c` before, and then into `syscall_trace_exit` after (also located in `arch/arm64/kernel/ptrace.c`). Which one taken is dictated by whether the current thread flags intersect with `_TIF_SYSCALL_WORK`.

By defining our own thread flags,

```
#define TIF_CSBLAM            12
#define TIF_CSBLAM_ENTERED     13
```

And changing `_TIF_SYSCALL_WORK` to include `_TIF_CSBLAM`,

```
#define _TIF_CSBLAM        (1 << TIF_CSBLAM)
#define _TIF_SYSCALL_WORK   (_TIF_SYSCALL_TRACE | _TIF_SYSCALL_AUDIT | \
                        _TIF_SYSCALL_TRACEPOINT | _TIF_SECCOMP | \
                        _TIF_NOHZ | _TIF_CSBLAM)
```

Then the slow-path will be taken if the `TIF_CSBLAM` thread flag is set, giving us the opportunity to initiate userspace tracing. Note that if the `TIF_CSBLAM` thread flag is not set, there will be negligible overhead for system calls since the fast-path will be taken which bypasses all the same-thread-tracing code.

3.1.17.1.2 Tracing mechanism

If the `TIF_CSBLAM` thread flag is set, and the system-call number is marked for reporting, it shall be traced on the same thread.

This is essentially equivalent to immediately delivering a signal on the CSBLAM stack for the current thread (with the handler being the post-syscall procedure). We harness the existing machinery in the kernel to perform this. The `TIF_CSBLAM` thread flag is cleared before returning to userspace; if we make system calls in the syscall procedure, we do not want them to be traced.

The return address of the syscall procedure points into the vDSO(7), where the `csblam_sigreturn(2)` system-call is invoked. Its semantics are nearly equivalent to `sigreturn(2)`, except the `TIF_CSBLAM` thread flag is set before returning to userspace – thus we continue tracing system calls following execution of the syscall procedure.

3.1.17.1.3 Kernel interfaces

Here we define the kernel interfaces we have added.

```
long sys_csblam_setup(void __user *sys_entr_proc,
                  void __user *sys_exit_proc,
                  void __user *sig_proc)
```

Caller provides the addresses of three functions: (1) a pre-syscall procedure, (2) a post-syscall procedure, and (3) signal-handler-wrapper. Provided that `csblam_set_stack(2)` has been called, this system call effectively starts tracing for the current thread.

Return value: On success, returns `0`. Furthermore the follow-on-clone and follow-on-exec task struct fields are set to `1`. On error, returns `-EAGAIN` (if this system call was previously invoked).

```
long sys_csblam_set_stack(void __user *stack, size_t size)
```

This is for userspace to define the stack on which the tracing procedures execute, akin to `sigaltstack(2)`.

Return value: On success, returns `0`. On error, returns `-EAGAIN` (if this system call was previously invoked).

```
long sys_csblam_set_thd_flg(void)
```

Sets the `TIF_CSBLAM` flag for the current thread.

Return value: On success, returns `0`. On error, returns `-EPERM` (if this system call was not called by either `libc.so`, `libcsblam.so`, or `provmsgr`).

```
long sys_csblam_clr_thd_flg(void)
```

Clears the `TIF_CSBLAM` flag for the current thread.

Return value: On success, returns `0`. On error, returns `-EPERM` (if this system call was not called by either `libc.so`, `libcsblam.so`, or `provmsgr`).

```
long sys_csblam_get_thd_flg(void)
```

Return value: Returns whether the `TIF_CSBLAM` thread flag is set in the current thread.

```
long sys_csblam_tst_follow_exec(void)
```

Return value: Returns the "follow-on-exec" field in the current thread.

```
long sys_csblam_set_next_tag(unsigned long)
```

Sets the "next tag", i.e. the next number to be returned from `sys_csblam_request_tags()`

Return value: On success, returns `0`. On error, returns `-EPERM` (if this system call was not called by either `libc.so`, `libcsblam.so`, or `provmsgr`).

```
long sys_csblam_request_tags(void)
```

Return value: On success, returns tag. On error, returns `-EAGAIN` (if `sys_csblam_set_next_tag` hasn't been called yet).

```
long sys_csblam_get_next_tag(void)
```

Return value: The "next tag" (i.e. the next value to be returned by `sys_csblam_request_tags`).

```
long sys_csblam_finalize(void)
```

Disables tracing for all threads in the current thread group. Intended to be called before the application cleanly shuts down.

Return value: On success, returns `0`. On error, returns `-EPERM` (if this system call was not called by either `libc.so`, `libcsblam.so`, or `provmsgr`).

The "handler" for this so-called interrupt is designated by the first successful call to `sys_csblam_setup(2)`.

3.1.17.2 User-Level Design and Implementation

In this section, we describe the user-level modifications we have made to the system for binary LAM tracking and reporting.

3.1.17.2.1 Runtime linker (`/system/bin/linker`)

Tracing is set in motion in the dynamic linker if:

1. The follow-on-exec flag is set, or
2. `readlink("/proc/self/exe")` is *not* on the whitelist in `bionic/linker/csblam_whitelist.hpp`.

Doing so can be understood as leveraging the `LD_PRELOAD` environment variable to add `libcsblam.so` to the list of DSO's loaded into the dynamically-linked process.

3.1.17.2.2 Same-thread-tracing library (`/system/lib/libcsblam.so`)

Constructor

The constructor, a `DT_INIT` function which is called by the dynamic linker after loading all dependent shared objects, immediately establishes a connection with `provmsgr`. Then it passes

the addresses of the pre-syscall procedure, post-syscall procedure, and signal handler wrapper to the `sys_csblam_setup` system call, *unless* `"CSBLAM_HAS_PROGRAM_START"` is present in the environment (e.g. for an android app), in which case this setup is postponed until later.

Syscall Procedures

In the post syscall procedure, provenance is propagated for file `read(2)`'s and `write(2)`'s wherever possible. A map from file descriptor numbers to provenance file descriptor numbers is maintained, where regular files have provenance counterparts (whose size is four times that of their counterpart), and pipes have provenance counterparts (whose size is also set to four times of their counterpart via `fcntl(2)` with `F_GETPIPE_SZ` / `F_SETPIPE_SZ`).

Generally speaking, if `open(2)` or `pipe(2)` is traced, provenance map entries will be created accordingly.

Dynamic memory allocation

Care is taken regarding calling functions from libc. This is because libc may make a system call in the middle of a non-reentrant function (e.g. `malloc`). So in order to dynamically allocate memory, we call into jemalloc that has been built inside of `libcsblam.so` (for internal use only). And to make use of that with standard containers, we define our own C++ allocator (which is passed as a template parameter to `std::unordered_map`, `std::list`, etc...).

3.1.17.2.3 Provenance union

The provenance union provides an over-approximation of the sinks and sources of interest. It is what it sounds like- an ongoing join of tags.

```
static thread_local jtag _initial_provenance_union = NULL_TAG;
thread_local jtag *provenance_union = &_initial_provenance_union;
```

Note that it is a pointer. Along the way in a Java thread's execution, after a native function is called a "new" provenance union (initialized to `NULL_TAG`) on the stack space is set to the provenance union pointer, before saving the old address (to be restored after the native code is returned). And after a native call, one can call back into Java, and after that back into native- creating a second provenance union. Thus there is really a stack of provenance unions in a thread's execution. Under two more circumstances will "new" provenance unions be created: at the execution of a signal handler, and the execution of a `DT_INIT` "constructor" function.

3.1.17.2.4 Lang transitions

As a hybrid system, we disable userspace same-thread tracing when calling into Java, since we know the instrumented Java will do the reporting on its own. We do this via `csblam_clr_thd_flg(2)`. We save `tls::provenance_union`, before it becomes `nullptr`. After Java returns, we set `tls::provenance_union` back to its old value, and reenable userspace same-thread tracing via `csblam_set_thd_flg(2)`.

If a `DT_INIT` function is called by the dynamic linker or a signal handler is called, we save the current `tls::provenance_union` and create (on the stack) a new provenance union to set

`tls::provenance_union` to point at. After it returns, we restore `tls::provenance_union` to point to the original provenance union.

3.1.17.2.5 Reporting

Establishing a connection to "/dev/socket/provmsg" is necessary to perform any reporting. The `O_NONBLOCK` file status flag on the resulting file descriptor is set. Furthermore the value of `SO_SNDBUF` is set to `/proc/sys/net/core/wmem_max`, which we have configured to have a value of 128MiB.

The reporting format expected by `provmsgr` is a struct, whose first byte always indicates what type of struct message it is. Following is an example of `frameworks/native/include/provmsgr/msg/event1b.hpp`, which defines a provmsg:

```
BEG_PROVMSG(Event1WithBytes)

PROVMSG_FIELD(uint32_t, program_id)
PROVMSG_FIELD(uint32_t, flow)
PROVMSG_FIELD(uint64_t, time)
PROVMSG_FIELD(uint32_t, tid)
PROVMSG_FIELD(uint32_t, sys_call)
PROVMSG_FIELD(uint32_t, app_ppt)

PROVMSG_STR_FIELD(sym0)
PROVMSG_FIELD(int64_t, arg0)
PROVMSG_STR_FIELD(str0)

PROVMSG_FIELD(int64_t, r)
PROVMSG_FIELD(uint32_t, pred1)
PROVMSG_FIELD(uint32_t, pred2)
PROVMSG_FIELD(uint32_t, provenance_union_entr)
PROVMSG_FIELD(uint32_t, provenance_union_exit)

PROVMSG_STR_FIELD(bytes)
PROVMSG_VLA_FIELD(uint32_t, bytes_tags)

END_PROVMSG(Event1WithBytes)
```

This "Event1WithBytes" message is used for reporting system calls that take 1 argument, and has a variable-length stream of `bytes` as well. Pre-processor tricks are used to generate definitions for sending and receiving each of these messages with zero-intermediate copies for the variable-length data.

3.1.17.2.6 Provenance Messenger (`/system/bin/provmsgr`)

All reporting goes through `provmsgr` before being written to the "final output" file descriptor (e.g. /data/progmsgr/prov-output), whose throughput may obviously vary. Thus `provmsgr` acts as a high-speed buffer and authenticator.

Definition of daemon in `system/core/rootdir/init.rc`

```
service provmsgr /system/bin/provmsgr
   class main
   priority -20
   critical
```

```
   user prov_msgr
   group prov_msgr inet readproc
   socket provmsg seqpacket 666 prov_msgr prov_msgr
   socket provctl stream 666 prov_msgr prov_msgr
   writepid /dev/cpuset/foreground/tasks
```

`provmsgr` listens on `/dev/socket/provmsg`, a Unix domain socket of the `SOCK_SEQPACKET` variety. For each connection established, a thread is created to service that connection. The `SO_RCVBUF` size is set to `/proc/sys/net/core/rmem_max`, which we have configured to 128MiB.

Here follows the main message receive loop:

```
  constexpr unsigned MaxMsgHdrSize = std::max<size_t>({0UL
#define BEG_PROVMSG(x) , sizeof(struct x##MsgHdr)
#include <provmsgr/msg/all_msgs.hpp>
  });

  uint8_t msghdrbuff[MaxMsgHdrSize];
  for (;;) {
    ssize_t ret =
        recv(args->data_socket, &msghdrbuff[0], MaxMsgHdrSize, MSG_PEEK);

    if (unlikely(ret <= 0)) {
      if (ret < 0)
        print("recv failed (%s)", strerror(errno));
      return nullptr;
    }

    if (unlikely(!msgHdrSizes[msghdrbuff[0]])) {
      print("bad message header type %u", static_cast<unsigned>(msghdrbuff[0]));
      return nullptr;
    }

    if (unlikely(ret < msgHdrSizes[msghdrbuff[0]])) {
      print("failed to peek %s message (got %zd < %u)", msgNames[msghdrbuff[0]], ret,
msgHdrSizes[msghdrbuff[0]]);
      return nullptr;
    }

    if (unlikely(!receiveMsgProcs[msghdrbuff[0]](&msghdrbuff[0],
                                         args->data_socket,
                                         args->fd))) {
      print("failed to receive %s message", msgNames[msghdrbuff[0]]);
      return nullptr;
    }
  }
```

Each message type is indexed into an array of function pointers, `receiveMsgProcs`, which perform the final `recvmsg(2)` and then (optionally convert the provmsg to a ProtoBuf) writing the final output to disk (or usb).

3.1.17.3 System Calls

The folowing is the list of system calls that are reported.

accept
accept4
access

acct
add\s\do4(k)ey
adjtimex
bdflush
bind
bpf
brk
capget
capset
chdir
chmod
chown
chown16
chroot
clock\s\do4(a)djtime
clock\s\do4(g)etres
clock\s\do4(g)ettime
clock\s\do4(n)anosleep
clock\s\do4(s)ettime
clone
close
compat\s\do4(f)anotify\s\do4(m)ark
connect
creat
delete\s\do4(m)odule
dup
dup2
dup3
epoll\s\do4(c)reate
epoll\s\do4(c)reate1
epoll\s\do4(c)tl
epoll\s\do4(p)wait
epoll\s\do4(w)ait
eventfd
eventfd2
execve
execveat
exit
exit\s\do4(g)roup
faccessat
fadvise64
fadvise64\s\do4(6)4
fallocate
fanotify\s\do4(i)nit
fanotify\s\do4(m)ark
fchdir
fchmod
fchmodat
fchown
fchown16
fchownat
fcntl
fcntl64
fdatasync
fgetxattr

finit\s\do4(m)odule
flistxattr
flock
fork
fremovexattr
fsetxattr
fstat
fstat64
fstatat
fstatat64
fstatfs
fstatfs64
fsync
ftruncate
ftruncate64
futex
futimesat
getcpu
getcwd
getdents
getdents64
getegid
getegid16
geteuid
geteuid16
getgid
getgid16
getgroups
getgroups16
getitimer
get\s\do4(m)empolicy
getpeername
getpgid
getpgrp
getpid
getppid
getpriority
getrandom
getresgid
getresgid16
getresuid
getresuid16
getrlimit
get\s\do4(r)obust\s\do4(l)ist
getrusage
getsid
getsockname
getsockopt
gettid
gettimeofday
getuid
getuid16
getxattr
init\s\do4(m)odule
inotify\s\do4(a)dd\s\do4(w)atch

inotify\s\do4(i)nit
inotify\s\do4(i)nit1
inotify\s\do4(r)m\s\do4(w)atch
io\s\do4(c)ancel
ioctl
io\s\do4(d)estroy
io\s\do4(g)etevents
ioprio\s\do4(g)et
ioprio\s\do4(s)et
io\s\do4(s)etup
io\s\do4(s)ubmit
kcmp
kexec\s\do4(l)oad
keyctl
kill
lchown
lchown16
lgetxattr
link
linkat
listen
listxattr
llistxattr
llseek
lookup\s\do4(d)cookie
lremovexattr
lseek
lsetxattr
lstat
lstat64
madvise
mbind
membarrier
memfd\s\do4(c)reate
migrate\s\do4(p)ages
mincore
mkdir
mkdirat
mknod
mknodat
mlock
mlock2
mlockall
mmap
mmap2
mount
move\s\do4(p)ages
mprotect
mq_getsetattr
mq_notify
mq_open
mq_timedreceive
mq_timedsend
mq_unlink
mremap

msgctl
msgget
msgrcv
msgsnd
msync
munlock
munlockall
munmap
name_to_handle_at
nanosleep
newuname
nice
open
openat
open_by_handle_at
pause
pciconfig_iobase
pciconfig_read
pciconfig_write
perf_event_open
personality
pipe
pipe2
pivot_root
poll
ppoll
prctl
pread64
preadv
prlimit64
process_vm_readv
process_vm_writev
pselect6
ptrace
pwrite64
pwritev
quotactl
read
readahead
readlink
readlinkat
readv
reboot
recv
recvfrom
recvmmsg
recvmsg
remap_file_pages
removexattr
rename
renameat
renameat2
request_key
restart_syscall
rmdir

rt_sigaction
rt_sigpending
rt_sigprocmask
rt_sigqueueinfo
rt_sigreturn
rt_sigsuspend
rt_sigtimedwait
rt_tgsigqueueinfo
sched_getaffinity
sched_getattr
sched_getparam
sched_get_priority_max
sched_get_priority_min
sched_getscheduler
sched_rr_get_interval
sched_setaffinity
sched_setattr
sched_setparam
sched_setscheduler
sched_yield
seccomp
select
semctl
semget
semop
semtimedop
send
sendfile
sendfile64
sendmmsg
sendmsg
sendto
setdomainname
setfsgid
setfsgid16
setfsuid
setfsuid16
setgid
setgid16
setgroups
setgroups16
sethostname
setitimer
set_mempolicy
setns
setpgid
setpriority
setregid
setregid16
setresgid
setresgid16
setresuid
setresuid16
setreuid
setreuid16

setrlimit
set_robust_list
setsid
setsockopt
set_tid_address
settimeofday
setuid
setuid16
setxattr
shmat
shmctl
shmdt
shmget
shutdown
sigaction
sigaltstack
signalfd
signalfd4
sigpending
sigprocmask
sigreturn
sigsuspend
socket
socketpair
splice
stat
stat64
statfs
statfs64
swapoff
swapon
symlink
symlinkat
sync
sync_file_range
sync_file_range2
syncfs
sysctl
sysfs
sysinfo
syslog
Tee
tgkill
timer_create
timer_delete
timerfd_create
timerfd_gettime
timerfd_settime
timer_getoverrun
timer_gettime
timer_settime
times
tkill
truncate
truncate64

umask
umount
unlink
unlinkat
unshare
uselib
userfaultfd
ustat
utimensat
utimes
vfork
vhangup
vmsplice
wait4
waitid
write
writev

Table 3.1: List of system calls tracked and reported by the native tracking component of ClearScope.

3.1.17.4 Whitelist

The following system binaries are not traced by our system currently. We choose to disable provenance tracking and reporting for these binaries because they were out-of-scope for the Transparent Computing (TC) engagements and/or tracing would significantly increase overhead.

/data/local/tmp/lldb-server
/init
/su
/system/bin/adbd
kk/system/bin/app_process32
/system/bin/app_process64
/system/bin/audioserver
/system/bin/bootanimation
/system/bin/bootstat
/system/bin/cameraserver
/system/bin/cmd
/system/bin/crash_dump32
/system/bin/crash_dump64
/system/bin/dex2oat
/system/bin/drmserver
/system/bin/e2fsck
/system/bin/folio_daemon
/system/bin/gatekeeperd
/system/bin/healthd
/system/bin/htop
/system/bin/hw/android.hidl.allocator@1.0-service
/system/bin/hwservicemanager
/system/bin/idmap
/system/bin/installd
/system/bin/instd

/system/bin/ip6tables
/system/bin/iptables
/system/bin/keystore
/system/bin/lmkd
/system/bin/logcat
/system/bin/logd
/system/bin/mediadrmserver
/system/bin/mediaextractor
/system/bin/mediametrics
/system/bin/mediaserver
/system/bin/netd
/system/bin/oatdump
/system/bin/patchoat
/system/bin/preopt2cachename
/system/bin/profman
/system/bin/provcat
/system/bin/provmsgr
/system/bin/pstree
/system/bin/recovery-persist
/system/bin/recovery-refresh
/system/bin/sdcard
/system/bin/secilc
/system/bin/sensorservice
/system/bin/servicemanager
/system/bin/sh
/system/bin/storaged
/system/bin/surfaceflinger
/system/bin/thermalserviced
/system/bin/tombstoned
/system/bin/toybox
/system/bin/tune2fs
/system/bin/tzdatacheck
/system/bin/update_engine
/system/bin/update_verifier
/system/bin/vdc
/system/bin/vold
/system/bin/webview_zygote32
/system/bin/wificond
/system/xbin/su
/vendor/bin/ATFWD-daemon
/vendor/bin/KmInstallKeybox
/vendor/bin/PktRspTest
/vendor/bin/StoreKeybox
/vendor/bin/WifiLogger_app
/vendor/bin/adsprpcd
/vendor/bin/athdiag
/vendor/bin/btnvtool
/vendor/bin/chre
/vendor/bin/cnd
/vendor/bin/cnss-daemon
/vendor/bin/cnss_diag
/vendor/bin/cplay
/vendor/bin/diag_callback_sample
/vendor/bin/diag_dci_sample
/vendor/bin/diag_klog

/vendor/bin/diag_mdlog
/vendor/bin/diag_socket_log
/vendor/bin/diag_uart_log
/vendor/bin/easel_boot_test
/vendor/bin/ese-ls-provision
/vendor/bin/ese-replay
/vendor/bin/ese_load
/vendor/bin/esed
/vendor/bin/ezlsh
/vendor/bin/ezlspi
/vendor/bin/grep
/vendor/bin/halutil
/vendor/bin/hdrplus_client_tests
/vendor/bin/hostapd
/vendor/bin/hostapd_cli
/vendor/bin/hw/android.hardware.audio@2.0-service
/vendor/bin/hw/android.hardware.biometrics.fingerprint@2.1-service.wahoo
/vendor/bin/hw/android.hardware.bluetooth@1.0-service
/vendor/bin/hw/android.hardware.boot@1.0-service
/vendor/bin/hw/android.hardware.camera.provider@2.4-service
/vendor/bin/hw/android.hardware.cas@1.0-service
/vendor/bin/hw/android.hardware.configstore@1.0-service
/vendor/bin/hw/android.hardware.contexthub@1.0-service
/vendor/bin/hw/android.hardware.drm@1.0-service
/vendor/bin/hw/android.hardware.drm@1.0-service.widevine
/vendor/bin/hw/android.hardware.dumpstate@1.0-service.wahoo
/vendor/bin/hw/android.hardware.gatekeeper@1.0-service-qti
/vendor/bin/hw/android.hardware.gnss@1.0-service-qti
/vendor/bin/hw/android.hardware.graphics.allocator@2.0-service
/vendor/bin/hw/android.hardware.graphics.composer@2.1-service
/vendor/bin/hw/android.hardware.keymaster@3.0-service-qti
/vendor/bin/hw/android.hardware.light@2.0-service
/vendor/bin/hw/android.hardware.media.omx@1.0-service
/vendor/bin/hw/android.hardware.memtrack@1.0-service
/vendor/bin/hw/android.hardware.nfc@1.0-service
/vendor/bin/hw/android.hardware.oemlock@1.0-service
/vendor/bin/hw/android.hardware.power@1.1-service.wahoo
/vendor/bin/hw/android.hardware.sensors@1.0-service
/vendor/bin/hw/android.hardware.usb@1.1-service.wahoo
/vendor/bin/hw/android.hardware.vibrator@1.1-service.wahoo
/vendor/bin/hw/android.hardware.vr@1.0-service.wahoo
/vendor/bin/hw/android.hardware.wifi.offload@1.0-service
/vendor/bin/hw/android.hardware.wifi@1.0-service
/vendor/bin/hw/rild
/vendor/bin/hw/wpa_supplicant
/vendor/bin/ims_rtp_daemon
/vendor/bin/imsdatadaemon
/vendor/bin/imsqmidaemon
/vendor/bin/imsrcsd
/vendor/bin/init.insmod.sh
/vendor/bin/init.power.sh
/vendor/bin/init.qcom.devstart.sh
/vendor/bin/init.qcom.ipastart.sh
/vendor/bin/init.radio.sh
/vendor/bin/ipacm

/vendor/bin/irsc_util
/vendor/bin/loc_launcher
/vendor/bin/lowi-server
/vendor/bin/msm_irqbalance
/vendor/bin/netmgrd
/vendor/bin/nl_listener
/vendor/bin/oemlock-bridge
/vendor/bin/oemlock_provision
/vendor/bin/pbserver
/vendor/bin/pbticlient
/vendor/bin/pbtiserver
/vendor/bin/pd-mapper
/vendor/bin/perfd
/vendor/bin/pktlogconf
/vendor/bin/pm-proxy
/vendor/bin/pm-service
/vendor/bin/port-bridge
/vendor/bin/qseecom_sample_client
/vendor/bin/qseecomd
/vendor/bin/qseeproxydaemon
/vendor/bin/qseeproxysampledaemon
/vendor/bin/qti
/vendor/bin/radish
/vendor/bin/rmt_storage
/vendor/bin/sensors.qcom
/vendor/bin/sensors_test
/vendor/bin/sh
/vendor/bin/smlog_dump
/vendor/bin/sns_cm_test
/vendor/bin/sns_daf_test
/vendor/bin/spectraltool
/vendor/bin/ssr_diag
/vendor/bin/ssr_setup
/vendor/bin/subsystem_ramdump
/vendor/bin/test_diag
/vendor/bin/tftp_server
/vendor/bin/thermal-engine
/vendor/bin/time_daemon
/vendor/bin/toybox_vendor
/vendor/bin/vndservice
/vendor/bin/vndservicemanager
/vendor/bin/wcnss_filter
/vendor/bin/wpa_cli
/vendor/bin/xtra-daemon

Table 3.2: List of system binaries that are not tracked by ClearScope.

3.1.17.5 Protections

At the top of most of the syscall definitions, there is this little prologue:

```
asmlinkage long sys_csblam_(struct pt_regs *regs)
{
       if (!is_trusted_code(regs->pc) ||
```

```
          !is_trusted_code(regs->regs[30]))
             return -EPERM;
```

`is_trusted_code` checks to make sure that the program counter and return address registers point into either `libc.so`, `libcsblam.so`, or `provmsgr`.

```
static bool is_trusted_code(unsigned long pc) {
       bool res = false;
       struct mm_struct *mm = current->mm;
       struct vm_area_struct *vma = NULL;
       char *p;
       char *tmp = NULL;
       struct file *f = NULL;

       down_read(&mm->mmap_sem);

       vma = find_vma(mm, pc);

       if (!vma || !vma->vm_file)
              goto out;

       tmp = (char *)__get_free_page(GFP_TEMPORARY);
       if (!tmp)
              goto out;

       f = get_file(vma->vm_file);

       p = file_path(f, tmp, PAGE_SIZE);
       if (IS_ERR(p)) {
              //pr_info("(CSBLAM) is_trusted_code: failed to get file path\n");
       } else {
              __kernel_size_t n = strlen(p);

              const char s1[] = { 'l', 'i', 'b', 'c', '.', 's', 'o' };
              const char s2[] = { 'l', 'i', 'b', 'c', 's', 'b', 'l', 'a', 'm', '.', 's', 'o'
};
              const char s3[] = { 'p', 'r', 'o', 'v', 'm', 's', 'g', 'r' };

#define _CSBLAM_CHECK_ENDSWITH(suffix)                \
  do {                                                \
    unsigned i;                                       \
                                                      \
    if (res)                                          \
      break;                                          \
                                                      \
    if (n <= sizeof(suffix))                          \
      break;                                          \
                                                      \
    res = true;                                       \
    for (i = 0; i < sizeof(suffix); ++i) {            \
      if (p[n - sizeof(suffix) + i] != suffix[i]) {   \
        res = false;                                  \
        break;                                        \
      }                                               \
    }                                                 \
  } while (0)

              _CSBLAM_CHECK_ENDSWITH(s1);
              _CSBLAM_CHECK_ENDSWITH(s2);
              _CSBLAM_CHECK_ENDSWITH(s3);

#undef _CSBLAM_CHECK_ENDSWITH
       }
```

```
	fput(f);
	free_page((unsigned long)tmp);
out:
	up_read(&mm->mmap_sem);
	return res;
}
```

As a defensive maneuver we also modify `mprotect(2)` in the following way:

```
/* mm/mprotect.c */

enum userspace_code_classification_t {
	USERSPACE_CODE_UNKNOWN = 0,
	USERSPACE_CODE_LINKER = 1,
	USERSPACE_CODE_LIBC = 2,
	USERSPACE_CODE_LIBCSBLAM = 3
};

SYSCALL_DEFINE3(mprotect, unsigned long, start, size_t, len,
		unsigned long, prot)
{
	unsigned long vm_flags, nstart, end, tmp, reqprot;
	struct vm_area_struct *vma, *prev;
	int error = -EINVAL;
	const int grows = prot & (PROT_GROWSDOWN|PROT_GROWSUP);
	prot &= ~(PROT_GROWSDOWN|PROT_GROWSUP);
	if (grows == (PROT_GROWSDOWN|PROT_GROWSUP)) /* can't be both */
		return -EINVAL;

	if (start & ~PAGE_MASK)
		return -EINVAL;
	if (!len)
		return 0;
	len = PAGE_ALIGN(len);
	end = start + len;
	if (end <= start)
		return -ENOMEM;
	if (!arch_validate_prot(prot))
		return -EINVAL;

	{
		enum userspace_code_classification_t start_class =
			classify_userspace_code(start);
		if (start_class == USERSPACE_CODE_LIBC ||
		    start_class == USERSPACE_CODE_LIBCSBLAM ||
		    start_class == USERSPACE_CODE_LINKER) {
			struct pt_regs *regs = current_pt_regs();
			enum userspace_code_classification_t pc_class =
				classify_userspace_code(regs->pc);

			/* only the linker shall be allowed to do this */
			if (pc_class != USERSPACE_CODE_LINKER) {
				pr_info("mprotect(libc|libcsblam|linker) by <%s>\n",
					userspace_code_classification_desc_tbl
						[pc_class]);
				return -EPERM;
			}
		}
	}

   /* ... */
```

If start is either `libc.so`, `libcsblam.so`, or linker, the system call fails with `-EPERM` unless the program counter of the caller resides in the dynamic linker (i.e. `/system/bin/linker`).

The rest of the code above is straightforward:

```
/* linux/mm/mprotect.c */

enum userspace_code_classification_t classify_userspace_code(unsigned long pc)
{
	enum userspace_code_classification_t res = USERSPACE_CODE_UNKNOWN;
	struct mm_struct *mm = current->mm;
	struct vm_area_struct *vma = NULL;
	char *p;
	char *tmp = NULL;
	struct file *f = NULL;

	down_read(&mm->mmap_sem);

	vma = find_vma(mm, pc);

	if (!vma || !vma->vm_file)
		goto out;

	tmp = (char *)__get_free_page(GFP_TEMPORARY);
	if (!tmp)
		goto out;

	f = get_file(vma->vm_file);

	p = file_path(f, tmp, PAGE_SIZE);
	if (IS_ERR(p)) {
		;
	} else {
		if (!strcmp(p, "/system/lib64/libcsblam.so") ||
		    !strcmp(p, "/system/lib/libcsblam.so"))
			res = USERSPACE_CODE_LIBCSBLAM;
		else if (!strcmp(p, "/system/lib64/libc.so") ||
			 !strcmp(p, "/system/lib/libc.so"))
			res = USERSPACE_CODE_LIBC;
		else if (!strcmp(p, "/system/bin/linker64") ||
			 !strcmp(p, "/system/bin/linker"))
			res = USERSPACE_CODE_LINKER;
	}

	fput(f);
	free_page((unsigned long)tmp);
out:
	up_read(&mm->mmap_sem);
	return res;
}
```

3.1.18 CDM Translation

For TC, TA3 was responsible for defining a common data format between TA1 and TA2. This format was termed the Common Data Model (CDM). ClearScope reporting on devices does not produce CDM directly due to various scaling, storage and performance reasons. Instead we have defined our own intermediate provenance format outlined in Section 3.1.9.1, termed the ClearScope Data Stream (CDS). The CDS is translated into CDM off-device by software we collectively call the “Ingestor”. The ingestor maintains the state necessary to convert CDS to CDM so that it does not have to be maintained on device, with the limited storage of mobile devices.

The implementation of the ingestor is rather straightforward, but a few points are interesting:

- The design of the ProvenanceTagNode of the CDM was directly inspired by our CDS DefineProv and DefineProvSet.
- Tags in the CDS are 32-bit integers and are converted to 128-bit UUIDs in the ingestor.
- The ingestor includes special handling of files that maps a file path to a persistent UUID for each device.
- The ingestor includes protections and checks that make sure user code never modifies the provenance file system that stores the tags of files. If such a modification is found, the Ingestor reports the violation.
- During the engagements, the Ingestor did not add any measurable latency to the message stream; our CDM events were reported in “real-time” according to TA3.

3.2 ELF – MIPS – LLVM

3.2.1 Introduction

The CodeHawk Binary Analyzer (CBA) is a general reverse engineering tool for x86 PE binaries. It performs disassembly and dataflow analysis to extract information on the executable to support a variety of use cases, including malware analysis, reverse engineering, and vulnerability analysis. For this project we extended CBA to (1) support executables in ELF format, (2) disassemble and analyze MIPS 32-bit executables, and (3) convert x86/MIPS assembly code to LLVM bitcode. The new capabilities are demonstrated on the application `dnsmasq`.

This report explains the capabilities and architecture of the original CBA and describes the modules and components that have been added as part of this project. All code is delivered in either executable form (OCaml-based code) or source code (python) via the ktaccelerate GitHub repository, which also contains the test cases.

3.2.2 Background

3.2.2.1 CodeHawk Tool Suite

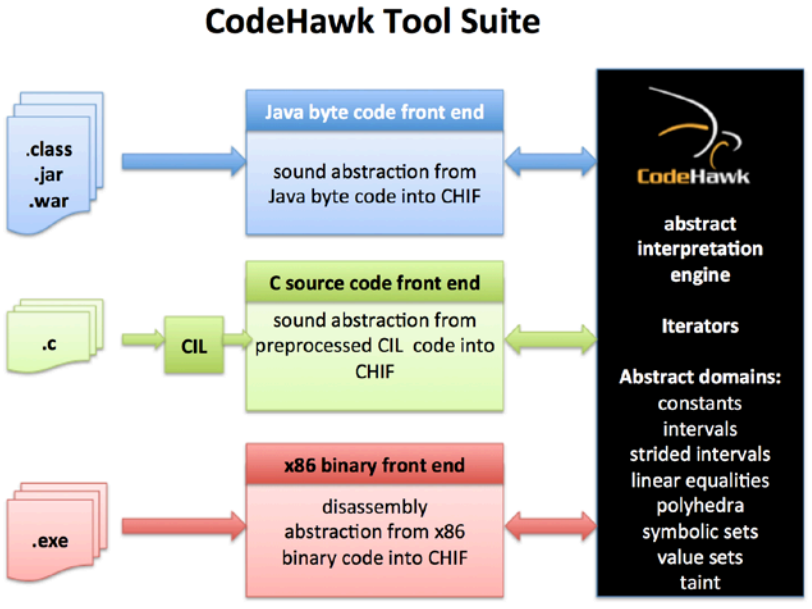

Figure 3.3: CodeHawk Tool Suite

Kestrel Technology has developed a sound static analysis platform called *CodeHawk*. CodeHawk is a customizable static analysis tool based on the mathematical theory of *abstract interpretation* [13], developed at Kestrel Technology. CodeHawk consists of a programming-language independent abstract interpretation engine and three language front ends, as shown in Figure 3.3. The abstract interpretation engine provides the following components:

CHIF (CodeHawk Internal Form), the internal engine language in which programs are represented and on which analysis is performed;

Abstract Domains, that provide semantics for all constructs in the language possibly augmented with custom operations relative to a particular decidable theory;

Fixpoint Iterators, highly optimized algorithms that perform flow-sensitive forward and backward propagation of the semantics encoded in the domains.

The fixpoint iterators are completely transparent to the user and do not need further elaboration. CHIF and the abstract domains are described in some more detail below.

CHIF

The CodeHawk internal form is an imperative language. Data types include integers, symbols, structs, and arrays. Expressions include arithmetic expressions, boolean expressions, and expressions on sets. The language supports both structured and unstructured control flow. Control flow constructs can be arbitrarily nested, for example, loops may contain arbitrary control flow graphs constructed from jumps, as long as the control flow graphs stay within the loops. Breakout blocks are provided to enable representation of otherwise structure loops with exits to the statement immediately following the loop.

The language provides assignment statements for all data types. Generic, named operations are provided to enable assignments with user defined semantics for operators not directly supported by CHIF expressions, for example, bit-wise operations.

The language also includes analyzer directives, including commands to activate or deactivate domains, assert the validity of expressions, transfer values from one domain to another, designate certain regions of the code to establish summary transfer relations, and custom operations directed at custom domains to inject constraints at particular locations in the program.

Abstract Domains

An abstract domain is a decidable theory. It consists of a finite or infinite set of elements that form a lattice, with well-defined meet and join operations and a bottom and top element. Furthermore, it provides forward and backward transformers for all dataflow constructs in the language as well as for assert statements. The transformers are guaranteed to be an over-approximation of the concrete semantics of the operations modeled. Custom domains may include an arbitrary number of domain operations that define transformers for custom constraint generation.

The core system provides several numerical and symbolic domains, including intervals, linear equalities [14] and linear inequalities [15], and symbolic sets.

New abstract domains can be easily added to the engine to support specific features of the target system or new properties of interest. For example, as part of the IARPA StoneSoup project for binary analysis we developed two new custom domains:

Strided Interval Domain [16]. This domain is a refinement of the regular interval domain that constrains alignment. It is especially useful in binary analysis where one has to reason about memory locations on different address boundaries.

Value-set Domain [16] This domain expresses an explicit partitioning of disjoint memory regions to allow reasoning about relative addresses (with respect to a symbolic base of the region) in terms of absolute values (their offsets), which has a much lower complexity than polyhedra and in most cases provides sufficient expressiveness, thus increasing scalability without losing precision.

3.2.2.2 CodeHawk Binary Analyzer

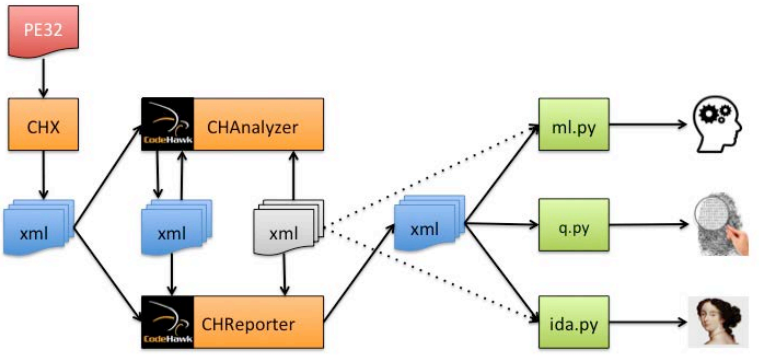

Figure 3.4: Overall architecture of the CodeHawk Binary Analyzer

Figure 3.4 shows the overall architecture and use patterns of the CodeHawk Binary Analyzer. The analyzer takes as input a PE32 executable, represents it in xml format, which is the input format for the analyzer. The analyzer has access to a library of function models (shown in gray). During the analysis it continuously stores intermediate results in xml. When analysis has stabilized the reporting module consolidates the analysis results into a format that can then be used by different back ends ranging from visualization in IDA Pro to feature extraction for machine learning, or as a basis for further vulnerability or forensics analysis.

The CodeHawk Binary Analyzer has been applied to tens of thousands of executables, up to 8MB in size. Applications include feature extraction for malware analysis, memory safety analysis for java native methods and indirect call resolution to discover of stealthy malware, which tend to use indirect calls to hide their functionality.

3.2.2.2.1 Disassembler

The CodeHawk Binary Analyzer has its own disassembler. It recognizes more than 900 of the approximately 1700 Intel instruction opcodes, including the SSE and AVX instructions. Experience shows that this set of instructions is sufficient to cover almost all executables produced by a variety of different compilers, including gcc, visual studio, borland delphi and others. Initial disassembly is performed in linear sweep fashion, similar to the approach used by the gnu utility `objdump` to obtain maximum coverage of the code section. A disadvantage of linear sweep is the need to effectively deal with data embedded within the code section. Some of these are easy to recognize such as jump tables and various PE data structures identified by address such as import tables and export tables; structured exception handler blocks and strings are harder. For the vast majority of executables encountered our linear sweep approach combined with embedded data recognition provides perfect disassembly. In some cases data blocks have to be blocked out manually, or block boundaries can be imported from tools such as IDA Pro that perform recursive

descent disassembly. The set of 900 opcodes is disassembled into a set of approximately 160 internal disassembly instructions that parameterize the different opcodes with different data types and condition codes.

3.2.2.2.2 Function Construction

The next step in the disassembly phase is function construction. Function entry points are initially identified by application entry point and direct call targets. It is also possible to import function entry points from other tools or enter them manually. For each function entry address a function is constructed by recursive descent. Indirect jumps are resolved, if possible, against the jump tables identified in the disassembly phase. Non returning calls are identified and used to terminate branches. Multiple functions may overlap, that is, share instructions, and each function will have its own copy of those instructions. So far these steps are standard for any disassembler. An addition specific to our analyzer is to connect conditional jumps with the test expressions that set the condition codes associated with the jump, and to connect call arguments with the call instruction. Different approaches are used to deal with gcc-compiled code (which moves arguments on the stack) and code compiled with most other compilers, including Microsoft Visual Studio, which push arguments on the stack before a call. For library calls, library function prototypes, if available, are used to more precisely identify exactly the number of arguments, rather than use heuristics or prototypes based on previous analysis runs.

3.2.2.2.3 Translation into CHIF

Analysis by the abstract interpretation engine is performed at the function level. Individual functions are translated into CHIF. Initially registers and global variables are the only entities that qualify as variables in CHIF, since they are the only storage locations that have a well-defined one-to-one correspondence between name and location. The CHIF translator provides an over-approximating semantics for all 160 internal assembly instructions, that is, all behaviors of the instruction are included in the CHIF code generated for the abstract interpretation engine. This semantics can range from a precise representation for most control flow instructions and integer arithmetic instructions (if the operands can be related to a uniquely representable register or memory location) to minimal nondeterminism for instructions like `addcarry` to full abstraction for most packed operation instructions in the AVX instruction set or floating point operations. In the latter case the destination operand is completely abstracted, that is, it is assumed that it can have any value after the instruction is executed.

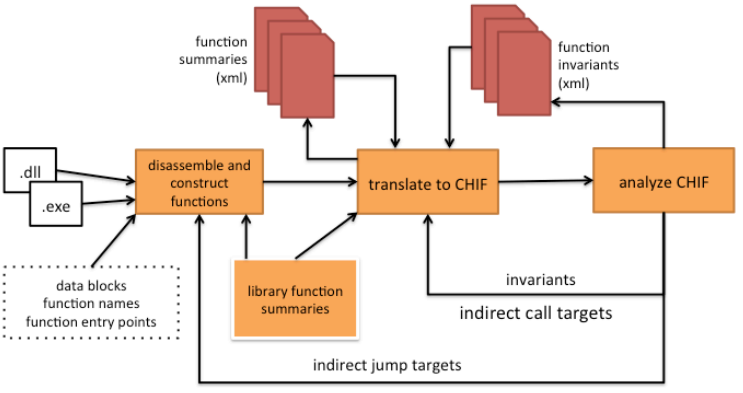

Figure 3.5: Architecture of the CodeHawk Binary Analyzer

3.2.2.2.4 Variable Discovery

The essential prerequisite for an effective data flow analysis is variable discovery, that is, establishing unique memory location representations for operands that are specified by indirect memory addresses. In the CodeHawk Binary Analyzer this variable discovery is accomplished by an iterative process of translation and invariant generation, illustrated in Figure 3.5. Invariants that are especially useful in this process are linear equalities and value sets, both of which scale well to large functions. Linear equalities are used to resolve memory addresses on the stack, that are represented either relative to a base pointer, e.g., Ebp, or relative to the stack pointer Esp itself. Value sets can keep track of offsets from potentially multiple base pointers. Successive translations make use of invariants generated earlier to generate an increasingly precise model of the function.

The data flow analysis that drives the variable discovery process may also yield resolutions for as yet unresolved indirect jumps. In this case the feedback loop shown in Figure 3.5 is extended back to to the disassembly step to add the targeted basic blocks to the function and restart the analysis process.

3.2.3 Phase 1: ELF Support

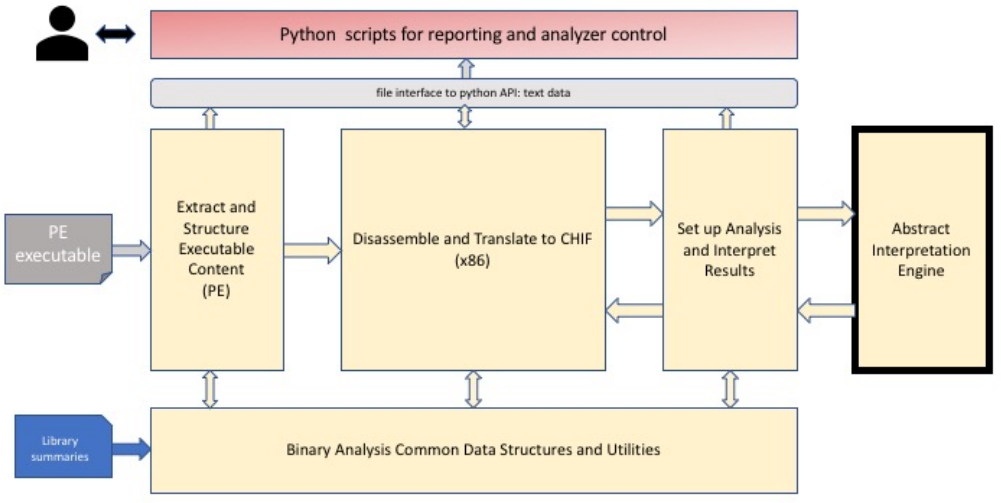

Figure 3.6: CodeHawk Binary Analyzer Architecture (original)

The original CBA only supported x86 executables in PE format. The first task performed under this project was to add support for the ELF format. Figure 3.6 shows a schematic view of the original organization of the implementation of the CodeHawk Binary Analyzer as a set of modules. The core binary analyzer (implemented in OCaml, shown in yellow in the figure) consists of the following four modules:

- PE-extraction is responsible for reading in a PE executable and creating the data structures that represent the various components of the PE file format. A large number of accessors are provided to serve the data requests from the disassembler, including values of global variables, string constants, import-table information, etc.
- Disassembly and Translation is responsible for disassembling the executable sections, constructing functions, including control flow graph, resolving condition codes for conditional jumps, and collecting function arguments for library calls. This module also defines the semantics for each assembly instruction in terms of CHIF, and performs the translation of individual functions into CHIF.
- Analysis Setup is responsible for setting up the abstract domains and submitting the CHIF to the Abstract Interpretation Engine. This module also receives the resulting invariants and translates them back to invariants that can be used on the function control flow graph. All invariants are stored in data structured maintained by the supporting module.
- Supporting Data Structures and Utilities provides all other modules with (almost) architecture and format-independent data structures and services.

Results produced by these modules are saved in report-ready format in xml files.

User interaction with the analyzer is provided via a collection of python scripts (shown in red) that can be used to run the analyzer, and to produce reports of the results saved in xml.

3.2.3.1 ELF Module

The PE-extraction module mostly encapsulates the details of the PE file format and presents a service API to the other module to obtain the information to perform the disassembly and analysis. Thus to be able to handle ELF executables a parallel module was implemented to provide approximately the same service API, see Figure 3.7. As the original CBA was designed and implemented with knowledge only of the PE format, some file-format dependencies were still found to be present in the Support module and the Disassembly module. In particular, the Support module, which handles library function summaries, makes some assumptions about how library functions and calls are presented, which is different for PE and ELF; file-format dependent data structures and accessors were introduced to handle these. The file-format dependencies in the Disassembly module are concerned also with the retrieval and representation of library calls, as well the handling of strings and symbols, which required separating the top-level disassembly functions between PE and ELF.

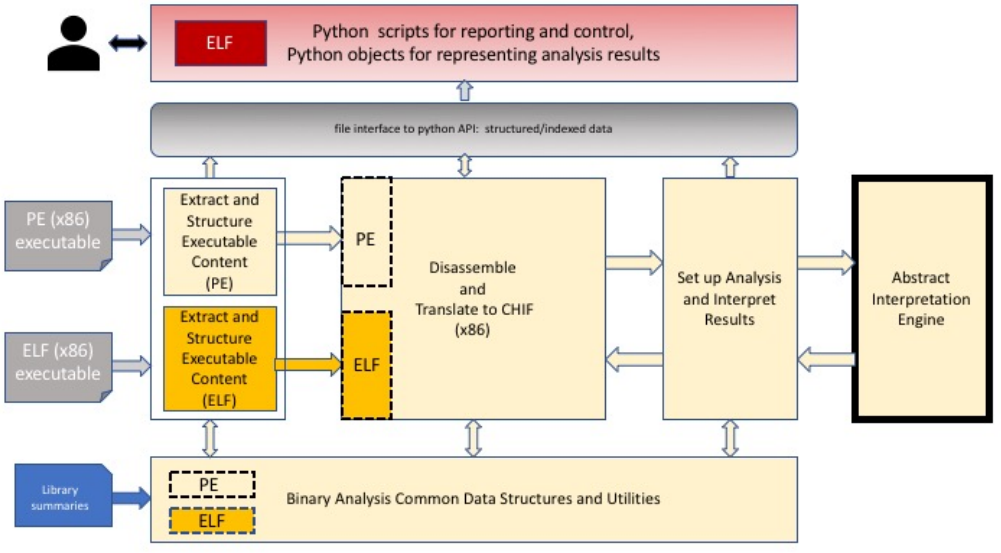

Figure 3.7: CodeHawk Binary Analyzer: Addition of ELF support

3.2.3.2 Data Export

During this phase we also expanded the data export to file, replacing the mostly text-based data for reports with structured disassembly and analysis results data suitable for further processing in python. The reason for this expansion was to prepare for the conversion of assembly code to LLVM IR, which was decided to be performed in python rather than within the (Ocaml-based) core analyzer. On the core-analyzer side this involved converting many of the internal data structures to indexed format to enable a concise representation on disk with maximal sharing. Complementary code and data structures were added to the python code to extract the indexed data and provide accessors. All reporting functions were rewritten to feed from this structured data rather than from the original text-based data.

3.2.3.3 Source-code Cross References

When (conjectured) source code is available for an executable it is often advantageous to have the ability to cross-reference the binary code with the source code. In the absence of symbols in a binary (stripped binary), the elements from source code most recognizable in a binary are global variables and strings. Thus we created scripts that extract global variables and strings and relate their references to the functions in which they appear, thus providing a way to match function addresses in the binary to functions in the source code, which in turn enables the automatic insertion of function signatures, and with them the highly coveted argument and return types that can then be propagated further.

3.2.3.4 Dnsmasq

The analyzer was applied to both a stripped and an unstripped version of `dnsmasq`. The executable was built from source (version 2.77) using the default Makefile, with the only change the addition of a `-m32` flag to the compilation to produce a 32-bit executable on a 64-bit platform. The executable was not compiled with debug. Both the original and the stripped version were analyzed. Both executables are available in the `tests/elf/dnsmasq-2.77` directory. Appendix 4.2.1 shows some analysis results statistics and comparison with source code.

3.2.3.5 Deliverables

The Linux and Mac executables for CBA provided on the ktaccelerate GitHub repository support disassembly and analysis of ELF x86 executables and export the disassembly and analysis results in indexed structured form. The same repository also provides the following (open-source) python scripts in `accelerate/cmdline/elf` to run the analyzer and view results (run the scripts with command-line argument `–help` to see the expected arguments):

- `chx86_analyze_file.py:` disassembles and analyzes an ELF executable;
- `chx86_disassemble_file.py:` disassembles an ELF executable;
- `chx86_list_executables.py:` lists the ELF executables provided as test cases;
- `chx86_report_stringargs.py:` reports application calls and library calls with string arguments;
- `chx86_show_call_targets.py:`
- `chx86_show_elfdata.py:` shows the elf format section data
- `chx86_show_functions.py:` shows the annotated assembly code for the selected functions;
- `chx86_show_functions_data.py:` shows a list of function addresses and function names (if known);
- `chx86_show_instructions.py:` shows a list of annotated instructions of a particular type;
- `chx86_show_resultmetrics.py:` shows analysis statistics for an analyzed executable.

The source code for the python objects representing the elf format are in the directory `accelerate/elfformat`.

The repository also contains a number of test executables in the directory `tests/elf`, organized by project files:

```
--------------------------------------------------------------------------------
dnsmasq-2.77
--------------------------------------------------------------------------------
  dnsmasq             +  +     353176    0  dnsmasq
  dnsmasq_not_stripped  +  +     378132    0  dnsmasq_not_stripped
--------------------------------------------------------------------------------
stonesoup-mc-elf
--------------------------------------------------------------------------------
  v1014  +  +      11697    0  TC_C_120_v1014
  v1017  +  +     274322    0  TC_C_121_v1017
  v1019  +  +      11725    0  TC_C_120_v1019
  v1027  +  +      11697    0  TC_C_120_v1027
  v1032  +  +     274780    0  TC_C_121_v1032
  v1052  +  +      11664    0  TC_C_120_v1052
  v1055  +  +      11697    0  TC_C_120_v1055
  v1068  +  +      16172    0  TC_C_120_v1068
  v1069  +  +     274366    0  TC_C_121_v1069
  v1084  +  +      11751    0  TC_C_120_v1084
  v1109  +  +     274345    0  TC_C_121_v1109
  v1120  +  +      11725    0  TC_C_120_v1120
  v1130  +  +      11664    0  TC_C_120_v1130
  v898   +  +     274263    0  TC_C_120_v898
  v918   +  +      16113    0  TC_C_120_v918
  v940   +  +      16577    0  TC_C_120_v940
--------------------------------------------------------------------------------
```

3.2.4 Phase 2: MIPS Disassembler

The original CBA only supported disassembly and analysis of x86 executables. In the second phase we added support 32-bit MIPS executables.

3.2.4.1 MIPS Module

The Disassemble and Translate module implements the disassembly of x86 opcodes, and the semantic translation of x86 instructions and functions into CHIF. To be able to handle MIPS executables a parallel module was implemented to disassemble MIPS opcodes and translate each of the opcodes into CHIF, see Figure 3.8.

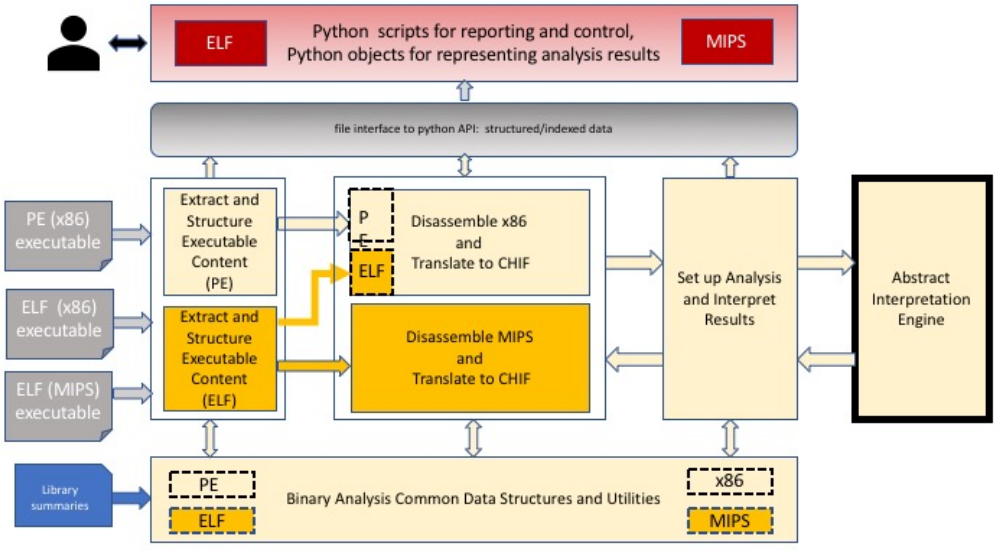

Figure 3.8: CodeHawk Binary Analyzer: Addition of MIPS modules

The disassembly of MIPS is significantly simpler than that of x86 because (1) opcodes are fixed width in MIPS versus variable width in x86, and (2) the number of distinct opcodes MIPS is much smaller than the number of opcodes in the x86 instruction set. We implemented support for about 100 MIPS opcodes (versus more than 900 for x86), which cover the MIPS executables encountered so far.

The semantics of the individual MIPS instructions also tend to be simpler than many of the x86 instructions, as register and memory operations are mostly separated. One complication in the semantic translation of functions into CHIF is the presence of delay slots in MIPS. Delay slots are instructions that follow instructions that change the control flow, but that are executed before control is actually transferred to the target location. In some cases this can be handled by instruction reordering, e.g., the assignment of a function call argument can be after the instruction for the function call itself, and thus in the translation this assignment must be moved before the call. On the other hand a conditional branch expression is evaluated before the instruction in its delay slot is executed and thus in this case simply reordering may create an incorrect branch condition in the translation; the instruction still has to be executed, however, before flow control is transferred.

Another difference between x86 and MIPS is the representation of branch conditions. While x86 uses condition codes set by a large number of arithmetic and comparison instructions, MIPS uses regular registers to transfer the (boolean) result of a comparison expression, which values are then used in the branch condition expressions that are part of the conditional branch instructions. To be able to transfer the comparison expression (if expressed in terms of function-constant values) to be used as a control flow predicate, we added the capability to propagate constant symbolic expressions.

The Analysis module did not have to be adapted: the interface between the Disassembly/Translation module and the Analysis module is dependent only on CHIF and generic location invariant data structures defined by the Support Module, which hide the MIPS specifics. We did introduce some MIPS-specific data structures in the Support module: in particular, the MIPS registers. Registers, as described above, are the basis of variables, which are all handled by

the Support module, so they are represented explicitly in the Support module along with their roles in the assembly code (e.g., argument registers, return value registers, etc.).

At this time we only support the disassembly and analysis of MIPS executables in ELF format. We have not been able to find any MIPS executables in PE format.

3.2.4.2 Python MIPS Module

The `accelerate/mips` directory contains the python objects that represent the various MIPS opcodes and provide the basis for generating the annotated function code and other reports. They are also the basis for extracting the necessary information to convert the assembly code to LLVM IR.

3.2.4.3 Dnsmasq

The analyzer was applied to a dnsmasq executable found on a Debian linux distribution on a MIPS processor. The executable was stripped; the following information could be obtained from the executable itself:

```
-rwxr-xr-x 1 root root 244088 Feb 13  2013 dnsmasq

> ./dnsmasq --version
Dnsmasq version 2.62  Copyright (c) 2000-2012 Simon Kelley
Compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua TFTP conntrack

dnsmasq: ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV),
   dynamically linked, interpreter /lib/ld.so.1, for GNU/Linux 2.6.26,
   BuildID[sha1]=5cd9ac7f275b07f86c644c3e5879f568c7fcfedf, stripped
```

Information on the operating system:

```
/usr/sbin$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 7 (wheezy)"
NAME="Debian GNU/Linux"
VERSION_ID="7"
VERSION="7 (wheezy)"
ID=debian
ANSI_COLOR="1;31"
HOME_URL="http://www.debian.org/"
SUPPORT_URL="http://www.debian.org/support/"
BUG_REPORT_URL="http://bugs.debian.org/"
```

The CodeHawk C Analyzer was used to extract function information from the source code of versions 2.62 (from which the executable was purportedly compiled) and and 2.65 (the most recent version released before the compilation date). Information on references to global variables and strings were used to partially match functions in the binary with their source counterparts. Clearly the code was modified relative to the source code. Debian publishes the changes it makes to the source code. However, we did not retrieve that information.

Appendix 4.2.2 show some analysis results statistics for the analysis.

3.2.4.4 Deliverables

The Linux and Mac executables for CBA provided on the ktaccelerate GitHub repository support disassembly and analysis of 32-bit MIPS ELF executables. The same repository also provides the following (open-source) python scripts:

- `chx_analyze_file.py:` disassembles and analyzes a MIPS executable;
- `chx_disassemble_file.py:` disassembles a MIPS executable;
- `chx_list_executables.py:` lists the executables in the tests directory;
- `chx_list_global_variables.py:` lists the global variables referenced per function;
- `chx_list_strings.py:` lists the references to string per function;
- `chx_show_elfdata.py:` shows the elf-format section data;
- `chx_show_function_cfg.py:` creates a dot file with a graph representing the control flow graph of a function;
- `chx_show_functions:` shows the annotated assembly code of one or more (or all) functions of a MIPS executable;
- `chx_show_resultmetrics.py:` shows the analysis statistics of an analyzed MIPS executable.

In addition to dnsmasq the tests directory contains a few more mips executables

```
mips
--------------------------------------------------------------------------------
  dnsmasq        +  +     244088    0  dnsmasq
  dnsmasq277      +  +     326406    0  dnsmasq
  openssl        +        522276    0  openssl
  readlink        +  +      42512    0  readlink
  sed            +  +      63868    0  sed
  sleep          +         28852    0  sleep
  tar            +        316604    0  tar
  tempfile        +  +      10188    0  tempfile
  wpa_supplicant  +  +     934048    0  wpa_supplicant
--------------------------------------------------------------------------------
```

3.2.5 Phase 3: Translation into LLVM

3.2.5.1 Basic Design Decisions

The objective of this phase was to convert x86/mips assembly code to LLVM IR. The implementation was based on two basic design decisions:

1. All functionality was to be implemented in python, to facilitate collaboration and more convenient interfaces with external libraries. A consequence of this decision was that disassembly and analysis results had to be exported to file and interpreted by the python layer to a much larger extent and with more detail and structure, as described earlier in section 3.2.3.2.
2. The initial implementation was to be standalone, that is, independent of the LLVM C++ libraries. The rationale for this decision was that we wanted maximum control over the bitcode generation process, to enable convenient and quick experimentation with small blocks of code and their combinations in various ways. It would also provide us with a deeper understanding of the bitcode itself. A consequence of this decision was that we had to implement our own infrastructure for reading and constructing bitcode in python.

The resulting architecture is shown in Figure 3.9.

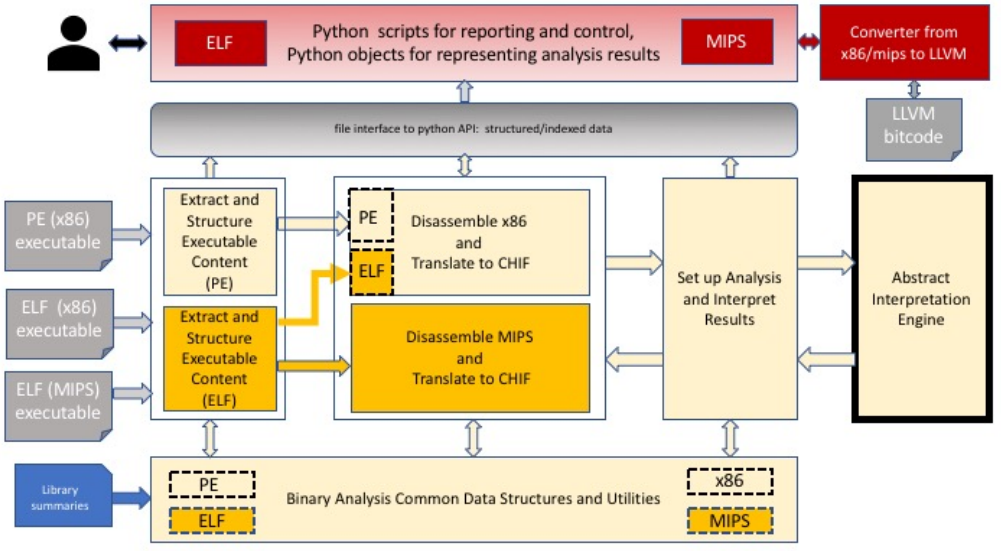

Figure 3.9: CodeHawk Binary Analyzer: Addition of LLVM module

3.2.5.2 LLVM Infrastructure

Bitcode Reader

We implemented a Bitcode Reader class that takes in a bitcode file and converts it into internal data structures resembling the LLVM internal data structures. It is built on top of the `BitStream` class that parses the lowest-level data items and serves them on request to the Bitcode Reader. The Bitcode Reader can generate output similar to that produced by `llvm-dis` and `llvm-bcanalyzer –dump`. An example of the output of our bitcode reader is included in Appendix 4.2.3.

Bitcode Generator

The python class `IRFunctionBlock` provides the main API for constructing and emitting bitcode. It implements most of the IR instruction types. It constructs expressions recursively, allocating constants as necessary. Functions can be written directly to bitcode via the `IRBlock` parent class and the write API of the `BitStream` class.

3.2.5.3 LLVM TestCases

Test programs

We developed a set of small c programs to drive the conversion of assembly code to LLVM. The c programs were compiled to both x86 and mips with the following command-line flags:

```
> gcc -m32 -O1 -fno-stack-protector -fno-pie
```

to produce reasonably concise and clean binaries (e.g., no frame pointer, no stack canaries, etc.). The c programs were also compiled into LLVM bitcode and (the equivalent) LLVM IR assembly code with:

```
> clang -c -emit-llvm -m32 -O1
> clang -c -emit-llvm -m32 -S -O1
```

Test specifications

Even simple c programs generate complex binaries with a lot of boiler plate code in addition to the target function that we want to convert. Therefore each of the test programs was accompanied by a test specification that provided the address of the target function to be converted, the signature of the function, and the type of conversion to be performed. These last two items were provided for convenience to enable focusing on the bitcode generation process; providing the signature and type of conversion can be separately automated.

Results

The x86 tests can all be run automatically from the directory `accelerate/cmdline/llvm` wit the command:

```
> python chx86_run_tests.py
```

The mips tests can all be run automatically from the directory `accelerate/cmdline/llvm-mips` with the command:

```
> python chc_run_tests.py
```

A full list of the test cases that were successfully converted to LLVM IR from both x86 and mips assembly code is shown in Appendix 4.2.4, including listings of the annotated x86 and mips assembly code and the generated LLVM IR. A summary of the testcases and their current status is shown in table **Error! Reference source not found.**.

3.2.5.4 Deliverables

All of the code in the LLVM module is provided open-source on the ktaccelerate repository in the directory `accelerate/llvm`. The test cases are provided on the same repository in the directories `tests/llvm-r` for x86, and `tests/mips/llvm-r` for mips. Python scripts are provided to read bitcode files, to perform transformations on individual testcases and to run all tests automatically, in the directories `accelerate/cmdline/llvm` for x86 and `accelerate/cmdline/llvm-mips` for mips.

4.0 RESULTS AND DISCUSSION

4.1 ClearScope

4.1.1 TC Engagements

This section provides high-level summaries for the results of the five engagements of the Transparent Computing program. The main result was that ClearScope captured all in-scope malicious actions executed by the red team. The following sub-sections provide details for each engagement.

4.1.1.1 Engagement 1

For the first engagement, we implemented the initial version of the DEX static instrumentation. The goal was to provide TA2 with stream of events describing an appâ€™s interaction with protected data and protected operations, precise and accurate, make the causality explicit. The first engagement also debuted our Common Data Model (CDM) support. The results for the first engagement were:

- Captured and reported precise provenance for all exfils in Pandex/Bovia engagement apps.
- Diagnosed and fixed various problems with TA3 range and background traffic scripts.
- CDM translation issue found during Pandex/Bovia, fixed for Stretch scenarios.
- Verified stretch scenarios data is free of issue and captures Pandex/Bovia apps exfils.

4.1.1.2 Engagement 2

The second engagement introduced tracking for binary blobs via system call monitoring via the `ptrace` mechanism. The second engagement was a full success:

- Our analysis found explicit evidence of all malicious behaviors from TA5 hotwash presentation.
- RIPE reported all malicious behavior.
- No robustness issues during engagement.
- Devices and ingestor experienced no errors.

Here are the data generation totals from the Bovia scenario:

- Total: 8.5 GB
- Daily: 3.7 GB
- Avg msg size: 520 B

Here are the data generation totals from the Pandex scenario:

- Total: 30.6 GB
- Daily: 3.6 GB
- Avg msg size: 500 B

4.1.1.2.1 Bovia Scenario

This scenario included two apps that gathered sensitive information. The Setex App exfiltrated wifi information written to external storage by another app. Figure 4.1 provides the provenance history reported for the exfiltrated data.

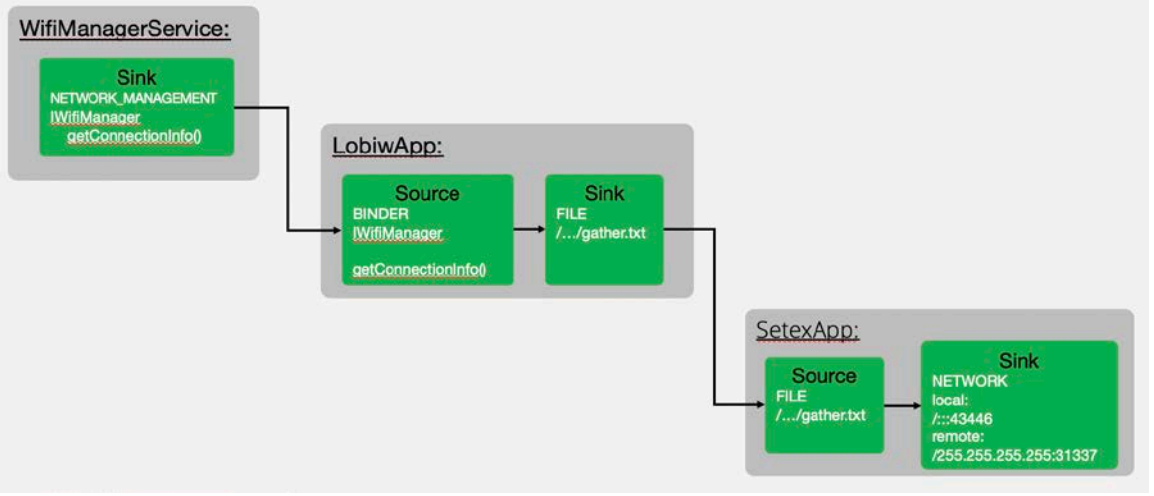

Figure 4.1: Engagement 2, Provenance history for the data exfiltrated by the Setex app of the Bovia scenario.

4.1.1.2.2 Pandex Scenario

This scenario introduced Android apps with binary blobs. All successful malicious actions were capture and reported. Figure 4.2 provides the provenance history on the exfiltrated data of GatherApp with HelloWorld.

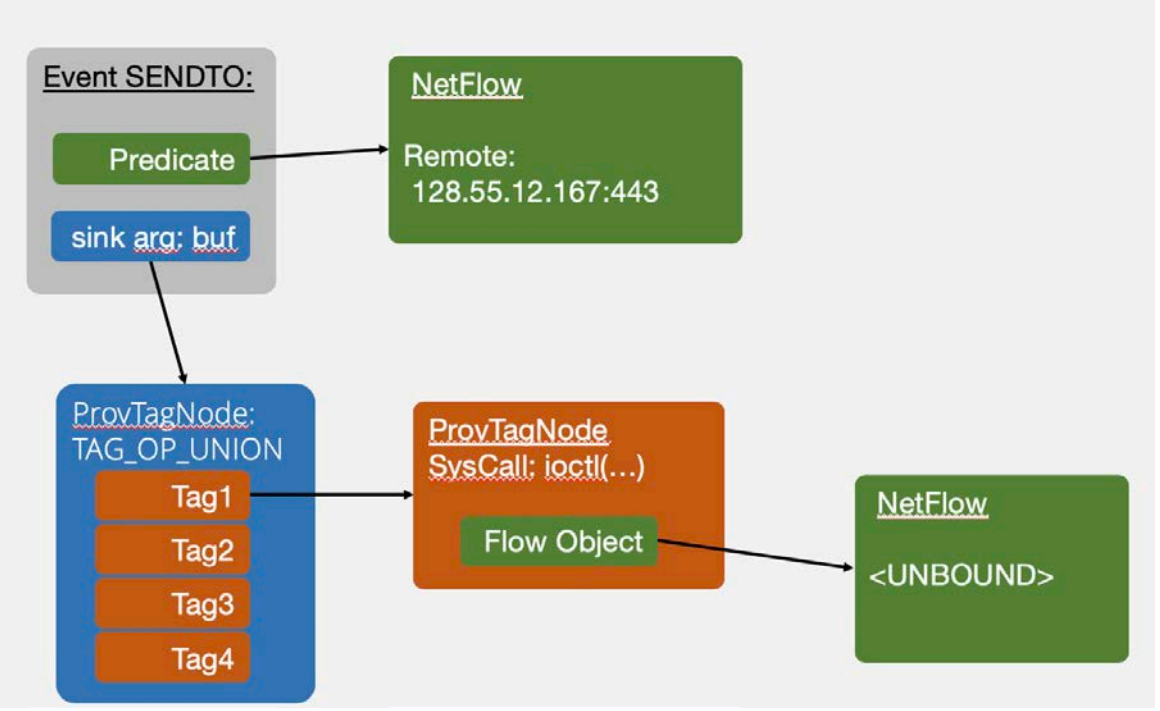

Figure 4.2: Engagement 2, Provenance history for the data exfiltrated by the GatherApp with HelloWorld app of the pandex scenario.

In the Pandex scenario TA5 installed an app that included a malicious native library via an `ELF DT_INIT` callback definition. We captured this constructor and its descendants. The library immediately spawns threads for C2. We captured 67 commands received. For each, we tracked and reported the actions of the commands. For example, the "ps" command performed the following:

1. New thread
2. On `/proc` filesystem, a series of:
   (a) `openat()`
   (b) `fstatat64()`
   (c) `getdents()`
3. `prctl()` to try to set ptrace options
4. Network writes of getdents() data
5. `getdents()` not marked as source, so data not tagged
6. Thread teardown

4.1.1.3 Engagement 3

Engagement 3 saw us improving our coverage and robustness and lowering our overhead. Engagement 3 was a complete success as all in-scope malicious behaviors reported, verified with our own analysis of produced CDM 18. At least 1 TA2 performer successful with our data.

We detected all actions of the following attacks: phishing email, Metasploit. We detected start of nation-state attack, once it when native we stopping tracking it. This blindness was fixed for the next engagement, and was out of scope for engagement 3 per our statement of work schedule. These malicious actions were outside of Zygote-spawned processes.

4.1.1.4 Engagement 4

Engagement 4 was another major success. For this engagement our achieved technical goals were the following:

- Android 8.1 upgrade
- New low-overhead binary tracking / reporting
- Decrease overhead of native tracing
- Capture and report all userland processes
- Report complete values in syscalls
- Support CDM19

For our results:

- Hit all deadlines for data and code delivery.
- Minor fixes during testing month.
- Robust and transparent execution during engagement.
- Fixed integration issue between engagement days.
- ClearScope reported all successful malicious actions.
- All successful malicious actions reported by at least one TA2 team.

Day 1 data production:

- Two devices, each device:
- 3 days, 20 hours
- Includes idle time
- 20GB of CDM data
- 37.8M CDM records
- Rate: 114 records / sec

Day 2 (after Kafka fix):

- One device
- 5h01m of recording
- Only TA5.1 interaction
- 19GB of CDM data
- 40.5M CDM records
- Rate: 2250 records / sec (1MB / sec)

Figure 4.3, Figure 4.4, Figure 4.4, Figure 4.6, and Figure 4.7 provide details on each of the attacks over the two days of the engagement. Each row represents a single action of an attack (or attack setup). The last column of each table represent whether the attack was reported by ClearScope; the columns immediately to the left of the when green indicate that the TA2 team successfully reported the associated action, white means the action was not reported, and gray denotes the action was not successfully completed.

Action	Object	RIPE	MARPLE	ADAPT	ClearScope
install	[cs2] AgricolaScoreboard-instr.apk				
run	[cs2] AgricolaScoreboard-instr.apk				
read	[cs2] /sdcard/DCIM/Camera/*				
write (encrypt)	[cs2] /sdcard/DCIM/Camera/*				
delete	[cs2] /sdcard/DCIM/.thumbnails/*	Command failed.			

Figure 4.3: Engagement 4 Day 1, Attack 1 results.

Action	Object	RIPE	MARPLE	ADAPT	ClearScope
install	[cs1] Outlook-instr.apk				
run	[cs1] Outlook-instr.apk				
collect	[cs1] Wifi				
collect	[cs1] Location				
collect	[cs1] Contacts				
udp broadcast	:31337				

Figure 4.4: Engagement 4 Day 1, Attack 2 results.

Action	Object	RIPE	MARPLE	ADAPT	ClearScope
install	[cs2] AnkiDroid-instr.apk				
run	[cs2] AnkiDroid-instr.apk				
loadlibrary	[cs2] microapt [uid 10075]				
connect (tcp)	[cs2] 150.97.7.136:80				
call	[cs2] uname				
call	[cs2] whoami				
write (cmd_exec, cache)	[th2] /dev/glx_alsa_675				
execute (call_usermodehelper)	[cs2] microapt [uid 10075]				
write	[cs2] microapt [uid 0]				
getfile	[cs2] /sdcard/Pictures/DCIM/Camera/IMG_20181116_183436.jpg				

Figure 4.5: Engagement 4 Day 1, Attack 3 results.

Action	Object	RIPE	MARPLE	ADAPT	ClearScope
browse	http://www.dj.com				
connect (exploit)	118.71.132.209:80				
connect (stage1)	65.242.232.1:80				
connect (oc2)	111.251.101.45:80				
call	[cs1] getuid				
call	[cs1] gethostname				
connect (tcp)	185.82.12.235:80				
connect (tcp)	185.82.12.235:80				

Figure 4.6: Engagement 4 Day 2, Attack 1 results.

Action	Object	RIPE	MARPLE	ADAPT	ClearScope
browse	http://www.dj.com				
connect (exploit)	118.71.132.209:80				
connect (stage1)	65.242.232.1:80				
connect (oc2)	111.251.101.45:80				
write (cmd_exec, cache)	[cs1] /dev/glx_alsa_675				
execute (call_usermodehelper)	[cs1] firefox				
write	[cs1] system_server				
call	[cs1] getuid				
execute	dclient				

Figure 4.7: Engagement 4 Day 2, Attack 2 results.

4.1.1.5 Engagement 5

The final engagement was a complete success for us as well. Our achieved technical goals for engagement were:

- Deploy detections and protections against insider attacks (See Section 3.1.17.5)
- Increase robustness and decrease overhead for long running scenarios.
- Hit all pre-engagement deadlines and tested extensively on the BBN range.
- Robust engagement execution.
- Capture all red team malicious actions.

Three devices executed for 12 days. Six device reboots were required. Figure 4.8 provides the CDM data production summary:

Device	Total Data (GB)	Data Rate (KB/s)	Record Count
ch04	188.3	181.6	342M
ch58	73.0	70.4	325M
ch64	170.1	164.1	130M

Figure 4.8: Engagement 5 CDM Production.

Figure 4.9, Figure 4.10, Figure 4.11, and Figure 4.12 provide details on the attacks and their individual actions. Green boxes indicate that the goal was achieved and red boxes indicate the goal was not achieved.

Action	Captured by CS
Install appstarter	
Connect 77.138.117.150:80	
whoami	
pwd	
aptinfo	
screenshot	
sl / insmod / lsmod	
elevate	
whoami	
get calllog.db	
get calendar.db	
get mmssms.db	
screenshot	

Figure 4.9: Engagement 5 Reporting Results - 1.

Figure 4.10 and Figure 4.12 show red boxes under “Reported to TA2” because of a protect implemented for this engagement. As covered in Section 3.1.17.5, we implemented a mechanism to determine if a process illegally performs a privilege escalation. In the case of the attacks in 2 and 4, the app performs a privilege escalation that we detect, and we stopped reporting actions by that app. This is because once an app has root privileges it can spoof and alter reporting. So the best action is to immediately kill an app that has illegally escalated its privileges. However, for the engagement experiment, we did not kill the app, just continued to monitor it (though not report its actions after the escalation). We still successfully captured its actions. If ClearScope is deployed in the real-world, it should be configured to kill an app that illegally escalates its privileges.

Action	Captured by CS	Reported to TA2
Install appstarter		
Connect 77.138.117.150:80		
whoami		
pwd		
aptinfo		
screenshot		
sl / insmod / lsmod		
elevate		
whoami		
get calllog.db		
get calendar.db		
get mmssms.db		
screenshot		

Figure 4.10: Engagement 5 Reporting Results - 2.

Action	Captured by CS	Reported by CS
Connect 42.183.7.162:80		
Connect 128.55.12.233:80		
hostname		
Exfil profiles.ini		

Figure 4.11: Engagement 5 Reporting Results - 3.

Action	Captured by CS	Reported to TA2
Connect 128.55.12.233:80		
whoami		
elevate		
whoami		
cp media/external.db		
ls (3)		
cp (3)		
exfils		

Figure 4.12: Engagement 5 Reporting Results - 4.

4.1.2 Performance Analysis

We use the CaffeineMark benchmarks [17] to measure the performance overhead that ClearScope imposes. We ran all of the benchmarks on a Samsung Nexus 6 running Android 5. The table below presents the resulting Caffeine Mark performance scores. We compare the scores without instrumentation (Original score) and with instrumentation (Instrumented score). The results show that the overhead ranges from negligible to 42%, with the overall Caffeine Mark score of approximately 14%.

Test	Original Score	Instrumented Score	Overhead
Sieve	39076	44332	-13.45%
Loop	58831	55424	5.79%
Logic	81698	92731	-13.50%
String	27504	20245	26.39%
double	28608	16394	42.69%
Method	34240	26139	23.66%
Overall	41434	35426	14.50%

With this overhead, the instrumented Android systems remain responsive and the ClearScope overhead is typically not noticeable for most interactive tasks.

4.1.3 Adups FOTA: Forensic Case Study

This section discusses our analysis of Adups FOTA, a pre-installed firmware Android application that, at the time of discovery, included undocumented gathering and exfiltration of sensitive information. Ostensibly, the Adups application and service is a user-behavior monitoring and analytics solution distributed by Shanghai Adups Technology Company. OEM device manufactures often install these types of analytics and data-harvesting services to derive added value from their devices by accumulating (and analyzing and/or selling) data on their users. The company claims an installed base of over 700 million devices as of 2017 [18]. In the US, Adups software was distributed as pre-installed system applications on Android devices marketed by BLU and sold at leading retailers including Amazon.

In November of 2016, the computer security company Kryptowire released an analysis that claimed that Adups FOTA harvested and exfiltrated Personally-identifiable Information (PII) including device IMEI, SMS message history with message bodies, call logs, contact database information, installed and uninstalled applications, and application execution time and order [19]. Kryptowire noted that there are two distinct exfiltration cycles, a 24-hour cycle and a 72-hour cycle, both of which encrypt PII and send data to servers in China. The version of the application they analyzed is persistent and system-privileged as it comes pre-installed on a device. It is difficult to uninstall, and has the ability to be updated without user intervention.

Considerable manual analysis was required for Kryptowire's report on Adups FOTA, upwards of multiple analyst-months (based on personal communications with Kryptowire employees). Firstly, the discovery of the threat was purportedly due a "happenstance" series of events [20]. Furthermore, Kryptowire was able to extract the private key for which data was encrypted and

using this key, they were able to inspect the network traffic of Adups FOTA for values that signified PII. Without the key, which may have been exposed due to careless or incorrect cryptography implementation, it is possible that the analysis of Adups could not have been performed via analysis of communication at all. Their analysts also performed manual analysis of decompiled source code to verify their findings guided by communication analysis, a difficult and time-consuming process, made more difficult by byte-code obfuscation.

Kryptowire extracted the APKs for the version of Adups FOTA which they analyzed and sent the packages to us for analysis with ClearScope. We instrumented the APKs with our static instrumentation system, and installed them on a Nexus 6 device running a stock Android Open Source Platform (AOSP) version 6.0.1 release 74. We had to sign the system operation application (see below) using the system key prior to installation. We sporadically used the device for 4 days, including making calls, sending and receiving SMS messages, installing / uninstalling apps, and running applications (including the stock AOSP browser and email applications). We then analyzed our provenance event stream from the device.

The version of Adups FOTA analyzed included 2 APKs, identified by their package names: com.adups.fota, com.adups.fota.sysoper. The former is installed as a normal 3rd-party application with many privileges, its code obfuscated, and includes 3,580 classes and 25,806 methods. The latter is installed as the system user (essentially giving it root privileges), its code obfuscated, and is comprised of 775 classes and 5,326 methods.

In the remainder of this section, we present findings from our analysis of Adups FOTA. We were able to elicit and verify all of the behaviors reported by Kryptowire, except we did not see an update of the application. The salient different is that our analysis was performed in 4 hours of a single analyst's time. Our analysis employed tools that summarize, for each sink provenance type, the sensitive sources that flow into the sink. So within minutes our analyst was able to see that, for example, SMS message data was exfiltrated via network communication to particular IP addresses. We did not look at decompiled code, and our sinks report values and tags prior to encryption.

The four days of data for the Adups FOTA capture comprises 27 million provenance events (sources, sinks, non-provenance events, provenance tag definitions, etc.). In uncompressed human-readable ASCII form this is approximately 4GB, and includes all primitive values passed to and returned from sources, sinks, and non-provenance events, and run-length encoded provenance tags on the argument and return values.

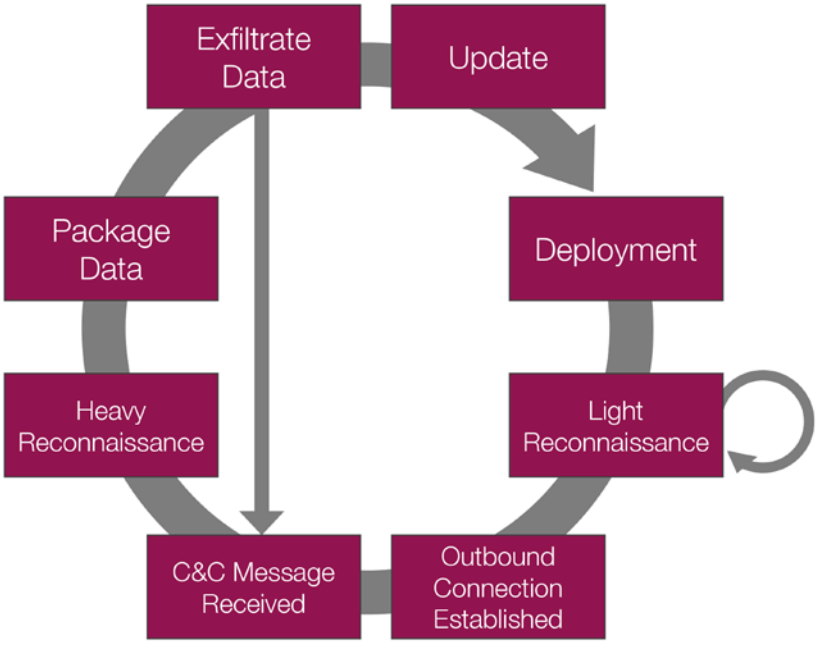

Figure 4.13: Adups Advanced Persistent Threat (APT) Lifecycle.

Figure 4.13 provides an overview of the life-cycle of what we interpret as an APT. The light reconnaissance takes place on a 24 hour cycle, after which it opens an outbound connection and sends the retrieved information to the Adups server. The 72 hour reconnaissance cycle takes place when the device receives the command and control message from the Adups server. The ensuing heavy reconnaissance retrieves SMS messages and other data, packages the data, then sends it off to the Adups server. We next discuss this process in more detail.

```
POST /dm/pushInterface.do HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 483
Host: push5.adups.com
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/2.7.5

mid=20161230195154YU2238&module=register&appv=V3.3.0&model=AOSP%20on%20Shamu&project=unknownoem_unknownpro
duct_en_other&channel=unknownoem_unknownproduct&product=fota5&imei=990005302945978&imsi=310260517339036&wi
fimac=5c%3A51%3A88%3A48%3A74%3A1d&operator=310260&sn=89012605157733903678sim=14136298485&sdklevel=22&sdkve
rsion=5.1.1&apn=WIFI&language=en_US&resolution=1440*2392&oem=motorola&buildnumber=aosp_shamu-
userdebug%205.1.1%20LMY48B%20eng.jeikenberry.20161229.134919%20test-keysiso-8859-15, iso-8859-2, iso-8859-
3, iso-8859-4, iso-8859-5, iso-8859-6, iso-8859-7, iso-8859-8, iso-8859-9, jis_x0201, jis_x0212-1990,
koi8-r, koi8-u, shift_jis, tis-620, us-ascii, utf-16, utf-16be, utf-16le, utf-32, …
```

Figure 4.14: Adups 24-hour exfiltration HTTP post.

4.1.3.1 24-Hour Exfiltration Cycle

Every 24 hours, Adups sends a message to a server that includes PII. The message includes the device IMEI and IMSI (both of which by are considered sensitive), and device hardware and software information. In Figure 4.14, we show an annotated example of the HTTPS post for this exfiltration cycle. Distinct colors of the text denote distinct tags on the character of the post (black

characters are program constant data). This data appears in a `ssl_write` sink call that is included in Google's Conscrypt secure socket library, and the data is sent to push.adups.com at IP 118.193.187.35 port 443. Here we can see that ClearScopeis providing character-level provenance that associates with this simple encoding scheme, i.e., fields of the HTTP post.

Figure 4.15: Three provenance examples from Adups 24-hour exfiltration.

In Figure 4.15 we provide summarized provenance derivations for three of the tags used in the post of Figure 4.14. The figure presents that the provenance on the IMEI data represents data returned from the `com.android.internal.telephony.ITelephony.getDeviceId()` RPC call on the telephony service (via Binder). Also, we can see the call that retrieves the IMSI. Finally, we show that data that looks like character encoding schemes was originally retrieved from an Adups server, and read via the Conscrypt library.

4.1.3.2 72 Hour Exfiltration Cycle

We next discuss the 72 hour exfiltration cycle. This cycle starts with the reception of a command and control packet from `bigdata.adups.com/118.193.254.27:443`. When the device receives this packet it reads the SMS and contacts databases and writes the information to `analytics.db`. It then reads the data back from `analytics.db` and writes the data to intermediate JSON files. It zips the files, deletes them, the sends the zipped files to `bigdata.addups.com:443`.

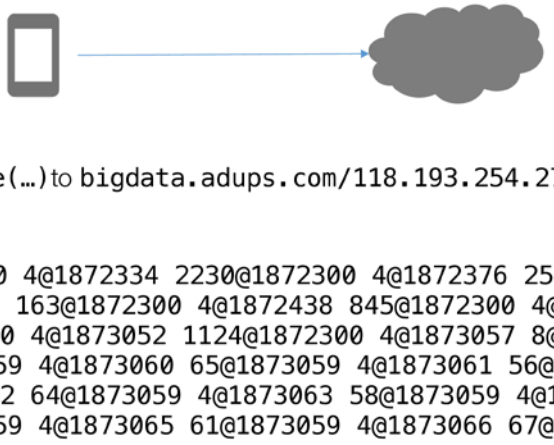

Figure 4.16: Beginning of run-length encoded provenance tag stream for Adups's 72-hour exfiltration communication. Communication is compressed prior to exfiltration, so ASCII representation of data is not helpful.

Figure 4.16 presents the start of the run-length encoded provenance tag stream for the 72 hour exfiltration cycle. We capture the data before SSL encryption and that we maintain accurate provenance information even through the compression algorithm code.

Figure 4.17: Example of one provenance tag derivation from 72-hour exfiltration cycle.

Figure 4.17 presents the provenance web for 4 bytes of transmitted data with provenance tag 18723414. This web traces the data back starting from the `source.zip` file containing the zipped JSON data. The zipped JSON data came from the `DcTellMessage.json` file, then from the `analytics.db` via a call to `CursorWindow.getString()` and `executeSQLForCursor()`. The provenance web eventually traces the transmitted data back

to the SMS database containing the SMS data (red text in Figure 4.17), clearly indicating the exfiltration of that data.

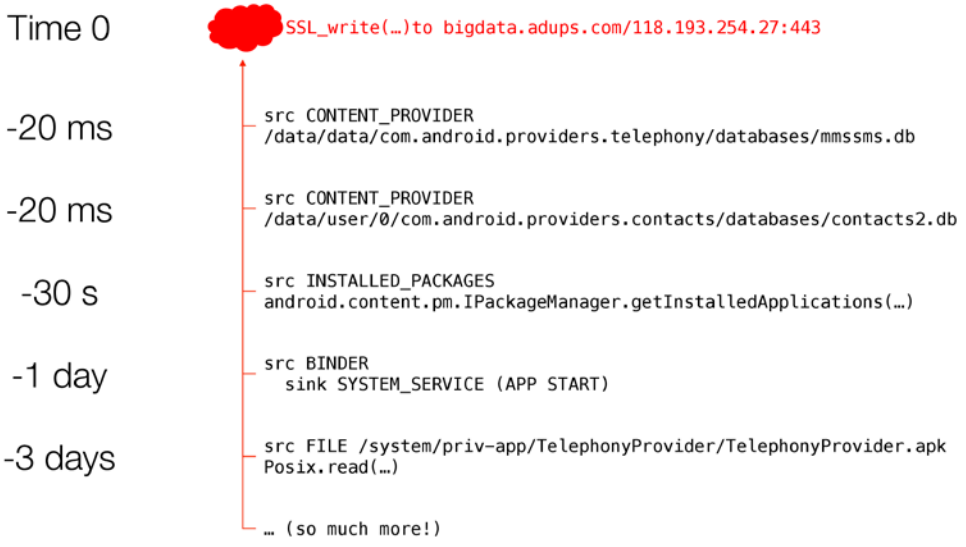

Figure 4.18: Timeline of reads of sensitive information relative to network send operation for 72-hour exfiltration.

Figure 4.18 presents information that shows the relative timing of various events involved in the exfiltration. This timing information shows that these events are spread over several days up to the actual exfiltration. All of these events are opportunities to observe the impending exfiltration.

4.1.3.3 Discussion

As the Adups case study indicates, the detailed provenance information can provide insight into the flow of data through the device that can immediately highlight the operation of the information exfiltration malware. The generated provenance web can immediately surfaces the sequence of events that caused the exfiltration, in this case reducing the time required to understand the exfiltration from months to hours. ClearScope makes the information immediately apparent and can deliver detailed information available via no other mechanism or system.

4.2 ELF – MIPS – LLVM

4.2.1 Analysis Results: x86 dnsmasq

4.2.1.1 stripped

function	esp	reads	writes	unrc	blocks	instrs	time
0x6cd0	100.0	100.0	100.0		1	2	0.0117
0x6cc5	100.0	100.0	100.0		1	2	0.0125
0xe128	100.0	100.0	100.0		1	2	0.0129
0xa034	100.0	100.0	100.0		1	2	0.0132
0xa03c	100.0	100.0	100.0		1	2	0.0134
0x6dc9	100.0	100.0	100.0		1	2	0.0141
0x1e547	100.0	100.0	100.0		1	2	0.0144
0xa038	100.0	100.0	100.0		1	2	0.0146
0xeaf0	100.0	100.0	100.0		1	6	0.0169
0x29b60	100.0	100.0	100.0		1	7	0.0171
0x3a200	100.0	100.0	100.0		1	4	0.0213
0xea10	100.0	100.0	100.0		1	6	0.022
0xf880	100.0	33.33	50.0		1	11	0.0224
0x33f70	100.0	50.0	0.0		3	8	0.0236
0xea90	100.0	100.0	100.0		1	9	0.025
0x22650	100.0	100.0	100.0		3	8	0.0261
0x233d0	100.0	33.33	0.0		4	11	0.0262
0x1f1b0	100.0	100.0	100.0		1	11	0.0265
0x40d3	100.0	0.0	100.0		4	10	0.0268
0x7450	100.0	100.0	100.0		1	15	0.0276
0x77e0	100.0	100.0	100.0		4	9	0.0279
0x22720	100.0	100.0	100.0		3	7	0.0288
0xf7b0	100.0	100.0	100.0		3	12	0.0291
0x7970	100.0	100.0	100.0		4	10	0.0293
0x2c30	30.0	100.0	60.0		6	10	0.0303
0x2c90	30.0	100.0	60.0		6	10	0.0305
0x3170	30.0	100.0	60.0		6	10	0.0306
0x2f50	30.0	100.0	60.0		6	10	0.0307
0x2fc0	30.0	100.0	60.0		6	10	0.0308
0x2d20	30.0	100.0	60.0		6	10	0.0309
0x3100	30.0	100.0	60.0		6	10	0.031
0x2a50	0.0	100.0	50.0		4	7	0.0311
0x3010	30.0	100.0	60.0		6	10	0.0312
0x2b80	30.0	100.0	60.0		6	10	0.0313
0x3140	30.0	100.0	60.0		6	10	0.0314
0x2d30	30.0	100.0	60.0		6	10	0.0315
0x2dd0	30.0	100.0	60.0		6	10	0.0315
0x2fe0	30.0	100.0	60.0		6	10	0.0315
0x3c010	100.0	100.0	100.0		1	12	0.0315
0x2bb0	30.0	100.0	60.0		6	10	0.0316
0x2ff0	30.0	100.0	60.0		6	10	0.0316
0x30c0	30.0	100.0	60.0		6	10	0.0316

0x32a0	30.0	100.0	60.0	6	10	0.0316
0x2b30	30.0	100.0	60.0	6	10	0.0317
0x2ae0	30.0	100.0	60.0	6	10	0.0318
0x2da0	30.0	100.0	60.0	6	10	0.0318
0x3260	30.0	100.0	60.0	6	10	0.0318
0x2d00	30.0	100.0	60.0	6	10	0.0319
0x3110	30.0	100.0	60.0	6	10	0.0319
0x2ca0	30.0	100.0	60.0	6	10	0.0321
0x3120	30.0	100.0	60.0	6	10	0.0321
0x31b0	30.0	100.0	60.0	6	10	0.0321
0x31a0	30.0	100.0	60.0	6	10	0.0322
0x31f0	30.0	100.0	60.0	6	10	0.0322
0x3200	30.0	100.0	60.0	6	10	0.0322
0x2ad0	30.0	100.0	60.0	6	10	0.0323
0x2f20	30.0	100.0	60.0	6	10	0.0324
0x3080	30.0	100.0	60.0	6	10	0.0324
0x2b40	30.0	100.0	60.0	6	10	0.0326
0x2bf0	30.0	100.0	60.0	6	10	0.0326
0x2ef0	30.0	100.0	60.0	6	10	0.0326
0x2fb0	30.0	100.0	60.0	6	10	0.0327
0x2b70	30.0	100.0	60.0	6	10	0.0328
0x2d80	30.0	100.0	60.0	6	10	0.0329
0x2ee0	30.0	100.0	60.0	6	10	0.0329
0x2af0	30.0	100.0	60.0	6	10	0.033
0x2cb0	30.0	100.0	60.0	6	10	0.033
0x3280	30.0	100.0	60.0	6	10	0.033
0x2aa0	30.0	100.0	60.0	6	10	0.0331
0x2e10	30.0	100.0	60.0	6	10	0.0332
0x2e90	30.0	100.0	60.0	6	10	0.0332
0x2e60	30.0	100.0	60.0	6	10	0.0333
0x2fd0	30.0	100.0	60.0	6	10	0.0333
0x2a60	30.0	100.0	60.0	6	10	0.0334
0x2db0	30.0	100.0	60.0	6	10	0.0334
0x31d0	30.0	100.0	60.0	6	10	0.0334
0x3270	30.0	100.0	60.0	6	10	0.0334
0x2d50	30.0	100.0	60.0	6	10	0.0335
0x2df0	30.0	100.0	60.0	6	10	0.0335
0x2e30	30.0	100.0	60.0	6	10	0.0336
0x32d0	30.0	100.0	60.0	6	10	0.0336
0x2e50	30.0	100.0	60.0	6	10	0.0337
0x2cf0	30.0	100.0	60.0	6	10	0.0338
0x32e0	30.0	100.0	60.0	6	10	0.0339
0x2a90	30.0	100.0	60.0	6	10	0.034
0x2f40	30.0	100.0	60.0	6	10	0.034
0x3070	30.0	100.0	60.0	6	10	0.034
0x31e0	30.0	100.0	60.0	6	10	0.034
0x2c00	30.0	100.0	60.0	6	10	0.0341
0x3060	30.0	100.0	60.0	6	10	0.0341
0x2bc0	30.0	100.0	60.0	6	10	0.0342
0x3090	30.0	100.0	60.0	6	10	0.0342
0x30f0	30.0	100.0	60.0	6	10	0.0342

0x3250	30.0	100.0	60.0	6	10	0.0342
0x2c70	30.0	100.0	60.0	6	10	0.0343
0x3130	30.0	100.0	60.0	6	10	0.0343
0x2bd0	30.0	100.0	60.0	6	10	0.0344
0x2cc0	30.0	100.0	60.0	6	10	0.0344
0x2c20	30.0	100.0	60.0	6	10	0.0345
0x30a0	30.0	100.0	60.0	6	10	0.0345
0x3050	30.0	100.0	60.0	6	10	0.0346
0x2e70	30.0	100.0	60.0	6	10	0.0347
0x22370	100.0	33.33	0.0	4	11	0.0348
0x2a10	100.0	100.0	100.0	3	11	0.0348
0x2c80	30.0	100.0	60.0	6	10	0.0348
0x2f90	30.0	100.0	60.0	6	10	0.0348
0x2cd0	30.0	100.0	60.0	6	10	0.0349
0x3300	30.0	100.0	60.0	6	10	0.0349
0x2b00	30.0	100.0	60.0	6	10	0.035
0x2de0	30.0	100.0	60.0	6	10	0.035
0x2e80	30.0	100.0	60.0	6	10	0.035
0x3150	30.0	100.0	60.0	6	10	0.035
0x2ec0	30.0	100.0	60.0	6	10	0.0351
0x3220	30.0	100.0	60.0	6	10	0.0351
0x3290	30.0	100.0	60.0	6	10	0.0351
0x3040	30.0	100.0	60.0	6	10	0.0352
0x31c0	30.0	100.0	60.0	6	10	0.0352
0x2c60	30.0	100.0	60.0	6	10	0.0354
0x2f10	30.0	100.0	60.0	6	10	0.0354
0x2b10	30.0	100.0	60.0	6	10	0.0355
0x3190	30.0	100.0	60.0	6	10	0.0355
0x340f0	100.0	100.0	100.0	3	10	0.0355
0x2b20	30.0	100.0	60.0	6	10	0.0356
0x2d40	30.0	100.0	60.0	6	10	0.0356
0x30d0	30.0	100.0	60.0	6	10	0.0356
0x3000	30.0	100.0	60.0	6	10	0.0357
0x3180	30.0	100.0	60.0	6	10	0.0357
0x2b90	30.0	100.0	60.0	6	10	0.0358
0x2c40	30.0	100.0	60.0	6	10	0.0359
0x2fa0	30.0	100.0	60.0	6	10	0.0359
0x2b50	30.0	100.0	60.0	6	10	0.0361
0x2f00	30.0	100.0	60.0	6	10	0.0361
0x32f0	30.0	100.0	60.0	6	10	0.0361
0x22280	100.0	40.0	100.0	6	16	0.0363
0x2ab0	30.0	100.0	60.0	6	10	0.0363
0x2ce0	30.0	100.0	60.0	6	10	0.0363
0x2f80	30.0	100.0	60.0	6	10	0.0363
0x2a70	30.0	100.0	60.0	6	10	0.0365
0x2dc0	30.0	100.0	60.0	6	10	0.0365
0x6ce0	100.0	0.0	100.0	4	18	0.0365
0x2a80	30.0	100.0	60.0	6	10	0.0366
0x2d60	30.0	100.0	60.0	6	10	0.0366
0x3030	30.0	100.0	60.0	6	10	0.0366
0x32b0	30.0	100.0	60.0	6	10	0.0366

0x2d10	30.0	100.0	60.0	6	10	0.0367
0x2d90	30.0	100.0	60.0	6	10	0.0367
0x2f60	30.0	100.0	60.0	6	10	0.0367
0x2b60	30.0	100.0	60.0	6	10	0.0368
0x2eb0	30.0	100.0	60.0	6	10	0.0368
0x3230	30.0	100.0	60.0	6	10	0.0368
0x3240	30.0	100.0	60.0	6	10	0.0368
0x2c50	30.0	100.0	60.0	6	10	0.0369
0x2ed0	30.0	100.0	60.0	6	10	0.037
0x30e0	30.0	100.0	60.0	6	10	0.037
0x2ba0	30.0	100.0	60.0	6	10	0.0372
0x30b0	30.0	100.0	60.0	6	10	0.0374
0x2e00	30.0	100.0	60.0	6	10	0.0375
0x2ea0	30.0	100.0	60.0	6	10	0.0375
0x3210	30.0	100.0	60.0	6	10	0.038
0x2e20	30.0	100.0	60.0	6	10	0.0385
0x20a30	100.0	40.0	100.0	8	15	0.0386
0x2d70	30.0	100.0	60.0	6	10	0.0386
0x2be0	30.0	100.0	60.0	6	10	0.0387
0x2f30	30.0	100.0	60.0	6	10	0.0388
0x32c0	30.0	100.0	60.0	6	10	0.0394
0x2ac0	30.0	100.0	60.0	6	10	0.0396
0x3160	30.0	100.0	60.0	6	10	0.0398
0x3020	30.0	100.0	60.0	6	10	0.0405
0x340d0	100.0	100.0	100.0	3	13	0.0406
0x2c10	30.0	100.0	60.0	6	10	0.0408
0x3a220	100.0	100.0	100.0	1	11	0.041
0x34110	100.0	100.0	100.0	1	17	0.0412
0x40e9	100.0	100.0	100.0	1	12	0.0425
0x7420	100.0	100.0	100.0	6	15	0.0425
0x2e40	30.0	100.0	60.0	6	10	0.0431
0xeab0	100.0	100.0	100.0	7	18	0.0477
0xfda0	100.0	33.33	33.33	4	24	0.048
0x360c	100.0	100.0	66.67	4	19	0.0489
0x33ee0	100.0	100.0	0.0	1	15	0.0499
0x329c0	100.0	100.0	100.0	4	21	0.0518
0xebc0	100.0	100.0	50.0	3	16	0.0522
0x79a0	100.0	100.0	100.0	7	17	0.053
0x340a0	100.0	100.0	100.0	3	19	0.0547
0x33f20	100.0	100.0	100.0	3	19	0.0551
0x1c550	100.0	100.0	100.0	3	28	0.0583
0x2f70	30.0	100.0	60.0	6	10	0.0584
0x3b540	100.0	100.0	100.0	1	13	0.0599
0x3b86	100.0	100.0	100.0	1	23	0.0602
0xf700	100.0	50.0	50.0	6	19	0.0605
0x34010	100.0	100.0	100.0	3	21	0.062
0x36710	100.0	100.0	100.0	5	33	0.0622
0x28730	100.0	83.33	85.71	6	34	0.0633
0x2bdd0	100.0	100.0	100.0	5	24	0.0634
0x22670	100.0	100.0	100.0	3	20	0.0643
0x21d60	100.0	100.0	100.0	3	18	0.0646

0x33490	100.0	57.14	100.0	6	27	0.0646
0x1d940	100.0	66.67	50.0	4	22	0.0664
0xe5c0	100.0	85.71	100.0	8	24	0.0671
0x1cfa0	100.0	50.0	100.0	9	25	0.0681
0xe550	100.0	85.71	100.0	8	24	0.0681
0x1fe10	100.0	100.0	100.0	3	26	0.0685
0x7790	100.0	71.43	100.0	6	27	0.0691
0x23ac0	100.0	100.0	100.0	5	23	0.0694
0x21690	100.0	100.0	100.0	4	26	0.0705
0xe7d0	100.0	100.0	100.0	3	25	0.0717
0x29c20	100.0	87.5	100.0	5	30	0.0721
0x2ec70	100.0	50.0	85.71	6	28	0.0724
0x19a00	100.0	100.0	100.0	4	22	0.0727
0x29d80	100.0	100.0	100.0	1	24	0.0729
0x39630	100.0	50.0	100.0	5	22	0.0732
0x18790	100.0	100.0	100.0	3	25	0.0741
0x23390	100.0	100.0	100.0	3	19	0.0741
0x28820	100.0	100.0	100.0	5	28	0.0745
0x21ea0	100.0	100.0	100.0	4	23	0.0747
0x3a1a0	100.0	42.86	100.0	9	30	0.0747
0x2da50	100.0	50.0	85.71	6	28	0.0781
0x3a250	100.0	83.33	100.0	4	27	0.0794
0x35710	100.0	62.5	60.0	6	31	0.081
0xe8a0	100.0	100.0	100.0	3	29	0.0827
0xf910	100.0	100.0	100.0	4	35	0.0839
0x34050	100.0	100.0	100.0	4	30	0.084
0x7a80	100.0	75.0	100.0	4	20	0.0843
0xf6a0	100.0	100.0	100.0	5	35	0.0861
0x4109	100.0	75.0	100.0	4	31	0.0885
0x226b0	100.0	100.0	100.0	5	32	0.092
0x6e50	100.0	63.64	66.67	15	44	0.092
0x36a4	100.0	100.0	100.0	5	37	0.0922
0x29c70	100.0	75.0	66.67	6	31	0.0927
0x21490	100.0	100.0	100.0	4	31	0.0931
0x395d0	100.0	54.55	100.0	8	32	0.0933
0x3ef2	100.0	83.33	100.0	12	33	0.0942
0x4138	100.0	80.0	100.0	4	37	0.0944
0x34140	100.0	80.0	100.0	6	33	0.0963
0x6ee0	100.0	50.0	60.0	8	39	0.0967
0xea30	100.0	80.0	100.0	5	32	0.1009
0x33390	100.0	57.14	100.0	6	41	0.1011
0xaa90	100.0	100.0	100.0	5	27	0.1034
0xe630	100.0	80.0	100.0	8	35	0.1037
0x359e0	100.0	70.0	71.43	9	39	0.1042
0xeb80	100.0	80.0	100.0	3	26	0.1058
0xf130	100.0	83.33	100.0	8	32	0.1061
0x1b360	100.0	81.82	100.0	12	36	0.1067
0x1f110	100.0	100.0	100.0	8	49	0.1085
0xaeb0	100.0	100.0	100.0	11	49	0.1092
0xeb10	100.0	80.0	100.0	4	39	0.1092
0xf4f0	100.0	77.78	100.0	9	31	0.1092

0x1cf40	100.0	55.56	100.0	6	31	0.1108
0xe830	100.0	100.0	100.0	7	38	0.1129
0x33fa0	100.0	100.0	100.0	4	31	0.1131
0xf740	100.0	87.5	85.71	8	42	0.115
0x2f3b0	100.0	100.0	100.0	8	51	0.1178
0x6dd0	100.0	75.0	100.0	6	40	0.12
0xa040	100.0	100.0	100.0	10	42	0.121
0x2ecc0	100.0	92.86	100.0	3	43	0.1211
0x1c490	100.0	40.0	100.0	10	28	0.1214
0x29dd0	100.0	100.0	100.0	4	35	0.1219
0x79c0	100.0	62.5	30.0	13	52	0.1224
0x215f0	100.0	100.0	92.31	5	44	0.1227
0x23860	100.0	84.62	62.5	8	45	0.1228
0x21da0	100.0	87.5	100.0	5	28	0.1235
0xe6b0	100.0	75.0	100.0	13	39	0.1239
0x33310	100.0	57.14	100.0	6	41	0.1248
0xf640	100.0	50.0	83.33	10	37	0.1267
0x225e0	100.0	76.92	100.0	9	43	0.1276
0x27e00	100.0	100.0	100.0	10	49	0.128
0xf5d0	100.0	100.0	100.0	5	47	0.1291
0x32570	100.0	100.0	100.0	3	41	0.1293
0x363f	100.0	54.55	88.89	8	38	0.1306
0x2b490	100.0	100.0	100.0	3	29	0.1307
0x29b80	100.0	100.0	100.0	9	49	0.1353
0xf180	100.0	100.0	100.0	7	50	0.1377
0x326c0	100.0	45.45	66.67	9	39	0.1399
0x2dbf0	100.0	100.0	100.0	5	49	0.1405
0x2e230	100.0	69.23	100.0	8	45	0.1409
0x3f3a	100.0	75.0	57.14	18	48	0.1431
0x7800	100.0	54.55	100.0	10	30	0.1443
0x1d990	100.0	60.0	87.5	8	48	0.145
0xe900	100.0	100.0	100.0	8	56	0.1453
0x33410	100.0	80.0	100.0	7	35	0.1455
0x1ced0	100.0	55.56	100.0	8	36	0.1469
0x36760	100.0	93.33	100.0	10	54	0.1512
0x342d0	100.0	100.0	100.0	7	64	0.1524
0x187f0	100.0	100.0	100.0	7	44	0.1567
0x20890	100.0	53.85	100.0	14	49	0.1603
0x2b4f0	100.0	100.0	100.0	11	63	0.1618
0x36690	100.0	40.0	100.0	7	48	0.1674
0x2d5e0	100.0	80.0	100.0	7	47	0.1675
0xf450	100.0	100.0	100.0	5	52	0.168
0xf8b0	100.0	100.0	100.0	9	65	0.1734
0x2e180	100.0	100.0	100.0	21	64	0.1746
0x9750	100.0	58.33	100.0	15	42	0.1748
0xf7d0	100.0	77.78	93.33	11	65	0.1765
0x32c00	100.0	69.23	100.0	11	46	0.1775
0x28790	100.0	90.91	100.0	12	58	0.179
0xe990	100.0	100.0	100.0	12	43	0.1817
0xf970	100.0	100.0	100.0	3	53	0.1826
0x236e0	100.0	85.71	100.0	15	53	0.183

0x23b50	100.0	76.47	100.0	11	57	0.1894
0x29cd0	100.0	100.0	100.0	1	56	0.191
0x223a0	100.0	70.59	100.0	11	45	0.1913
0x2bf10	40.48	50.0	35.29	3	42	0.1941
0x1eec0	100.0	75.0	94.44	7	52	0.1943
0x2d660	100.0	100.0	100.0	9	66	0.203
0x331d0	100.0	50.0	100.0	16	63	0.203
0x22420	100.0	84.62	100.0	8	47	0.2035
0x7ad0	100.0	80.0	72.73	15	46	0.2039
0x3a7c0	100.0	100.0	100.0	9	60	0.2047
0x2db60	100.0	69.23	90.0	11	52	0.2094
0x21df0	100.0	50.0	73.33	10	53	0.2123
0x33e30	100.0	100.0	100.0	3	57	0.2132
0x8b50	100.0	100.0	100.0	5	46	0.2142
0x1f1e0	100.0	100.0	100.0	10	59	0.2174
0xf550	100.0	82.35	100.0	15	54	0.2194
0xac60	100.0	100.0	100.0	7	44	0.2201
0x8f50	100.0	92.31	93.33	10	59	0.221
0x72d0	100.0	90.0	100.0	12	59	0.2218
0x29a80	100.0	100.0	95.0	8	61	0.2222
0x2d0e0	100.0	88.24	100.0	9	54	0.2278
0x323e0	100.0	50.0	100.0	16	67	0.2358
0xae00	100.0	100.0	100.0	7	64	0.24
0xe450	100.0	71.43	90.91	19	76	0.2402
0x2ebb0	100.0	93.75	100.0	12	68	0.2513
0x3c6b	100.0	61.54	100.0	4	49	0.259
0x28670	100.0	85.71	100.0	10	63	0.2605
0x3bda0	100.0	80.95	93.75	12	71	0.261
0x1e430	100.0	100.0	100.0	19	76	0.2658
0x3508	100.0	54.17	66.67	25	89	0.266
0x3bbe	100.0	66.67	100.0	12	65	0.2702
0x1b2d0	100.0	100.0	100.0	9	46	0.2728
0x222c0	100.0	68.97	100.0	13	61	0.2755
0xf210	100.0	95.0	100.0	6	61	0.2772
0x1ce00	100.0	75.0	94.44	10	69	0.284
0x20920	100.0	52.38	95.65	17	78	0.2846
0x21380	100.0	100.0	100.0	10	86	0.2889
0x1c960	100.0	100.0	100.0	12	83	0.2925
0x1ed90	100.0	83.33	100.0	4	97	0.297
0x9790	100.0	88.24	100.0	17	85	0.3027
0x7090	100.0	64.71	100.0	14	73	0.3119
0x32b20	100.0	100.0	100.0	11	69	0.3151
0x8be0	100.0	66.67	62.5	11	47	0.3182
0x38fe0	100.0	80.0	100.0	16	63	0.3193
0x29970	100.0	100.0	96.15	8	76	0.323
0x3441	100.0	90.91	94.44	7	65	0.3245
0x33280	100.0	76.92	100.0	12	84	0.3256
0x6f80	100.0	86.67	90.91	18	86	0.3256
0x198f0	100.0	75.0	58.33	13	84	0.336
0x23a20	100.0	93.33	100.0	23	112	0.3416
0x3be80	100.0	100.0	100.0	18	110	0.3452

0x1cd30	100.0	63.64	88.89	17	70	0.3457
0x397d0	100.0	73.68	90.0	16	68	0.3463
0x3320	100.0	100.0	96.77	12	99	0.3475
0x7180	100.0	68.18	63.16	21	90	0.3493
0x2dc70	100.0	100.0	96.88	6	84	0.3557
0x3aca0	100.0	52.94	77.78	25	135	0.3559
0x3310	100.0	100.0	96.77	16	103	0.3584
0x2be70	69.88	72.22	56.0	12	83	0.3619
0x32990	100.0	70.83	100.0	19	79	0.3632
0x324b0	100.0	66.67	100.0	15	69	0.3644
0x232b0	100.0	50.0	100.0	12	75	0.3665
0x2daa0	100.0	19.05	100.0	20	69	0.3773
0x238e0	100.0	96.55	100.0	16	100	0.3776
0x3318	100.0	100.0	96.77	14	101	0.3827
0x221a0	100.0	59.09	100.0	21	79	0.3908
0xa6a0	100.0	89.47	100.0	15	78	0.3936
0x20a60	100.0	76.0	80.95	17	91	0.3989
0x224b0	100.0	89.47	100.0	19	89	0.4009
0x32600	100.0	100.0	100.0	13	71	0.4036
0xed20	100.0	100.0	100.0	11	111	0.4082
0x22070	100.0	86.36	90.91	14	91	0.4098
0x7380	100.0	90.48	88.89	20	112	0.4156
0x3a2a0	100.0	92.86	89.29	14	109	0.4294
0x23400	100.0	72.97	88.46	23	118	0.4331
0xe720	100.0	78.95	71.43	19	65	0.4559
0x23770	100.0	75.0	86.67	15	74	0.4882
0x23580	100.0	96.88	95.24	25	111	0.4946
0x1c3a0	100.0	55.88	100.0	29	105	0.4951
0x2b5e0	61.02	75.0	85.19	23	118	0.5108
0x22740	100.0	100.0	100.0	22	126	0.55
0x3a39	71.79	36.36	60.53	24	117	0.5554
0x2dda0	100.0	100.0	97.14	10	102	0.5639
0x19a50	100.0	74.07	94.29	37	182	0.5722
0xbe80	100.0	95.0	50.0	25	134	0.58
0x3ceb	100.0	75.0	87.1	40	172	0.583
0xbae0	100.0	62.96	100.0	33	107	0.5845
0x2d4c0	100.0	78.26	100.0	17	81	0.6052
0x3a410	100.0	78.26	75.0	28	122	0.6107
0x1f7c0	100.0	77.27	83.87	22	136	0.612
0x3fb0	100.0	69.23	70.59	22	106	0.6319
0x1fbe0	100.0	100.0	100.0	25	188	0.646
0xbdb0	100.0	92.0	100.0	18	80	0.646
0xaad0	100.0	92.31	100.0	19	137	0.6788
0x32270	100.0	65.71	100.0	24	116	0.6883
0x39400	100.0	70.0	91.18	30	157	0.6983
0x1ef50	76.34	75.76	89.47	37	186	0.7485
0xebf0	100.0	96.43	91.67	24	212	0.7619
0x1d540	100.0	84.78	97.62	32	177	0.7913
0x39680	100.0	70.83	100.0	12	96	0.7964
0x35770	100.0	76.0	97.1	32	197	0.8112
0x36f7	23.89	28.81	33.33	39	226	0.8166

0x36580	100.0	54.84	94.44	16	81	0.8499
0x2d860	100.0	74.36	81.08	36	168	0.8524
0x1d770	100.0	58.14	85.71	23	150	0.8578
0x16d10	100.0	86.05	100.0	28	170	0.8656
0x98b0	100.0	75.47	100.0	41	214	0.9342
0x1b740	100.0	61.76	91.3	56	194	0.9677
0x32730	100.0	34.04	61.11	54	179	0.9686
0xf390	100.0	100.0	100.0	13	65	0.9944
0x27c80	100.0	96.3	100.0	29	149	1.0315
0x21f00	100.0	50.0	100.0	31	121	1.0347
0x1cc30	100.0	80.0	88.89	27	146	1.0487
0xba00	100.0	73.68	100.0	50	173	1.0536
0x32a00	100.0	76.0	100.0	14	92	1.1126
0x16f90	100.0	88.24	100.0	33	200	1.1173
0x7860	100.0	93.55	80.65	31	167	1.1869
0x2bb50	100.0	72.22	87.5	36	199	1.1922
0x29e40	100.0	98.0	84.0	34	185	1.1946
0x228f0	100.0	86.67	100.0	44	200	1.1986
0x2cfb0	100.0	94.74	100.0	16	138	1.202
0xfa10	100.0	92.16	96.72	44	259	1.2184
0x17390	100.0	32.79	88.89	44	216	1.2488
0xf2d0	100.0	100.0	100.0	22	114	1.2506
0x1da20	100.0	86.79	49.25	32	217	1.2595
0x32c80	100.0	71.88	100.0	59	226	1.3597
0xa0c0	100.0	86.36	88.89	32	160	1.3743
0x3ae60	100.0	100.0	100.0	32	178	1.3974
0x1f970	100.0	49.25	85.71	34	185	1.4055
0xfde0	100.0	73.58	53.42	59	247	1.5548
0x1b6b0	100.0	52.0	88.24	62	237	1.556
0x183c0	100.0	75.76	92.16	63	256	1.558
0x8100	100.0	79.69	94.55	42	246	1.5618
0x1c650	100.0	100.0	98.88	46	305	1.6619
0x2be20	100.0	75.0	88.37	41	222	1.6986
0x296b0	100.0	100.0	100.0	23	190	1.8112
0x1d460	100.0	87.3	98.44	40	250	1.8765
0x3bb10	100.0	71.7	62.5	50	187	1.9502
0x3a570	100.0	78.57	100.0	36	173	2.0359
0x2d720	100.0	82.46	88.14	47	252	2.1314
0xbc30	100.0	81.82	100.0	38	195	2.1437
0x8c70	100.0	81.58	80.25	51	304	2.276
0x3b670	100.0	77.78	86.41	42	324	2.381
0xee70	100.0	89.47	88.1	47	199	2.3837
0x1dd30	100.0	67.61	88.46	70	321	2.5138
0x1e140	100.0	96.55	95.59	46	279	2.5201
0x38c20	100.0	90.74	87.78	43	333	2.5807
0x9c20	100.0	100.0	98.08	72	271	2.6581
0xe130	100.0	76.4	93.55	32	287	2.6824
0xa2e0	100.0	78.87	90.32	65	342	2.7271
0x33a20	100.0	98.85	96.26	42	323	2.781
0x1ca80	100.0	75.38	94.12	56	281	2.8306
0x7480	100.0	57.38	50.0	65	239	2.8519

0xacd0	100.0	94.12	100.0	24	160	3.0192
0x1f590	100.0	78.0	92.06	46	282	3.0636
0x2dde0	100.0	54.29	98.61	32	280	3.1819
0x1c5a0	100.0	100.0	99.07	54	360	3.1828
0x39090	100.0	62.07	77.42	70	263	3.3001
0xa790	100.0	69.12	89.13	39	211	3.3249
0x1f4f0	100.0	79.25	93.51	53	333	3.5779
0x1b3e0	100.0	52.42	78.67	112	459	3.7328
0x8770	100.0	30.95	90.91	59	288	3.779
0x2b0a0	100.0	84.21	84.81	39	295	3.8141
0x1c500	100.0	53.17	78.95	117	473	3.9361
0x216f0	100.0	48.76	100.0	80	500	3.9726
0x32f40	100.0	70.0	86.21	33	186	4.0753
0x3b0d0	100.0	86.08	90.24	42	321	4.2172
0x1d2b0	100.0	78.65	97.85	59	365	4.271
0x3b570	100.0	70.09	86.55	55	405	4.3917
0x2b7e0	100.0	84.42	90.57	75	431	4.5278
0x3a860	100.0	83.75	92.65	39	293	4.7561
0x38a80	100.0	87.93	91.54	62	462	4.7771
0x36310	100.0	42.03	88.46	53	265	4.8708
0x35d10	100.0	61.4	98.48	34	232	5.0773
0x7e90	100.0	71.96	83.15	86	412	5.1761
0x398c0	100.0	70.83	93.65	28	214	5.5717
0x2ed50	100.0	75.38	94.56	80	546	5.7276
0x1e750	15.26	6.84	15.27	102	557	5.9456
0x31f10	100.0	79.63	94.12	47	357	5.9629
0x1f2b0	100.0	74.73	91.3	79	485	6.4939
0x20e00	100.0	63.06	86.96	96	479	6.6944
0x2c9d0	100.0	75.4	94.3	91	515	7.3909
0x22b80	100.0	91.1	97.8	69	543	7.4249
0x9290	100.0	60.87	95.65	49	328	7.8854
0x1a930	100.0	58.21	96.15	118	649	7.9893
0x334f0	100.0	57.79	88.2	124	702	8.0449
0x31db0	100.0	77.21	94.4	57	441	8.2301
0x1fe70	100.0	85.14	98.54	104	641	8.7212
0x1e550	32.76	21.05	28.39	137	702	9.3818
0x39e70	100.0	52.46	100.0	45	221	9.6178
0x7b80	100.0	65.79	74.05	142	611	9.9468
0x1d000	100.0	78.99	97.28	90	557	10.2212
0x2e2a0	100.0	54.55	95.88	138	664	10.2654
0x2c7e0	100.0	78.01	95.22	113	663	12.5983
0x20b80	5.29	12.57	11.72	140	662	12.8123
0x35a00	100.0	70.91	99.12	80	456	13.0801
0x36030	100.0	60.29	89.8	96	463	13.7801
0x1a430	100.0	57.19	95.73	190	1019	14.9014
0x4197	32.36	21.89	51.46	250	1057	14.9903
0x16760	100.0	74.23	82.67	89	410	15.7919
0x8330	100.0	58.17	96.03	96	574	16.3593
0x2d180	100.0	79.1	100.0	30	251	17.6751
0x39b90	100.0	54.72	97.96	80	418	21.8512
0x17300	100.0	84.55	87.21	92	455	23.3512

```
0x34520      100.0    63.8   84.64            219    1162   27.6242
0x2f440      100.0   44.51   86.26            317    1939   34.1508
0x16ff0      100.0   66.24   87.94            128     679   35.3681
0x2a0e0      100.0   58.31   95.44            177    1107   36.6407
0x19cd0      100.0   61.99   94.35            279    1497   45.9248
0x36800      100.0   58.75   95.35            417    2429   47.6399
0xaf60       100.0   77.42   93.83            167     866   64.4584
0x1b9c0      100.0   48.34   79.17            107     588   65.5314
0x23bf0      83.92   54.76   71.47            483    2804   88.1264
0x17670      60.96   24.73   50.78            230    1150   145.2208
0x28880       1.88    6.04   11.31            173    1171   168.9595
0x28040      100.0   55.77   100.0            106     480   237.4184
0x18f60      100.0   60.69   93.46             98     655   283.497
0x18890      100.0   58.63   93.41            170    1112   305.8778
0x18d90      100.0   61.34   94.59            111     765   307.18
0x27ea0      100.0    62.3   100.0            121     584   330.4836
0xc420       100.0   62.68   96.98            349    1905   832.5844
0xc2c0       100.0   63.49    95.1            364    2010   836.1099
--------------------------------------------------------------------------------
Disassembly Summary
--------------------------------------------------------------------------------
Instruction count:   64922
Unknown instrs   :       0
Function count   :     519
Function coverage:    81.1%
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Analysis Summary
--------------------------------------------------------------------------------
Esp precision   :   92.11%
Reads precision :   66.75%
Writes precision:   85.86%
Calls         :    4252
Analysis time   :  4722.08 secs
Iterations     :       8
--------------------------------------------------------------------------------
```

4.2.1.2 not stripped

```
function     esp    reads   writes  unrc  blocks instrs   time
--------------------------------------------------------------------------------
0x3c000      100.0   100.0   100.0          1    1   0.0113 (__libc_csu_fini)
0x1e547      100.0   100.0   100.0          1    2   0.0123 (__x86.get_pc_thunk.bp)
0xa03c       100.0   100.0   100.0          1    2   0.0124 (__x86.get_pc_thunk.di)
0xe128       100.0   100.0   100.0          1    2   0.0124 (__x86.get_pc_thunk.ax)
0x6dc9       100.0   100.0   100.0          1    2   0.0126 (__x86.get_pc_thunk.dx)
0xa034       100.0   100.0   100.0          1    2   0.0126 (__x86.get_pc_thunk.cx)
0x6cc5       100.0   100.0   100.0          1    2    0.013
0x3a200      100.0   100.0   100.0          1    4    0.015 (poll_reset)
```

```
0x6cd0      100.0   100.0   100.0          1    2   0.0159 (__x86.get_pc_thunk.bx)
0xeaf0      100.0   100.0   100.0          1    6   0.0166 (is_same_net)
0xa038      100.0   100.0   100.0          1    2   0.0168 (__x86.get_pc_thunk.si)
0x3c024     100.0   100.0   100.0          1    7   0.0178 (_fini)
0x29b60     100.0   100.0   100.0          1    7    0.018 (helper_buf_empty)
0xea90      100.0   100.0   100.0          1    9   0.0201 (dnsmasq_time)
0x33f70     100.0    50.0     0.0          3    8   0.0204 (save_counter)
0xea10      100.0   100.0   100.0          1    6   0.0207 (sa_len)
0xf880      100.0   33.33    50.0          1   11   0.0253 (set_option_bool.part.5)
0x1f1b0     100.0   100.0   100.0          1   11   0.0263 (queue_event)
0x22650     100.0   100.0   100.0          3    8   0.0264 (lease4_allocate)
0xf7b0      100.0   100.0   100.0          3   12   0.0268 (atoi_check16)
0x22370     100.0   33.33     0.0          4   11   0.0273 (lease6_reset)
0x40d3      100.0     0.0   100.0          4   10   0.0276 (in_list.part.3)
0x22720     100.0   100.0   100.0          3    7   0.0279 (lease_set_iaid)
0x233d0     100.0   33.33     0.0          4   11   0.0289 (rerun_scripts)
0x3c010     100.0   100.0   100.0          2   12   0.0289 (__stack_chk_fail_local)
0x2a50        0.0   100.0    50.0          4    7   0.0302
0x7970      100.0   100.0   100.0          4   10   0.0302 (cache_get_name)
0x30b0       30.0   100.0    60.0          6   10   0.0308
0x2fe0       30.0   100.0    60.0          6   10   0.0311
0x2d90       30.0   100.0    60.0          6   10   0.0313
0x32a0       30.0   100.0    60.0          6   10   0.0313
0x77e0      100.0   100.0   100.0          4    9   0.0313 (cache_get_cname_target
                                                        .part.5)
0x2ba0       30.0   100.0    60.0          6   10   0.0314
0x2eb0       30.0   100.0    60.0          6   10   0.0317
0x3010       30.0   100.0    60.0          6   10   0.0318
0x2e70       30.0   100.0    60.0          6   10    0.032
0x2da0       30.0   100.0    60.0          6   10   0.0321
0x3250       30.0   100.0    60.0          6   10   0.0321
0x3280       30.0   100.0    60.0          6   10   0.0321
0x2d50       30.0   100.0    60.0          6   10   0.0322
0x2ec0       30.0   100.0    60.0          6   10   0.0322
0x3140       30.0   100.0    60.0          6   10   0.0322
0x7450      100.0   100.0   100.0          1   15   0.0322 (is_expired.isra.2.
                                                       part.3)
0x2ef0       30.0   100.0    60.0          6   10   0.0323
0x2fc0       30.0   100.0    60.0          6   10   0.0323
0x3240       30.0   100.0    60.0          6   10   0.0323
0x2c30       30.0   100.0    60.0          6   10   0.0324
0x3090       30.0   100.0    60.0          6   10   0.0324
0x2fb0       30.0   100.0    60.0          6   10   0.0325
0x3150       30.0   100.0    60.0          6   10   0.0325
0x2b10       30.0   100.0    60.0          6   10   0.0327
0x2d30       30.0   100.0    60.0          6   10   0.0328
0x3170       30.0   100.0    60.0          6   10   0.0328
0x3260       30.0   100.0    60.0          6   10   0.0328
0x3270       30.0   100.0    60.0          6   10   0.0329
0x2c40       30.0   100.0    60.0          6   10    0.033
0x2c90       30.0   100.0    60.0          6   10    0.033
```

```
0x2b30       30.0   100.0    60.0          6   10   0.0331
0x31d0       30.0   100.0    60.0          6   10   0.0331
0x3300       30.0   100.0    60.0          6   10   0.0332
0x2a90       30.0   100.0    60.0          6   10   0.0333
0x2c10       30.0   100.0    60.0          6   10   0.0333
0x2f90       30.0   100.0    60.0          6   10   0.0333
0x2f40       30.0   100.0    60.0          6   10   0.0334
0x3100       30.0   100.0    60.0          6   10   0.0334
0x32e0       30.0   100.0    60.0          6   10   0.0334
0x2d20       30.0   100.0    60.0          6   10   0.0335
0x2db0       30.0   100.0    60.0          6   10   0.0335
0x2c20       30.0   100.0    60.0          6   10   0.0336
0x2f10       30.0   100.0    60.0          6   10   0.0336
0x3070       30.0   100.0    60.0          6   10   0.0336
0x3220       30.0   100.0    60.0          6   10   0.0336
0x340f0     100.0   100.0   100.0          3   10   0.0336 (put_opt6_char)
0x2ad0       30.0   100.0    60.0          6   10   0.0337
0x2b50       30.0   100.0    60.0          6   10   0.0337
0x2b80       30.0   100.0    60.0          6   10   0.0337
0x2d40       30.0   100.0    60.0          6   10   0.0337
0x2f60       30.0   100.0    60.0          6   10   0.0337
0x2ce0       30.0   100.0    60.0          6   10   0.0338
0x2d80       30.0   100.0    60.0          6   10   0.0338
0x2de0       30.0   100.0    60.0          6   10   0.0338
0x2ff0       30.0   100.0    60.0          6   10   0.0338
0x3200       30.0   100.0    60.0          6   10   0.0338
0x2a60       30.0   100.0    60.0          6   10   0.0339
0x2dd0       30.0   100.0    60.0          6   10   0.0339
0x2e30       30.0   100.0    60.0          6   10   0.0339
0x2f00       30.0   100.0    60.0          6   10   0.0339
0x3160       30.0   100.0    60.0          6   10   0.0339
0x2af0       30.0   100.0    60.0          6   10    0.034
0x2fd0       30.0   100.0    60.0          6   10    0.034
0x30c0       30.0   100.0    60.0          6   10    0.034
0x3230       30.0   100.0    60.0          6   10    0.034
0x2a10      100.0   100.0   100.0          3   11   0.0341 (_init)
0x2c70       30.0   100.0    60.0          6   10   0.0341
0x3110       30.0   100.0    60.0          6   10   0.0341
0x2b60       30.0   100.0    60.0          6   10   0.0342
0x3030       30.0   100.0    60.0          6   10   0.0342
0x2b40       30.0   100.0    60.0          6   10   0.0343
0x2e50       30.0   100.0    60.0          6   10   0.0343
0x31a0       30.0   100.0    60.0          6   10   0.0343
0x2a70       30.0   100.0    60.0          6   10   0.0344
0x2bd0       30.0   100.0    60.0          6   10   0.0344
0x31b0       30.0   100.0    60.0          6   10   0.0344
0x2b90       30.0   100.0    60.0          6   10   0.0345
0x2bb0       30.0   100.0    60.0          6   10   0.0347
0x2dc0       30.0   100.0    60.0          6   10   0.0347
0x3120       30.0   100.0    60.0          6   10   0.0347
0x2d00       30.0   100.0    60.0          6   10   0.0348
```

```
0x2e80       30.0   100.0    60.0          6   10   0.0348
0x2ea0       30.0   100.0    60.0          6   10   0.0348
0x3040       30.0   100.0    60.0          6   10   0.0348
0x2c60       30.0   100.0    60.0          6   10   0.0349
0x2ca0       30.0   100.0    60.0          6   10   0.0349
0x2fa0       30.0   100.0    60.0          6   10   0.0349
0x3020       30.0   100.0    60.0          6   10   0.0349
0x3080       30.0   100.0    60.0          6   10   0.0349
0x2cb0       30.0   100.0    60.0          6   10    0.035
0x2e10       30.0   100.0    60.0          6   10    0.035
0x31f0       30.0   100.0    60.0          6   10    0.035
0x2bf0       30.0   100.0    60.0          6   10   0.0351
0x30a0       30.0   100.0    60.0          6   10   0.0351
0x2d60       30.0   100.0    60.0          6   10   0.0352
0x2b70       30.0   100.0    60.0          6   10   0.0353
0x2cf0       30.0   100.0    60.0          6   10   0.0353
0x2d10       30.0   100.0    60.0          6   10   0.0353
0x2d70       30.0   100.0    60.0          6   10   0.0353
0x2ee0       30.0   100.0    60.0          6   10   0.0353
0x2b00       30.0   100.0    60.0          6   10   0.0354
0x2aa0       30.0   100.0    60.0          6   10   0.0355
0x3190       30.0   100.0    60.0          6   10   0.0355
0x2c00       30.0   100.0    60.0          6   10   0.0356
0x2cd0       30.0   100.0    60.0          6   10   0.0356
0x2e90       30.0   100.0    60.0          6   10   0.0357
0x2f20       30.0   100.0    60.0          6   10   0.0357
0x32b0       30.0   100.0    60.0          6   10   0.0357
0x2e40       30.0   100.0    60.0          6   10   0.0359
0x3180       30.0   100.0    60.0          6   10   0.0359
0x3050       30.0   100.0    60.0          6   10    0.036
0x6ce0      100.0     0.0   100.0          4   18    0.036 (deregister_tm_clones)
0x2c80       30.0   100.0    60.0          6   10   0.0361
0x32c0       30.0   100.0    60.0          6   10   0.0361
0x2b20       30.0   100.0    60.0          6   10   0.0362
0x3290       30.0   100.0    60.0          6   10   0.0362
0x2c50       30.0   100.0    60.0          6   10   0.0363
0x31e0       30.0   100.0    60.0          6   10   0.0363
0x2f30       30.0   100.0    60.0          6   10   0.0364
0x30f0       30.0   100.0    60.0          6   10   0.0364
0x31c0       30.0   100.0    60.0          6   10   0.0365
0x2ab0       30.0   100.0    60.0          6   10   0.0366
0x2f50       30.0   100.0    60.0          6   10   0.0366
0x2f80       30.0   100.0    60.0          6   10   0.0367
0x32d0       30.0   100.0    60.0          6   10   0.0367
0x2e00       30.0   100.0    60.0          6   10   0.0368
0x2e20       30.0   100.0    60.0          6   10   0.0368
0x2e60       30.0   100.0    60.0          6   10   0.0368
0x2ac0       30.0   100.0    60.0          6   10   0.0369
0x2cc0       30.0   100.0    60.0          6   10   0.0369
0x30d0       30.0   100.0    60.0          6   10   0.0372
0x32f0       30.0   100.0    60.0          6   10   0.0372
```

```
0x3060       30.0   100.0    60.0          6   10   0.0373
0x3210       30.0   100.0    60.0          6   10   0.0373
0x3000       30.0   100.0    60.0          6   10   0.0375
0x2ed0       30.0   100.0    60.0          6   10   0.0379
0x2ae0       30.0   100.0    60.0          6   10    0.038
0x2be0       30.0   100.0    60.0          6   10   0.0381
0x2a80       30.0   100.0    60.0          6   10   0.0382
0x2bc0       30.0   100.0    60.0          6   10   0.0383
0x30e0       30.0   100.0    60.0          6   10   0.0384
0x22280     100.0    40.0   100.0          6   16   0.0385 (lease_find_by_addr)
0x2df0       30.0   100.0    60.0          6   10   0.0392
0x16f50     100.0   100.0   100.0          3   16   0.0393 (reset_option_bool)
0x3130       30.0   100.0    60.0          6   10   0.0396
0x20a30     100.0    40.0   100.0          8   15   0.0402 (config_find_by_address)
0x340d0     100.0   100.0   100.0          3   13   0.0414 (put_opt6_short)
0x34110     100.0   100.0   100.0          1   17   0.0434 (put_opt6_string)
0x3a220     100.0   100.0   100.0          1   11   0.0438 (do_poll)
0x40e9      100.0   100.0   100.0          1   12   0.0441 (add_extradata_opt
                                                       .part.4)
0x7420      100.0   100.0   100.0          6   15   0.0442 (is_outdated_cname_pointer
                                                       .part.1)
0x33ee0     100.0   100.0     0.0          1   15   0.0458 (end_opt6)
0xeab0      100.0   100.0   100.0          7   18   0.0478 (netmask_length)
0x2f70       30.0   100.0    60.0          6   10   0.0491
0x1c550     100.0   100.0   100.0          3   28    0.05 (fix_fd)
0x360c      100.0   100.0   66.67          4   19   0.0501 (option_put)
0xebc0      100.0   100.0    50.0          3   16   0.0501 (setaddr6part)
0x21690     100.0   100.0   100.0          4   26   0.0509 (kill_name)
0xfda0      100.0   33.33   33.33          4   24   0.0519 (set_option_bool)
0x6c93      100.0   100.0   100.0          1   18   0.0526 (_start)
0x33490     100.0   57.14   100.0          6   27   0.0536 (lookup_dhcp_len)
0x340a0     100.0   100.0   100.0          3   19   0.0546 (put_opt6_long)
0x329c0     100.0   100.0   100.0          4   21   0.0548 (strip_hostname)
0x33f20     100.0   100.0   100.0          3   19   0.0548 (reset_counter)
0x79a0      100.0   100.0   100.0          7   17   0.0564 (cache_get_cname_target)
0x6d20      100.0    50.0   100.0          4   26   0.0573 (register_tm_clones)
0x6d70      100.0   100.0   100.0          5   20   0.0575 (__do_global_dtors_aux)
0x21d60     100.0   100.0   100.0          3   18   0.0579 (lease_ping_reply)
0x3b540     100.0   100.0   100.0          1   13   0.0585 (add_do_bit)
0x3b86      100.0   100.0   100.0          1   23   0.0597 (clear_packet)
0x1d940     100.0   66.67    50.0          4   22   0.0614 (mark_servers)
0x28730     100.0   83.33   85.71          6   34   0.0617 (my_setenv)
0xf700      100.0    50.0    50.0          6   19   0.0617 (unhide_metas.part.0)
0xe7d0      100.0   100.0   100.0          3   25   0.0626 (safe_malloc)
0x23ac0     100.0   100.0   100.0          5   23   0.0627 (extended_hwaddr.part.6)
0x36710     100.0   100.0   100.0          5   33   0.0641 (filter_zone)
0x22670     100.0   100.0   100.0          3   20   0.0657 (lease6_allocate)
0x34010     100.0   100.0   100.0          3   21   0.0663 (new_opt6)
0x6dc0      100.0   66.67   100.0          5   30   0.0666 (frame_dummy)
0xe550      100.0   85.71   100.0          8   24   0.0673 (rand16)
0x1cfa0     100.0    50.0   100.0          9   25   0.0684 (is_dad_listeners)
```

```
0xe5c0      100.0   85.71   100.0          8   24   0.0689 (rand32)
0x2da50     100.0    50.0   85.71          6   28   0.0697 (mark_config_used)
0x29d80     100.0   100.0   100.0          1   24   0.0698 (tftp_err_oops)
0x2bdd0     100.0   100.0   100.0          5   24   0.0699 (set_log_writer)
0x19a00     100.0   100.0   100.0          4   22   0.0704 (free_rfd)
0x23390     100.0   100.0   100.0          3   19   0.0712 (lease_set_interface)
0x7790      100.0   71.43   100.0          6   27   0.0718 (sanitise.part.4)
0x1fe10     100.0   100.0   100.0          3   26   0.0729 (dhcp_init)
0x21ea0     100.0   100.0   100.0          4   23   0.0748 (lease_make_duid)
0x29c20     100.0    87.5   100.0          5   30   0.0755 (next)
0xeb80      100.0    80.0   100.0          3   26   0.0758 (addr6part)
0x39630     100.0    50.0   100.0          5   22   0.0764 (get_domain6)
0x7a80      100.0    75.0   100.0          4   20   0.0777 (cache_start_insert)
0xf910      100.0   100.0   100.0          4   35   0.0779 (opt_string_alloc)
0x3a770     100.0    75.0   100.0          7   25   0.0783 (rrfilter_desc)
0xf6a0      100.0   100.0   100.0          5   35   0.0787 (is_tag_prefix)
0x6e50      100.0   63.64   66.67         15   44   0.0794 (cache_hash)
0x2ec70     100.0    50.0   85.71          6   28   0.0804 (mark_context_used
                                                       .isra.3)
0x28820     100.0   100.0   100.0          5   28   0.0808 (buff_alloc.part.1)
0x3a250     100.0   83.33   100.0          4   27   0.0834 (poll_check)
0x3a1a0     100.0   42.86   100.0          9   30   0.0839 (fd_search)
0x18790     100.0   100.0   100.0          3   25    0.084 (allocate_frec)
0xe8a0      100.0   100.0   100.0          3   29   0.0843 (whine_malloc)
0x4109      100.0    75.0   100.0          4   31   0.0862 (add_extradata_opt)
0x6ee0      100.0    50.0    60.0          8   39   0.0864 (cache_free)
0x35710     100.0    62.5    60.0          6   31   0.0867 (ra_start_unsolicited
                                                       .part.4)
0x4138      100.0    80.0   100.0          4   37   0.0885 (pxe_misc)
0x3ef2      100.0   83.33   100.0         12   33    0.089 (calc_time.isra.1)
0x21490     100.0   100.0   100.0          4   31   0.0905 (ourprintf)
0xf4f0      100.0   77.78   100.0          9   31   0.0923 (wildcard_match)
0x29c70     100.0    75.0   66.67          6   31   0.0933 (sanitise)
0x1f900     100.0   66.67    20.0          8   31    0.094 (check_listen_addrs)
0x34050     100.0   100.0   100.0          4   30   0.0947 (put_opt6)
0x34140     100.0    80.0   100.0          6   33   0.0957 (find_iface_param)
0xf130      100.0   83.33   100.0          8   32   0.0959 (memcmp_masked)
0x395d0     100.0   54.55   100.0          8   32    0.096 (get_domain)
0xea30      100.0    80.0   100.0          5   32   0.0993 (hostname_isequal)
0xa040      100.0   100.0   100.0         10   42   0.0994 (crec_ttl.isra.0)
0xe630      100.0    80.0   100.0          8   35   0.1003 (rand64)
0xe830      100.0   100.0   100.0          7   38   0.1019 (safe_pipe)
0xaa90      100.0   100.0   100.0          5   27   0.1032 (skip_questions)
0x36a4      100.0   100.0   100.0          5   37   0.1041 (option_put_string)
0xf5d0      100.0   100.0   100.0          5   47   0.1046 (parse_mysockaddr)
0x1cf40     100.0   55.56   100.0          6   31   0.1049 (warn_int_names)
0x226b0     100.0   100.0   100.0          5   32    0.105 (lease_set_expires)
0x33390     100.0   57.14   100.0          6   41   0.1056 (display_opts6)
0xaeb0      100.0   100.0   100.0         11   49   0.1057 (private_net)
0x1f110     100.0   100.0   100.0          8   49   0.1062 (send_alarm)
0x6dd0      100.0    75.0   100.0          6   40   0.1079 (hash_bucket)
```

```
0xeb10      100.0    80.0   100.0          4   39  0.1084 (is_same_net6)
0x3bfa0     100.0   100.0   100.0          4   32   0.109 (__libc_csu_init)
0xf640      100.0    50.0   83.33         10   37  0.1092 (split_chr)
0x1c490     100.0    40.0   100.0         10   28  0.1106 (label_exception)
0x29dd0     100.0   100.0   100.0          4   35  0.1119 (free_transfer)
0x33fa0     100.0   100.0   100.0          4   31  0.1119 (expand)
0xf740      100.0    87.5   85.71          8   42  0.1132 (atoi_check)
0x32570     100.0   100.0   100.0          3   41  0.1147 (dhcp_common_init)
0x1b360     100.0   81.82   100.0         12   36  0.1172 (server_gone)
0x2ecc0     100.0   92.86   100.0          3   43  0.1178 (build_ia.isra.4)
0x27e00     100.0   100.0   100.0         10   49  0.1182 (nl_async)
0x359e0     100.0    70.0   71.43          9   39  0.1194 (ra_start_unsolicited)
0x23860     100.0   84.62    62.5          8   45  0.1202 (sanitise)
0x1ced0     100.0   55.56   100.0          8   36  0.1211 (warn_wild_labels)
0x215f0     100.0   100.0   92.31          5   44   0.124 (lease_allocate)
0x21da0     100.0    87.5   100.0          5   28  0.1248 (lease_update_slaac)
0x79c0      100.0    62.5    30.0         13   52  0.1258 (cache_enumerate)
0x2b490     100.0   100.0   100.0          3   29  0.1262 (do_tftp_script_run)
0x29b80     100.0   100.0   100.0          9   49  0.1281 (helper_write)
0x7800      100.0   54.55   100.0         10   30  0.1286 (record_source.part.6)
0x33310     100.0   57.14   100.0          6   41  0.1318 (display_opts)
0x33410     100.0    80.0   100.0          7   35  0.1322 (lookup_dhcp_opt)
0x2f3b0     100.0   100.0   100.0          8   51  0.1323 (end_ia.part.7)
0x3f3a      100.0    75.0   57.14         18   48  0.1358 (do_opt)
0xe6b0      100.0    75.0   100.0         13   39  0.1369 (legal_hostname)
0x225e0     100.0   76.92   100.0          9   43   0.138 (lease_find_max_addr)
0x326c0     100.0   45.45   66.67          9   39  0.1391 (run_tag_if)
0x363f      100.0   54.55   88.89          8   38  0.1393 (prune_vendor_opts)
0xf8b0      100.0   100.0   100.0          9   65  0.1409 (opt_malloc)
0x9750      100.0   58.33   100.0         15   42  0.1411 (record_source)
0x2e230     100.0   69.23   100.0          8   45   0.142 (opt6_find.part.1)
0x187f0     100.0   100.0   100.0          7   44  0.1421 (free_frec)
0x1d990     100.0    60.0    87.5          8   48  0.1428 (cleanup_servers)
0xf180      100.0   100.0   100.0          7   50  0.1458 (expand_buf)
0x2dbf0     100.0   100.0   100.0          5   49  0.1465 (check_address)
0x34240     100.0   100.0   100.0          3   48  0.1487 (add_lla)
0x36690     100.0    40.0   100.0          7   48  0.1494 (find_addrlist)
0x21570     100.0   61.54    80.0          8   44  0.1508 (find_interface_v4)
0xf450      100.0   100.0   100.0          5   52  0.1511 (rand_init)
0x2b4f0     100.0   100.0   100.0         11   63  0.1517 (log_reopen)
0xe900      100.0   100.0   100.0          8   56  0.1547 (canonicalise)
0x1eec0     100.0    75.0   94.44          7   52  0.1588 (read_event)
0x36760     100.0   93.33   100.0         10   54  0.1599 (in_zone)
0x214f0     100.0   66.67   77.78          8   42  0.1606 (find_interface_v6)
0x2e180     100.0   100.0   100.0         21   64  0.1619 (calculate_times.isra.0)
0x342d0     100.0   100.0   100.0          7   64  0.1633 (new_timeout.isra.1)
0x20890     100.0   53.85   100.0         14   49  0.1662 (address_available)
0xf7d0      100.0   77.78   93.33         11   65  0.1713 (canonicalise_opt)
0x32c00     100.0   69.23   100.0         11   46  0.1732 (config_has_mac)
0x23af0     100.0   100.0   100.0          9   58  0.1771 (extended_hwaddr)
0x223a0     100.0   70.59   100.0         11   45  0.1786 (lease6_find_by_client)
```

```
0x28790      100.0    90.91    100.0          12    58   0.1798 (grab_extradata)
0x2d5e0      100.0     80.0    100.0           7    47   0.1799 (address6_valid)
0xe990       100.0    100.0    100.0          12    43   0.1806 (sockaddr_isequal)
0x8b50       100.0    100.0    100.0           5    46   0.1817 (a_record_from_hosts)
0x236e0      100.0    85.71    100.0          15    53   0.1845 (option_find1)
0xf970       100.0    100.0    100.0           3    53   0.1845 (add_txt)
0x29cd0      100.0    100.0    100.0           1    56   0.1871 (tftp_err)
0x323e0      100.0     50.0    100.0          16    67   0.1889 (is_config_in_context)
0x2bf10      40.48     50.0    35.29           3    42   0.1957 (die)
0x22420      100.0    84.62    100.0           8    47   0.1974 (lease6_find_by_addr)
0x7ad0       100.0     80.0    72.73          15    46   0.1983 (cache_end_insert)
0x21df0      100.0     50.0    73.33          10    53   0.2012 (lease_find_interfaces)
0x23b50      100.0    76.47    100.0          11    57   0.2036 (find_boot)
0x33e30      100.0    100.0    100.0           3    57   0.2052 (log_relay)
0xac60       100.0    100.0    100.0           7    44   0.2088 (skip_section)
0x2d0e0      100.0    88.24    100.0           9    54   0.2113 (config_find_by_address6)
0x2db60      100.0    69.23     90.0          11    52   0.2118 (get_context_tag)
0x8f50       100.0    92.31    93.33          10    59    0.212 (cache_make_stat)
0x3a7c0      100.0    100.0    100.0           9    60   0.2121 (expand_workspace)
0x331d0      100.0     50.0    100.0          16    63   0.2134 (whichdevice)
0x29a80      100.0    100.0     95.0           8    61   0.2209 (queue_arp)
0x1f1e0      100.0    100.0    100.0          10    59   0.2211 (clear_cache_and_reload)
0x72d0       100.0     90.0    100.0          12    59   0.2227 (eatspace)
0xf550       100.0    82.35    100.0          15    54   0.2228 (wildcard_matchn)
0x2ebb0      100.0    93.75    100.0          12    68   0.2299 (check_ia.isra.2)
0xe450       100.0    71.43    90.91          19    76   0.2324 (check_name)
0xae00       100.0    100.0    100.0           7    64   0.2336 (resize_packet)
0x28670      100.0    85.71    100.0          10    63   0.2412 (netlink_multicast)
0x3bda0      100.0    80.95    93.75          12    71   0.2512 (do_arp_script_run)
0x1b2d0      100.0    100.0    100.0           9    46   0.2519 (resend_query)
0x2d660      100.0    100.0    100.0           9    66   0.2527 (config_valid)
0x1e430      100.0    100.0    100.0          19    76   0.2547 (newaddress)
0xf210       100.0     95.0    100.0           6    61   0.2663 (print_mac)
0x222c0      100.0    68.97    100.0          13    61   0.2682 (lease6_find)
0x3508       100.0    54.17    66.67          25    89   0.2726 (free_space)
0x1ed90      100.0    83.33    100.0           4    97   0.2753 (fatal_event.isra.0)
0x3c6b       100.0    61.54    100.0           4    49    0.277 (log_options)
0x1ce00      100.0     75.0    94.44          10    69   0.2804 (warn_bound_listeners)
0x3bbe       100.0    66.67    100.0          12    65   0.2847 (apply_delay)
0x3441       100.0    90.91    94.44           7    65   0.2894 (add_rev6)
0x9790       100.0    88.24    100.0          17    85   0.2923 (querystr)
0x7090       100.0    64.71    100.0          14    73   0.2957 (add_hosts_cname)
0x324b0      100.0    66.67    100.0          15    69   0.2973 (match_netid.part.1)
0x2bfa0      100.0    100.0    100.0           6    70   0.2988 (make_duid1)
0x1c960      100.0    100.0    100.0          12    83   0.2995 (create_listeners)
0x6f80       100.0    86.67    90.91          18    86   0.3028 (rehash)
0x1f020      63.93    65.22    81.82          27   122   0.3038 (sig_handler)
0x198f0      100.0     75.0    58.33          13    84   0.3042 (allocate_rfd)
0x21380      100.0    100.0    100.0          10    86   0.3046 (host_from_dns)
0x33280      100.0    76.92    100.0          12    84   0.3084 (bindtodevice)
0x32b20      100.0    100.0    100.0          11    69    0.311 (match_bytes)
```

```
0x20920      100.0    52.38    95.65          17    78   0.3138 (narrow_context)
0x3be80      100.0    100.0    100.0          18   110   0.3215 (__umoddi3)
0x221a0      100.0    59.09    100.0          21    79   0.3271 (lease_find_by_client)
0x8be0       100.0    66.67     62.5          11    47    0.328 (cache_unhash_dhcp)
0x397d0      100.0    73.68     90.0          16    68   0.3309 (detect_loop)
0x38fe0      100.0     80.0    100.0          16    63    0.331 (search_domain6)
0x3318       100.0    100.0    96.77          14   101   0.3315
0x3320       100.0    100.0    96.77          12    99   0.3339 (add_rev4)
0x23a20      100.0    93.33    100.0          23   112   0.3414 (option_find)
0x29970      100.0    100.0    96.15           8    76   0.3431 (queue_tftp)
0x7180       100.0    68.18    63.16          21    90    0.345 (add_dhcp_cname)
0x3310       100.0    100.0    96.77          16   103   0.3473
0x1cd30      100.0    63.64    88.89          17    70   0.3491 (create_bound_listeners)
0x32990      100.0    70.83    100.0          19    79   0.3598 (match_netid)
0x3aca0      100.0    52.94    77.78          25   135   0.3601 (calc_subnet_opt)
0x2be70      69.88    72.22     56.0          12    83   0.3605 (flush_log)
0x232b0      100.0     50.0    100.0          12    75   0.3637 (lease_update_
                                                           from_configs)
0xed20       100.0    100.0    100.0          11   111   0.3652 (prettyprint_time)
0x2dc70      100.0    100.0    96.88           6    84   0.3818 (log6_packet)
0x2daa0      100.0    19.05    100.0          20    69   0.3866 (add_local_addrs)
0x22070      100.0    86.36    90.91          14    91   0.3894 (lease_prune)
0xa6a0       100.0    89.47    100.0          15    78   0.3895 (skip_name)
0x32600      100.0    100.0    100.0          13    71   0.3908 (recv_dhcp_packet)
0x224b0      100.0    89.47    100.0          19    89   0.3917 (lease_find_max_addr6)
0x20a60      100.0     76.0    80.95          17    91    0.392 (do_icmp_ping)
0x238e0      100.0    96.55    100.0          16   100   0.3992 (log_packet)
0x7380       100.0    90.48    88.89          20   112   0.4036 (gettok)
0x23400      100.0    72.97    88.46          23   118   0.4191 (do_script_run)
0xe720       100.0    78.95    71.43          19    65   0.4206 (do_rfc1035_name)
0x3a2a0      100.0    92.86    89.29          14   109   0.4335 (poll_listen)
0x23770      100.0     75.0    86.67          15    74   0.4375 (match_vendor_opts)
0x2b5e0      61.02     75.0    85.19          23   118   0.4733 (log_start)
0x3a39       71.79    36.36    60.53          24   117   0.4998 (pxe_uefi_workaround)
0x341a0      100.0     93.1     96.0          15   104    0.507 (send_ra_to_aliases)
0xbae0       100.0    62.96    100.0          33   107   0.5218 (check_for_local_domain)
0x22740      100.0    100.0    100.0          22   126   0.5223 (lease_set_hwaddr)
0x2dda0      100.0    100.0    97.14          10   102    0.526 (log6_quiet)
0x23580      100.0    96.88    95.24          25   111   0.5278 (lease_add_extradata)
0x343a0      100.0    70.59    96.15          23   120   0.5321 (iface_search)
0x2d4c0      100.0    78.26    100.0          17    81   0.5655 (address6_available)
0x1c3a0      100.0    55.88    100.0          29   105   0.5734 (loopback_exception)
0x19a50      100.0    74.07    94.29          37   182   0.5741 (get_new_frec)
0xbe80       100.0     95.0     50.0          25   134   0.5751 (add_resource_record)
0x3ceb       100.0     75.0     87.1          40   172   0.5792 (dhcp_packet_size)
0xbdb0       100.0     92.0    100.0          18    80   0.5845 (check_for_
                                                           ignored_address)
0x3a410      100.0    78.26     75.0          28   122   0.6102 (check_name)
0x3fb0       100.0    69.23    70.59          22   106    0.622 (do_encap_opts)
0x1f7c0      100.0    77.27    83.87          23   136   0.6252 (icmp_ping)
0x32270      100.0    65.71    100.0          24   116   0.6791 (relay_reply6)
```

```
0x39400      100.0    70.0   91.18          30   157   0.6819 (is_rev_synth)
0xebf0       100.0   96.43   91.67          24   212   0.6908 (prettyprint_addr)
0x1fbe0      100.0   100.0   100.0          25   188   0.7317 (make_fd)
0x1ef50      76.34   75.76   89.47          38   186   0.7375 (send_event)
0x36580      100.0   54.84   94.44          16    81   0.7408 (slaac_ping_reply)
0xaad0       100.0   92.31   100.0          19   137   0.7424 (find_soa)
0x36f7       23.89   28.81   33.33          39   226   0.7976 (pxe_opts)
0x35770      100.0    76.0    97.1          32   197   0.7994 (ra_init)
0x16d10      100.0   86.05   100.0          29   170   0.8156 (one_file)
0x1d540      100.0   84.78   97.62          32   177   0.8182 (allocate_sfd)
0x39680      100.0   70.83   100.0          12    96   0.8216 (loop_send_probes)
0x3b970      100.0   78.95    60.0          26   122   0.8391 (filter_mac)
0x2d860      100.0   74.36   81.08          36   168   0.8492 (dhcp_construct_contexts)
0xf390       100.0   100.0   100.0          13    65   0.8671 (read_write)
0x1b740      100.0   61.76    91.3          56   194   0.8803 (iface_check)
0x21f00      100.0    50.0   100.0          31   121   0.8999 (lease_update_dns)
0x98b0       100.0   75.47   100.0          41   214   0.9016 (log_query)
0x1d770      100.0   58.14   85.71          23   150   0.9194 (pre_allocate_sfds)
0x1cc30      100.0    80.0   88.89          27   146   0.9414 (create_wildcard
                                                          _listeners)
0x32a00      100.0    76.0   100.0          14    92   0.9492 (log_tags)
0x32730      100.0   34.04   61.11          54   179   0.9578 (option_filter)
0x27c80      100.0    96.3   100.0          29   149   0.9788 (netlink_recv)
0xba00       100.0   73.68   100.0          50   173   0.9795 (extract_request)
0x2cfb0      100.0   94.74   100.0          16   138   1.0127 (get_client_mac)
0x29e40      100.0    98.0    84.0          34   185   1.0699 (get_block)
0x7860       100.0   93.55   80.65          31   167   1.1247 (cache_init)
0x16f90      100.0   88.24   100.0          34   200   1.1362 (option_read_dynfile)
0x2bb50      100.0   72.22    87.5          36   199   1.1425 (log_write)
0x17390      100.0   32.79   88.89          44   216    1.153 (reread_dhcp)
0xf2d0       100.0   100.0   100.0          22   114   1.1757 (retry_send)
0x228f0      100.0   86.67   100.0          44   200   1.2462 (lease_set_hostname)
0x1da20      100.0   86.79   49.25          32   217   1.2735 (add_update_server)
0xfa10       100.0   92.16   96.72          44   259   1.3091 (parse_server)
0x1f970      100.0   49.25   85.71          34   185   1.3157 (complete_context)
0xa0c0       100.0   86.36   88.89          32   160   1.3456 (extract_name)
0x1c2e0      100.0    66.0   96.43          36   163   1.4144 (iface_allowed_v6)
0x183c0      100.0   75.76   92.16          63   256    1.417 (search_servers)
0x1b6b0      100.0    52.0   88.24          62   237   1.4182 (indextoname)
0x3ae60      100.0   100.0   100.0          32   178   1.4218 (find_pseudoheader)
0x32c80      100.0   71.88   100.0          59   226   1.4299 (find_config)
0xfde0       100.0   73.58   53.42          59   247   1.4581 (one_opt)
0x2be20      100.0    75.0   88.37          41   222   1.6029 (check_log_writer)
0x1c200      100.0   68.85   97.73          44   222   1.6265 (iface_allowed_v4)
0x1c650      100.0   100.0   98.88          46   305   1.6349 (make_sock)
0x8100       100.0   79.69   94.55          42   246   1.6776 (add_hosts_entry)
0x296b0      100.0   100.0   100.0          23   190    1.802 (queue_script)
0x1d460      100.0    87.3   98.44          40   250    1.823 (local_bind)
0x3bb10      100.0    71.7    62.5          50   187   1.8646 (find_mac)
0x2d720      100.0   82.46   88.14          47   252   1.9848 (make_duid)
0xbc30       100.0   81.82   100.0          38   195   1.9873 (check_for_
```

```
                                                    bogus_wildcard)
0x3a570      100.0    78.57    100.0          37   173   2.0717 (check_rrs)
0x8c70       100.0    81.58    80.25          51   304   2.0898 (cache_add_dhcp_entry)
0x2c080      100.0    77.78    76.92          60   231   2.1083 (complete_context6)
0x3b670      100.0    77.78    86.41          43   324   2.1227 (add_edns0_config)
0xee70       100.0    89.47     88.1          47   199   2.2109 (parse_hex)
0x38c20      100.0    90.74    87.78          43   333   2.3724 (add_to_ipset)
0x1e140      100.0    96.55    95.59          46   279   2.4171 (reload_servers)
0x1dd30      100.0    67.61    88.46          70   321   2.6042 (check_servers)
0x33a20      100.0    98.85    96.26          42   323   2.6161 (log_context)
0xa2e0       100.0    78.87    90.32          65   342    2.629 (in_arpa_name_2_addr)
0xe130       100.0     76.4    93.55          32   287   2.6567 (surf)
0x9c20       100.0    100.0    98.08          72   271     2.71 (cache_insert)
0x1ca80      100.0    75.38    94.12          56   281   2.7529 (tcp_interface)
0x7480       100.0    57.38     50.0          65   239   2.8412 (cache_scan_free)
0xacd0       100.0    94.12    100.0          24   160   2.8664 (questions_crc)
0x35220      100.0    81.52    95.92          63   328   2.9025 (add_prefixes)
0x2dde0      100.0    54.29    98.61          32   280   3.0338 (log6_opts)
0x1c5a0      100.0    100.0    99.07          54   360   3.0505 (set_ipv6pktinfo)
0x1f590      100.0     78.0    92.06          47   282    3.066 (delay_dhcp)
0xa790       100.0    69.12    89.13          39   211    3.188 (do_doctor)
0x39090      100.0    62.07    77.42          70   263    3.272 (is_name_synthetic)
0x1b3e0      100.0    52.42    78.67         112   459   3.3457 (enumerate_interfaces
                                                    .part.1)
0x216f0      100.0    48.76    100.0          80   500   3.4642 (lease_update_file)
0x1f4f0      100.0    79.25    93.51          54   333   3.4793 (make_icmp_sock)
0x1c500      100.0    53.17    78.95         117   473    3.515 (enumerate_interfaces)
0x2b0a0      100.0    84.21    84.81          39   295   3.6554 (check_tftp_listeners)
0x8770       100.0    30.95    90.91          59   288   3.6983 (cache_reload)
0x32f40      100.0     70.0    86.21          33   186   3.9202 (dhcp_update_configs)
0x3b570      100.0    70.09    86.55          56   405   3.9515 (check_source)
0x3b0d0      100.0    86.08    90.24          42   321   4.2092 (add_pseudoheader)
0x36310      100.0    42.03    88.46          53   265   4.4032 (periodic_slaac)
0x1d2b0      100.0    78.65    97.85          59   365   4.4705 (random_sock)
0x7e90       100.0    71.96    83.15          86   412   4.5101 (cache_find_by_addr)
0x35d10      100.0     61.4    98.48          34   232   4.5146 (periodic_ra)
0x38a80      100.0    87.93    91.54          62   462   4.5785 (ipset_init)
0x2b7e0      100.0    84.42    90.57          75   431   4.5855 (my_syslog)
0x3a860      100.0    83.75    92.65          39   293   4.6332 (rrfilter)
0x398c0      100.0    70.83    93.65          28   214   5.0515 (inotify_dnsmasq_init)
0x31f10      100.0    79.63    94.12          47   357    5.397 (relay_upstream6)
0x1e750      15.26     6.84    15.27         102   557   5.8808 (check_dns_listeners)
0x2ed50      100.0    75.38    94.56          80   546   5.8911 (add_address)
0x20e00      100.0    63.06    86.96          96   479   6.2549 (dhcp_read_ethers)
0x1f2b0      100.0    74.73     91.3          80   485   6.4184 (poll_resolv)
0x2c9d0      100.0     75.4     94.3          91   515   6.8037 (dhcp6_packet)
0x9290       100.0    60.87    95.65          49   328   7.3379 (dump_cache)
0x334f0      100.0    57.79     88.2         124   702   7.4719 (option_string)
0x22b80      100.0     91.1     97.8          69   543   7.8494 (lease_init)
0x1a930      100.0    58.21    96.15         118   649   8.0985 (receive_query)
0x31db0      100.0    77.21     94.4          57   441   8.2468 (dhcp6_reply)
```

```
0x1fe70      100.0    85.14    98.54         104   641   8.7407 (dhcp_packet)
0x7b80       100.0    65.79    74.05         142   611   9.4551 (cache_find_by_name)
0x1e550      32.76    21.05    28.39         137   702   9.4863 (set_dns_listeners)
0x39e70      100.0    52.46    100.0          45   221   9.5152 (inotify_check)
0x1d000      100.0    78.99    97.28          90   557   10.3167 (join_multicast)
0x2e2a0      100.0    54.55    95.88         138   664   10.5669 (add_options)
0x2c7e0      100.0    78.01    95.22         113   663   11.4995 (dhcp6_init)
0x20b80       5.29    12.57    11.72         140   662   12.4845 (address_allocate)
0x36030      100.0    60.29     89.8          96   463   13.0177 (slaac_add_addrs)
0x35a00      100.0    70.91    99.12          80   456   13.2661 (icmp6_packet)
0x4197       32.36    21.89    51.46         250  1057   14.0618 (do_options)
0x1a430      100.0    57.19    95.73         190  1019   14.9909 (reply_query)
0x8330       100.0    58.17    96.03          96   574   15.4437 (read_hostsfile)
0x16760      100.0    74.23    82.67          89   410   15.4847 (read_file)
0x2d180      100.0     79.1    100.0          30   251   16.5987 (address6_allocate)
0x39b90      100.0    54.72    97.96          80   418   20.8015 (set_dynamic_inotify)
0x17300      100.0    84.55    87.21          92   455   23.0008 (read_servers_file)
0x2c3d0      100.0    76.69     94.1         159   950   24.6418 (construct_worker)
0x34520      100.0     63.8    84.64         220  1162   25.6991 (send_ra_alias)
0x2f440      100.0    44.51    86.26         317  1939   29.9317 (dhcp6_maybe_relay)
0x2a0e0      100.0    58.31    95.44         177  1107   33.4803 (tftp_request)
0x16ff0      100.0    66.24    87.94         128   679   34.7735 (expand_filelist)
0x36800      100.0    58.75    95.35         417  2429   44.2194 (answer_auth)
0x19cd0      100.0    61.99    94.35         279  1497   44.3584 (forward_query.isra.2)
0xaf60       100.0    77.42    93.83         167   866   60.863 (extract_addresses)
0x1b9c0      100.0    48.34    79.17         107   588   61.8368 (iface_allowed)
0x23bf0      83.92    54.76    71.47         483  2804   82.9658 (dhcp_reply)
0x4f80       12.17     19.9    24.15         371  1726   129.8539 (main)
0x17670      60.96    24.73    50.78         230  1150   132.8848 (read_opts)
0x28880       1.88     6.04    11.31         173  1171   152.7092 (create_helper)
0x28040      100.0    55.77    100.0         106   480   222.2085 (iface_enumerate)
0x18f60      100.0    60.69    93.46          98   655   268.803 (tcp_request)
0x18d90      100.0    61.34    94.59         111   765   278.8898 (send_from)
0x18890      100.0    58.63    93.41         170  1112   287.7557 (process_reply
                                                            .isra.0.constprop.3)
0x27ea0      100.0     62.3    100.0         121   584   300.0993 (netlink_init)
0xc420       100.0    62.68    96.98         349  1905   761.5563 (answer_request)
0xc2c0       100.0    63.49     95.1         364  2010   777.2109 (setup_reply)
-------------------------------------------------------------------------------
Disassembly Summary
-------------------------------------------------------------------------------
Instruction count:   64922
Unknown instrs   :       0
Function count   :     544
Function coverage:   85.53%
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Analysis Summary
-------------------------------------------------------------------------------
Esp precision   :   90.54%
Reads precision :   66.27%
```

```
Writes precision:   84.65%
Calls         :    4605
Analysis time   :  4583.36 secs
Iterations     :       8
-------------------------------------------------------------------------------
```

4.2.1.3 comparison with source code

```
Source files with 0 functions compiled:

LOC      name
-----------------------------------
  98     src/conntrack.c
2277     src/dnssec.c
 857     src/dbus.c
 145     src/tables.c
 151     src/blockdata.c
 450     src/bpf.c
-----------------------------------
3978



Number of functions in dnsmasq

               CH (raw) CH (lib)  CH (net)   IDA (raw)  IDA (lib) IDA (net)
-------------------------------------------------------------------------------
32-bit stripped    519    142      377          682    286        396
32-bit unstripped   544    142      402          691    284        407
64-bit stripped                                826    403        423
-------------------------------------------------------------------------------
raw: raw function count
lib: number of dynamically loaded functions
net: number of application functions

-------------------------------------------------------------------------
The following functions exist in the C source code, but are not present
in the (unstripped) executable under the same name. Some of these have
apparently been replaced with an optimized version, as indicated.
constprop: constant propagation
isra    : interprocedural scalar replacement of aggregates
part    : partial evalution?
-------------------------------------------------------------------------
1: add_attr (src/ipset)
2: add_dns_client (src/edns0)
3: add_mac (src/edns0)
4: add_source_addr (src/edns0)
5: async_event (src/dnsmasq)
6: buff_alloc (src/helper)         -> buff_alloc.part.1
```

```
7: build_ia (src/rfc3315)          -> build_ia.isra.4
8: cache_link (src/cache)
9: cache_unlink (src/cache)
10: calc_interval (src/radv)
11: calc_lifetime (src/radv)
12: calc_prio (src/radv)
13: calc_time (src/rfc2131)         -> calc_time.isra.1
14: calculate_times (src/rfc3315)    -> calculate_times.isra.0
15: char64 (src/edns0)
16: check_ia (src/rfc3315)          -> check_ia.isra.2
17: check_tftp_fileperm (src/tftp)
18: crec_ttl (src/rfc1035)          -> crec_ttl.isra.0
19: dhcp6_no_relay (src/rfc3315)
20: dhcp_skip_opts (src/rfc2131)
21: do_usage (src/option)
22: encoder (src/edns0)
23: end_ia (src/rfc3315)            -> end_ia.part.7
24: fatal_event (src/dnsmasq)        -> fatal_event.isra.0
25: find_exclude (src/auth)
26: find_overload (src/rfc2131)
27: find_subnet (src/auth)
28: forward_query (src/forward)      -> forward_query.isra.2
29: free_entry (src/log)
30: get_addrp (src/edns0)
31: get_id (src/forward)
32: hide_meta (src/option)
33: in_list (src/rfc2131)            -> in_list.part.3
34: is_expired (src/cache)           -> is_expired.isra.2.part.3
35: is_outdated_cname_pointer (src/cache)  -> is_outdated_cname_pointer.part.1
36: lookup_frec (src/forward)
37: lookup_frec_by_sender (src/forward)
38: loop_make_probe (src/loop)
39: mark_context_used (src/rfc3315)    -> mark_context_used.isra.3
40: my_readlink (src/inotify)
41: new_add_to_ipset (src/ipset)
42: new_timeout (src/radv)           -> new_timeout.isra.1
43: next_uid (src/cache)
44: old_add_to_ipset (src/ipset)
45: opt6_find (src/rfc3315)          -> opt6_find.part.1
46: opt6_next (src/rfc3315)
47: opt6_uint (src/rfc3315)
48: option_addr (src/rfc2131)
49: option_find2 (src/rfc2131)
50: option_uint (src/rfc2131)
51: parse_dhcp_opt (src/option)
52: private_net6 (src/rfc1035)
53: process_reply (src/forward)       -> process_reply.isra.0.constprop.3
54: read_leases (src/lease)
55: relay_reply4 (src/dhcp)
56: relay_upstream4 (src/dhcp)
57: search_domain (src/domain)
```

```
58: send_ra (src/radv)
59: server_id (src/rfc2131)
60: set_prefix (src/option)
61: split (src/option)
62: unhide_meta (src/option)
63: unhide_metas (src/option)
64: update_leases (src/rfc3315)       -> unhide_metas.part.0


-------------------------------------------------------------------------------
The following functions exist in the (unstripped) executable, but are not
present in the C source code under that name. Some of these are optimized
functions that replace the original; some of these are optimized functions
that are in addition to the original function; some of these are internal
compiler-generated functions
-------------------------------------------------------------------------------
1: __do_global_dtors_aux
2: __libc_csu_fini
3: __libc_csu_init
4: __stack_chk_fail_local
5: __umoddi3
6: __x86.get_pc_thunk.ax
7: __x86.get_pc_thunk.bp
8: __x86.get_pc_thunk.bx
9: __x86.get_pc_thunk.cx
10: __x86.get_pc_thunk.di
11: __x86.get_pc_thunk.dx
12: __x86.get_pc_thunk.si
13: _fini
14: _init
15: _start
16: add_extradata_opt.part.4
17: buff_alloc.part.1               (replaces original)
18: build_ia.isra.4                 (replaces original)
19: cache_get_cname_target.part.5
20: calc_time.isra.1                (replaces original)
21: calculate_times.isra.0           (replaces original)
22: check_ia.isra.2                 (replaces original)
23: crec_ttl.isra.0                 (replaces original)
24: deregister_tm_clones
25: end_ia.part.7                   (replaces original)
26: enumerate_interfaces.part.1
27: extended_hwaddr.part.6
28: fatal_event.isra.0              (replaces original)
29: forward_query.isra.2             (replaces original)
30: frame_dummy
31: in_list.part.3                  (replaces original)
32: is_expired.isra.2.part.3          (replaces original)
33: is_outdated_cname_pointer.part.1   (replaces original)
34: mark_context_used.isra.3          (replaces original)
35: match_netid.part.1
```

```
36: new_timeout.isra.1              (replaces original)
37: opt6_find.part.1                (replaces original)
38: process_reply.isra.0.constprop.3   (replaces original)
39: ra_start_unsolicited.part.4
40: record_source.part.6
41: register_tm_clones
42: sanitise.part.4
43: set_option_bool.part.5
44: unhide_metas.part.0             (replaces original)
```

4.2.2 Analysis Results: mips dnsmasq

```
function     esp    reads    writes  unrc  blocks  instrs  time
--------------------------------------------------------------------------------
0x431020   100.0   100.0   100.0           1      2   0.0199 (__libc_csu_fini)
0x402790   100.0   100.0   100.0           1      4   0.0226 (chdir)
0x402c10   100.0   100.0   100.0           1      4   0.0226 (execl)
0x402a80   100.0   100.0   100.0           1      4   0.0228 (strchr)
0x402f90   100.0   100.0   100.0           1      4   0.0228 (dbus_message
                                                        _new_method_return)
0x402b20   100.0   100.0   100.0           1      4    0.023 (strcasecmp)
0x402eb0   100.0   100.0   100.0           1      4    0.023 (inet_addr)
0x402740   100.0   100.0   100.0           1      4   0.0231 (inet_ntoa)
0x402bc0   100.0   100.0   100.0           1      4   0.0231 (fileno)
0x402d60   100.0   100.0   100.0           1      4   0.0231 (fclose)
0x402860   100.0   100.0   100.0           1      4   0.0232 (geteuid)
0x402ba0   100.0   100.0   100.0           1      4   0.0232 (__errno_location)
0x402700   100.0   100.0   100.0           1      4   0.0233 (memcpy)
0x402c00   100.0   100.0   100.0           1      4   0.0233 (__vsnprintf_chk)
0x402680   100.0   100.0   100.0           1      4   0.0234 (dup)
0x4028b0   100.0   100.0   100.0           1      4   0.0234 (textdomain)
0x402db0   100.0   100.0   100.0           1      4   0.0234 (fopen64)
0x402e70   100.0   100.0   100.0           1      4   0.0234 (sigaction)
0x402760   100.0   100.0   100.0           1      4   0.0235 (ftruncate64)
0x4029f0   100.0   100.0   100.0           1      4   0.0235 (dbus_watch_get_flags)
0x402a40   100.0   100.0   100.0           1      4   0.0235 (kill)
0x402b60   100.0   100.0   100.0           1      4   0.0235 (ungetc)
0x402ff0   100.0   100.0   100.0           1      4   0.0235 (pclose)
0x4028c0   100.0   100.0   100.0           1      4   0.0236 (setgroups)
0x402ac0   100.0   100.0   100.0           1      4   0.0236 (fsync)
0x4027b0   100.0   100.0   100.0           1      4   0.0237 (dbus_connection_ref)
0x4027f0   100.0   100.0   100.0           1      4   0.0237 (recvfrom)
0x402910   100.0   100.0   100.0           1      4   0.0237 (__fxstat64)
0x402cc0   100.0   100.0   100.0           1      4   0.0237 (popen)
0x402e80   100.0   100.0   100.0           1      4   0.0237 (dbus_message_append
                                                        _args)
0x4029d0   100.0   100.0   100.0           1      4   0.0238 (malloc)
0x402b70   100.0   100.0   100.0           1      4   0.0238 (writev)
0x402ce0   100.0   100.0   100.0           1      4   0.0238 (setlocale)
0x4026c0   100.0   100.0   100.0           1      4   0.0239 (memmove)
```

```
0x4028a0   100.0   100.0   100.0           1      4   0.0239 (nfct_new)
0x4028d0   100.0   100.0   100.0           1      4   0.0239 (getsockopt)
0x402c20   100.0   100.0   100.0           1      4   0.0239 (pipe)
0x402c50   100.0   100.0   100.0           1      4   0.0239 (__printf_chk)
0x4027a0   100.0   100.0   100.0           1      4    0.024 (dbus_message_iter
                                                        _get_arg_type)
0x402950   100.0   100.0   100.0           1      4    0.024 (__vsyslog_chk)
0x402be0   100.0   100.0   100.0           1      4    0.024 (prctl)
0x402fa0   100.0   100.0   100.0           1      4    0.024 (opendir)
0x402c70   100.0   100.0   100.0           1      4   0.0241 (__longjmp_chk)
0x4025e0   100.0   100.0   100.0           1      4   0.0242 (dbus_message
                                                      _get_member)
0x402870   100.0   100.0   100.0           1      4   0.0242 (dbus_message
                                                        _iter_append_basic)
0x402ec0   100.0   100.0   100.0           1      4   0.0242 (nfct_callback
                                                        _register)
0x402690   100.0   100.0   100.0           1      4   0.0243 (if_indextoname)
0x402880   100.0   100.0   100.0           1      4   0.0243 (unlink)
0x402b50   100.0   100.0   100.0           1      4   0.0243 (memset)
0x4026f0   100.0   100.0   100.0           1      4   0.0244 (inet_pton)
0x402830   100.0   100.0   100.0           1      4   0.0244 (getuid)
0x402e20   100.0   100.0   100.0           1      4   0.0244 (dbus_message_is
                                                        _method_call)
0x402f00   100.0   100.0   100.0           1      4   0.0244 (umask)
0x402600   100.0   100.0   100.0           1      4   0.0245 (strcmp)
0x402770   100.0   100.0   100.0           1      4   0.0245 (sleep)
0x4029b0   100.0   100.0   100.0           1      4   0.0245 (nfct_set_attr_u32)
0x402a10   100.0   100.0   100.0           1      4   0.0246 (__memcpy_chk)
0x402d70   100.0   100.0   100.0           1      4   0.0246 (recvmsg)
0x402e60   100.0   100.0   100.0           1      4   0.0246 (__fprintf_chk)
0x4025b0   100.0   100.0   100.0           1      4   0.0247 (getpwnam)
0x402650   100.0   100.0   100.0           1      4   0.0247 (read)
0x402850   100.0   100.0   100.0           1      4   0.0247 (wait)
0x402af0   100.0   100.0   100.0           1      4   0.0247 (dbus_connection
                                                      _dispatch)
0x403000   100.0   100.0   100.0           1      4   0.0247 (__sprintf_chk)
0x402750   100.0   100.0   100.0           1      4   0.0248 (memcmp)
0x402780   100.0   100.0   100.0           1      4   0.0248 (select)
0x402ae0   100.0   100.0   100.0           1      4   0.0248 (fchown)
0x402da0   100.0   100.0   100.0           1      4   0.0248 (dbus_message_iter
                                                      _next)
0x402ef0   100.0   100.0   100.0           1      4   0.0248 (__xstat64)
0x402630   100.0   100.0   100.0           1      4   0.0249 (dbus_bus_get)
0x402810   100.0   100.0   100.0           1      4   0.0249 (sysconf)
0x402920   100.0   100.0   100.0           1      4   0.0249 (capget)
0x402a30   100.0   100.0   100.0           1      4   0.0249 (exit)
0x402b90   100.0   100.0   100.0           1      4   0.0249 (putchar)
0x402f50   100.0   100.0   100.0           1      4   0.0249 (recv)
0x402890   100.0   100.0   100.0           1      4    0.025 (nfct_open)
0x402de0   100.0   100.0   100.0           1      4    0.025 (getpeername)
0x402fe0   100.0   100.0   100.0           1      4    0.025 (__ctype_b_loc)
```

```
0x402a20   100.0   100.0   100.0            1      4   0.0251 (nfct_destroy)
0x402c60   100.0   100.0   100.0            1      4   0.0251 (sendto)
0x402ee0   100.0   100.0   100.0            1      4   0.0251 (if_nametoindex)
0x4025a0   100.0   100.0   100.0            1      4   0.0252 (setsockopt)
0x402710   100.0   100.0   100.0            1      4   0.0252 (nfct_close)
0x402a60   100.0   100.0   100.0            1      4   0.0252 (setsid)
0x402b10   100.0   100.0   100.0            1      4   0.0252 (open64)
0x402bf0   100.0   100.0   100.0            1      4   0.0252 (difftime)
0x402df0   100.0   100.0   100.0            1      4   0.0252 (dbus_message_unref)
0x402ea0   100.0   100.0   100.0            1      4   0.0252 (dbus_message
                                                          _iter_init)
0x402640   100.0   100.0   100.0            1      4   0.0253 (dbus_message
                                                          _iter_get_basic)
0x402aa0   100.0   100.0   100.0            1      4   0.0253 (fscanf)
0x402ab0   100.0   100.0   100.0            1      4   0.0253 (strlen)
0x402b00   100.0   100.0   100.0            1      4   0.0253 (write)
0x402bd0   100.0   100.0   100.0            1      4   0.0253 (dbus_watch_handle)
0x402f60   100.0   100.0   100.0            1      4   0.0253 (close)
0x402660   100.0   100.0   100.0            1      4   0.0254 (nfct_set_attr_u16)
0x402840   100.0   100.0   100.0            1      4   0.0254 (dbus_connection_send)
0x402990   100.0   100.0   100.0            1      4   0.0254 (nfct_get_attr_u32)
0x402a00   100.0   100.0   100.0            1      4   0.0254 (strerror)
0x402d80   100.0   100.0   100.0            1      4   0.0254 (getsockname)
0x402dc0   100.0   100.0   100.0            1      4   0.0254 (dbus_connection
                                                          _set_watch_functions)
0x402620   100.0   100.0   100.0            1      4   0.0255 (dbus_connection_set
                                                        _exit_on_disconnect)
0x402670   100.0   100.0   100.0            1      4   0.0255 (fflush)
0x402f40   100.0   100.0   100.0            1      4   0.0255 (getgrnam)
0x402f20   100.0   100.0   100.0            1      4   0.0256 (strtol)
0x430998   100.0   100.0   100.0            1      4   0.0256
0x402930   100.0   100.0   100.0            1      4   0.0257 (waitpid)
0x402c80   100.0   100.0   100.0            1      4   0.0257 (strtok)
0x402c90   100.0   100.0   100.0            1      4   0.0257 (nfct_query)
0x402d10   100.0   100.0   100.0            1      4   0.0257 (fputc)
0x402f80   100.0   100.0   100.0            1      4   0.0257 (nfct_set_attr_u8)
0x4027e0   100.0   100.0   100.0            1      4   0.0258 (dcgettext)
0x402b40   100.0   100.0   100.0            1      4   0.0258 (bind)
0x402e30   100.0   100.0   100.0            1      4   0.0258 (readdir64)
0x402800   100.0   100.0   100.0            1      4   0.0259 (rewind)
0x4028f0   100.0   100.0   100.0            1      4   0.0259 (accept)
0x402ca0   100.0   100.0   100.0            1      4   0.0259 (fork)
0x4028e0   100.0   100.0   100.0            1      4    0.026 (ioctl)
0x402a50   100.0   100.0   100.0            1      4    0.026 (dbus_watch_get
                                                          _enabled)
0x402bb0   100.0   100.0   100.0            1      4    0.026 (strncpy)
0x402e40   100.0   100.0   100.0            1      4    0.026 (nanosleep)
0x402fb0   100.0   100.0   100.0            1      4    0.026 (getgrgid)
0x4025d0   100.0   100.0   100.0            1      4   0.0261 (__snprintf_chk)
0x4027c0   100.0   100.0   100.0            1      4   0.0261 (ctime)
0x402a90   100.0   100.0   100.0            1      4   0.0261 (nfct_set_attr)
```

```
0x402dd0   100.0   100.0   100.0           1      4   0.0261 (dbus_connection
                                                        _register_object_path)
0x402e10   100.0   100.0   100.0           1      4   0.0261 (capset)
0x402940   100.0   100.0   100.0           1      4   0.0263 (strcat)
0x402970   100.0   100.0   100.0           1      4   0.0263 (lseek64)
0x402730   100.0   100.0   100.0           1      4   0.0264 (time)
0x4029e0   100.0   100.0   100.0           1      4   0.0264 (setgid)
0x4026a0   100.0   100.0   100.0           1      4   0.0265 (_exit)
0x4029c0   100.0   100.0   100.0           1      4   0.0265 (gethostname)
0x402d40   100.0   100.0   100.0           1      4   0.0265 (openlog)
0x402e90   100.0   100.0   100.0           1      4   0.0265 (bindtextdomain)
0x402f30   100.0   100.0   100.0           1      4   0.0265 (connect)
0x402820   100.0   100.0   100.0           1      4   0.0266 (_IO_getc)
0x402c40   100.0   100.0   100.0           1      4   0.0266 (dbus_message
                                                        _iter_init_append)
0x403010   100.0   100.0   100.0           1      4   0.0266 (strncat)
0x402cf0   100.0   100.0   100.0           1      4   0.0267 (listen)
0x402900   100.0   100.0   100.0           1      4   0.0268 (dbus_message
                                                        _new_signal)
0x402fc0   100.0   100.0   100.0           1      4   0.0268 (dbus_bus_request_name)
0x402980   100.0   100.0   100.0           1      4   0.0269 (getpid)
0x4027d0   100.0   100.0   100.0           1      4    0.027 (alarm)
0x402f10   100.0   100.0   100.0           1      4    0.027 (shutdown)
0x402d30   100.0   100.0   100.0           1      4   0.0272 (dbus_error_is_set)
0x402e00   100.0   100.0   100.0           1      4   0.0272 (inet_ntop)
0x402e50   100.0   100.0   100.0           1      4   0.0272 (socket)
0x4026e0   100.0   100.0   100.0           1      4   0.0274 (free)
0x402a70   100.0   100.0   100.0           1      4   0.0275 (getopt_long)
0x402960   100.0   100.0   100.0           1      4   0.0276 (strcpy)
0x402b80   100.0   100.0   100.0           1      4   0.0276 (dbus_watch_get
                                                        _unix_fd)
0x402cd0   100.0   100.0   100.0           1      4   0.0277 (sendmsg)
0x402d20   100.0   100.0   100.0           1      4   0.0277 (setuid)
0x402d00   100.0   100.0   100.0           1      4   0.0278 (strrchr)
0x402610   100.0   100.0   100.0           1      4   0.0279 (__vfprintf_chk)
0x4026b0   100.0   100.0   100.0           1      4   0.0279 (dbus_error_init)
0x402558   100.0   100.0   100.0           1      6    0.028
0x402720   100.0   100.0   100.0           1      4    0.028 (fgets)
0x402d50   100.0   100.0   100.0           1      4    0.028 (idna_to_ascii_lz)
0x4026d0   100.0   100.0   100.0           1      4   0.0281 (_setjmp)
0x402d90   100.0   100.0   100.0           1      4   0.0282 (sprintf)
0x402f70   100.0   100.0   100.0           1      4   0.0282 (closedir)
0x40d744   100.0   100.0   100.0           2      6   0.0285 (dnsmasq_time)
0x402c30   100.0   100.0   100.0           1      4   0.0287 (__strcpy_chk)
0x402ed0   100.0   100.0   100.0           1      4   0.0287 (strncmp)
0x402cb0   100.0   100.0   100.0           1      4   0.0292 (sigemptyset)
0x4025c0   100.0   100.0   100.0           1      4   0.0293 (dup2)
0x402fd0   100.0   100.0   100.0           1      4   0.0294 (__ctype_tolower_loc)
0x4025f0   100.0   100.0   100.0           1      4   0.0301 (strstr)
0x402b30   100.0   100.0   100.0           1      4   0.0335 (fcntl)
0x402ad0   100.0   100.0   100.0           1      4   0.0341 (setenv)
```

```
0x4262c8   100.0   100.0   100.0          1      4   0.0355
0x431180   100.0   100.0   100.0          1      6   0.0359
0x41f900   100.0   100.0   100.0          1      8    0.036
0x41d5d4   100.0   100.0   100.0          1     11   0.0362 (lease4_allocate)
0x40d6d0   100.0   100.0   100.0          3      8    0.037 (sa_len)
0x402548   100.0   100.0   100.0          2     10   0.0379
0x40d74c   100.0   100.0   100.0          1      4   0.0384 (is_same_net)
0x42ed18   100.0   100.0   100.0          3      9   0.0397 (save_counter)
0x40e05c   100.0   100.0   100.0          3      9   0.0437 (bump_maxfd)
0x405ebc   100.0   100.0   100.0          4     12   0.0514
0x42eee4   100.0   100.0   100.0          3     13   0.0552 (put_opt6_char)
0x42ee10   100.0   100.0   100.0          3     20   0.0577 (put_opt6)
0x40e290   100.0   100.0   100.0          3     18   0.0603 (set_prefix)
0x417c48   100.0   100.0   100.0          3     23   0.0609 (fix_fd)
0x40e078   100.0   100.0   100.0          4     20   0.0617 (retry_send)
0x42e3f4   100.0   100.0   100.0          3     24   0.0626 (join_multicast)
0x40d4b8   100.0   100.0   100.0          3     26    0.065 (safe_malloc)
0x41f2c4   100.0   100.0   100.0          1     24   0.0666
0x42eea8   100.0   100.0   100.0          3     15   0.0671 (put_opt6_short)
0x4062dc   100.0   100.0   100.0          4     11   0.0672
0x40d818   100.0     0.0   100.0          3     19    0.07 (addr6part)
0x40d864   100.0   100.0     0.0          3     12   0.0715 (setaddr6part)
0x4174e0   100.0   100.0   100.0          4     26   0.0725 (indextoname)
0x41e488   100.0   33.33     0.0          3     12   0.0725
0x41c3f0   100.0   100.0   100.0          3     26   0.0728
0x431140   100.0   100.0   100.0          2     22   0.0728 (__libc_start_main)
0x42ee60   100.0   100.0   100.0          3     18   0.0742 (put_opt6_long)
0x42ef18   100.0   100.0   100.0          4     31   0.0761 (put_opt_string)
0x40d5b4   100.0   100.0   100.0          3     29   0.0763 (whine_malloc)
0x42dccc   100.0   100.0   100.0          5     17   0.0763
0x405dcc    40.0   100.0     0.0          1     20    0.082
0x402508   100.0   100.0   100.0          5     26   0.0836 (_init)
0x40e4a4   100.0   100.0   100.0          4     18   0.0836
0x4310e0   100.0    80.0   100.0          4     21   0.0843
0x4251f8   100.0   100.0   100.0          5     22   0.0845 (my_setenv)
0x41d678   100.0   100.0   100.0          6     21   0.0854 (lease_set_expires)
0x4309b8   100.0   100.0   100.0          5     22   0.0863
0x42edb8   100.0   100.0   100.0          3     22    0.088 (new_opt6)
0x425190   100.0   100.0   100.0          7     26   0.0887
0x41d120   100.0    25.0   100.0          5     16   0.0929 (lease_find_by_addr)
0x4068d4   100.0   83.33   100.0          6     22    0.096
0x41b43c   100.0     0.0   100.0          7     17   0.0967
0x406274   100.0   100.0   100.0          5     26   0.0978 (is_expired)
0x40e2d8   100.0   100.0   100.0          6     26   0.0984 (is_tag_prefix)
0x41a1c8   100.0   100.0   100.0          5     34    0.11
0x40e1e0   100.0   100.0   100.0          8     45   0.1143 (rand_init)
0x419994   100.0   83.33   100.0          3     24   0.1167
0x41eba4   100.0   100.0    75.0          6     24   0.1168
0x42640c   100.0   100.0   100.0          6     29   0.1208
0x415080   100.0   100.0   100.0          3     21   0.1214
0x42ece0   100.0   100.0     0.0          1     14   0.1219 (end_opt6)
```

```
0x41d600   100.0   100.0   100.0          1     30   0.1221 (lease6_allocate)
0x42e73c   100.0     0.0   100.0         13     33   0.1243
0x41fdf0   100.0   100.0   100.0          9     23    0.128
0x42e5fc   100.0    50.0   100.0          5     38   0.1296
0x41ecfc   100.0   100.0   100.0          8     46   0.1302 (option_put_string)
0x42e564   100.0    50.0   100.0          5     38   0.1308
0x4286c0   100.0   100.0   100.0          5     46   0.1322
0x4263b0   100.0   100.0   100.0          5     27   0.1328
0x41c458   100.0   100.0   100.0          5     40   0.1376 (lease_allocate)
0x40e74c   100.0   100.0   100.0          3     17   0.1378
0x40de60   100.0     0.0   100.0          7     19    0.139 (memcmp_masked)
0x406224   100.0   100.0   100.0          6     20   0.1442 (is_outdated
                                                      _cname_pointer)
0x4084e0   100.0   77.78   100.0          5     43   0.1445 (querystr)
0x41a8fc   100.0    70.0   100.0          7     28   0.1478
0x426788   100.0   71.43    60.0          6     27   0.1516
0x40d75c   100.0   88.24   100.0          5     47    0.154 (is_same_net6)
0x41cc88   100.0   100.0   100.0          4     32   0.1541
0x40deac   100.0   100.0   100.0          8     44   0.1544 (expand_buf)
0x428618   100.0   100.0   100.0          8     42   0.1561
0x40d6f0   100.0     0.0   100.0          8     25   0.1565 (hostname_isequal)
0x428568   100.0    80.0    50.0          8     34   0.1599
0x42e694   100.0   66.67   100.0          9     42   0.1631
0x405ef0   100.0    60.0   100.0          7     33   0.1733
0x40e340   100.0   66.67    75.0         11     37   0.1759 (split_chr)
0x4061a8   100.0   100.0   100.0          6     33    0.177
0x40a7f0   100.0   100.0   100.0          3     28   0.1783 (add_mac)
0x431028   100.0    75.0   100.0          3     44   0.1785 (__libc_csu_init)
0x407a84   100.0   100.0   100.0          5     45   0.1793
0x419b28   100.0   100.0   88.89          7     55   0.1815
0x40d354   100.0   100.0   100.0         12     53    0.182 (canonicalise)
0x408480   100.0    40.0   100.0          9     24   0.1826 (record_source)
0x41c4f8   100.0   100.0   100.0          5     31   0.1836
0x40e3d4   100.0   66.67    75.0         12     39   0.1839 (split)
0x405e20   100.0   88.89   100.0          5     39   0.1851
0x41f22c   100.0   81.82   71.43          9     40   0.1878
0x415000   100.0   100.0   100.0          8     32   0.1935
0x409568   100.0   100.0   100.0          7     23    0.196
0x41eda4   100.0   100.0   100.0          5     45   0.2002 (pxe_misc)
0x42ed34   100.0   100.0   100.0          3     33   0.2119 (expand)
0x4267f4   100.0   100.0   100.0          1     50   0.2122
0x41d28c   100.0   84.62   100.0         10     50   0.2166 (lease6_find_by_addr)
0x408c04   100.0   100.0   100.0          6     31    0.218
0x41e850   100.0   33.33   100.0         18     50   0.2274
0x41850c   100.0   100.0   100.0          7     57   0.2283
0x40d520   100.0   100.0   100.0         11     83   0.2288 (safe_pipe)
0x427b24   100.0   100.0   100.0          3     27   0.2322
0x4065c8   100.0   69.23   85.71         14     52    0.234
0x425250   100.0   91.67   100.0         14     45   0.2371 (grab_extradata)
0x40d294   100.0    87.5   100.0         13     48   0.2409 (legal_hostname)
0x41b980   100.0    60.0   100.0          9     39   0.2438
```

```
0x427b90   100.0   100.0   100.0          13     67   0.2461
0x408d34   100.0   100.0   100.0           8     34   0.2634
0x40df5c   100.0   93.33   100.0           9     64   0.2634 (print_mac)
0x429578   100.0   43.75   71.43          17     68   0.2662
0x41b324   100.0   38.89   71.43          17     70   0.2738
0x417ca4   100.0   100.0   100.0           6     55   0.2742
0x40e534   100.0   81.82    75.0          12     44   0.2799
0x40e5dc   100.0   100.0   100.0           2     43   0.2805
0x4262d8   100.0   100.0   100.0          11     51    0.285
0x40d628   100.0   100.0   100.0          12     42    0.286 (sockaddr_isequal)
0x419c04   100.0   100.0   100.0          10     54   0.2895
0x418304   100.0    50.0   100.0           8     24   0.2913
0x42f9a0   100.0   90.91    87.5           7     40    0.311
0x40cd90   100.0   88.46    90.0          17     83   0.3148 (check_name)
0x408c80   100.0   100.0   100.0           7     45   0.3175
0x417430   100.0   81.25   100.0          11     43   0.3239
0x42d9c8   100.0    50.0    75.0           6     32   0.3249
0x429038   100.0   85.71   100.0          11     51    0.326
0x41fe4c   100.0   77.78   100.0          11     47   0.3281 (find_boot)
0x40692c   100.0   92.31   83.33          14     58   0.3351
0x419cdc   100.0   100.0   100.0          19    100   0.3426
0x41919c   100.0   73.33   100.0          11     54   0.3559
0x40b0d0   100.0   55.56   100.0          15     59   0.3664 (extract_request)
0x40d9c8   100.0   100.0   100.0          13    114    0.368 (prettyprint_time)
0x41d160   100.0   70.59    75.0          19     75   0.3709 (lease6_find)
0x41e910   100.0   57.14   83.33          10     46   0.3715
0x428bc8   100.0   100.0   100.0          10     80   0.3779
0x41facc   100.0   82.35   33.33          17     67   0.3792
0x417fd8   100.0   100.0   100.0          11     64   0.3802 (create_listeners)
0x40a14c   100.0   100.0   100.0           8     60   0.3816
0x40e4ec   100.0   83.33   85.71          20     88    0.393
0x40e790   100.0    80.0   92.86          25    106   0.4037
0x429850   100.0   100.0   100.0           6     57   0.4054
0x423bfc   100.0   100.0   100.0          12     83   0.4197
0x40d894   100.0   100.0   66.67          10     77   0.4208 (prettyprint_addr)
0x41e6f0   100.0   95.65   91.67          19     89   0.4254
0x403020   100.0   91.67   91.67          22     97   0.4432
0x4268bc   100.0   100.0   100.0           2     75   0.4437
0x424bf4   100.0   100.0   100.0           8     98   0.4494
0x407b38   100.0    37.5   33.33          11     38   0.4555 (cache_unhash_dhcp)
0x42e44c   100.0   66.67   100.0          13     70   0.4588
0x42de74   100.0   100.0   100.0          13     72   0.4615
0x408b00   100.0     0.0   100.0          18     65   0.4669
0x41e9c8   100.0   59.09   22.22          33    123   0.4715 (free_space)
0x41c2e0   100.0   100.0   100.0          11     68   0.4884
0x41ec04   100.0   78.57   83.33          13     62   0.4897
0x42d6ec   100.0   100.0   100.0          11     79     0.49
0x429940   100.0    50.0   100.0          10     43   0.4978
0x407240   100.0    80.0   100.0          14     60   0.4992
0x405f74   100.0   95.24   100.0          11     70   0.5068 (eatspace)
0x429428   100.0   73.68   100.0          15     84   0.5073
```

```
0x41dc30   100.0    55.0   100.0          13     58   0.5138
0x406690   100.0    90.0   88.89          21     88    0.517
0x42d8b8   100.0   69.57   100.0          19     68   0.5187
0x40e3dc   100.0   72.73    75.0          15     52   0.5235
0x40e678   100.0   84.62    80.0          16     72   0.5241 (canonicalise_opt)
0x426190   100.0   100.0   100.0           9     78   0.5274
0x42d308   100.0   95.83   93.75           7     67   0.5402
0x41b208   100.0   53.85   100.0          12     71   0.5438
0x41cfdc   100.0   60.87   100.0          24     81   0.5588
0x41f324   100.0   72.22   100.0           4     61    0.567
0x41d354   100.0   86.67   100.0          19     91   0.5754 (lease_find_max_addr6)
0x407178   100.0    37.5   100.0           8     50   0.5763
0x41cea0   100.0   86.36   93.75          11     79   0.5924 (lease_prune)
0x42d778   100.0   100.0   100.0          15     80    0.595 (recv_dhcp_packet)
0x4180d0   100.0   100.0   100.0           6     77   0.6282
0x41d4c0   100.0   66.67   100.0           9     69   0.6382
0x413bf4   100.0   83.33   100.0          29    128   0.6393
0x40b1bc   100.0   100.0   100.0          14    113   0.6457
0x41f808   100.0   89.47   100.0          28    114   0.6514
0x424fcc   100.0   100.0   100.0          21    115   0.6826
0x424e94   100.0   66.67   100.0          17     82   0.6908
0x424d7c   100.0    60.0    75.0          12     70    0.693
0x42fa40   100.0   85.71   100.0          19    140   0.7439
0x41a96c   100.0   100.0   100.0          16    132   0.7476
0x424334   100.0   80.65   100.0          20     93   0.7494
0x40b380   100.0   70.59   100.0          32    105   0.7545
0x423ac0   100.0   96.43   100.0          18     79   0.7779
0x418204   100.0    68.0   91.67          16     64   0.7848
0x41583c   100.0   86.36    60.0          15     85   0.7882
0x41fbd0   100.0   68.57   68.75          25    138    0.789
0x416220   100.0   74.19   84.62          34    135   0.9082
0x427c94   100.0   100.0   100.0          24    113   0.9381
0x423910   100.0   100.0   100.0          17     90   0.9664
0x40858c   100.0   91.67   100.0          41    143    0.985 (log_query)
0x429688   100.0   68.42   100.0          28    114   0.9857
0x409170   100.0   95.24   100.0          15    121      1.0
0x428224   100.0   81.82   80.77          37    211   1.0057
0x417d80   100.0   100.0   100.0          28    150   1.0128 (make_sock)
0x4287bc   100.0   100.0   100.0          14    119   1.0384
0x41f920   100.0   100.0   100.0          15    111   1.0432
0x418364   100.0   100.0   100.0          18    108   1.0645
0x40608c   100.0   95.35   88.24          22    141   1.0777 (gettok)
0x406cc8   100.0   78.38   94.44          27    146   1.0911
0x415990   100.0   100.0   100.0          13    125   1.0926
0x41ab7c   100.0    62.5   88.89          24    131   1.1344
0x40b504   100.0    72.5   100.0          17    132   1.1604
0x42df94   100.0   53.33   83.33          21    102   1.1746
0x42da48   100.0   26.32   57.14          43    165   1.2134 (option_filter)
0x417548   100.0   63.16    75.0          36    127    1.321 (iface_check)
0x419f6c   100.0    72.5   71.43          26    153   1.3221
0x41e4b8   100.0   66.67   66.67          30    142   1.3479
```

```
0x430a18   100.0    50.0   100.0          81    390   1.3531
0x4185f0   100.0   97.14   100.0          23    135   1.3575
0x41a6ac   100.0   56.25   71.43          25    150   1.3894
0x40d420   100.0    25.0     0.0          18     44  1.4702 (do_rfc1035_name)
0x419280   100.0    60.0    60.0          33    166   1.4857
0x42d420   100.0    89.8   92.31          31    179   1.5053 (join_multicast_worker)
0x4095c4   100.0   65.71   72.73          46    229   1.6033
0x41cd08   100.0   48.48   100.0          34    104   1.6692
0x40db80   100.0   85.71    80.0          38    188   1.6884 (parse_hex)
0x418804   100.0   88.89   92.31          16    128   1.7612
0x4285cc   100.0    80.7   80.77          42    230   1.7763
0x42ff64   100.0   88.89   88.89          14     72   1.7978
0x41ba1c   100.0   58.82   100.0          47    186   1.8072
0x426480   100.0   98.15   81.48          38    194   1.8858
0x40cedc   100.0   88.24   91.89          18    238   1.9943 (surf)
0x406308   100.0   46.15    68.0          46    176   2.0513
0x406ef8   100.0   69.77   65.52          44    170   2.0577
0x427e58   100.0   95.24   84.85          44    243   2.1149 (my_syslog)
0x425eec   100.0   100.0   100.0          29    171   2.1257 (queue_script)
0x423a70   100.0   95.65   86.11          45    263    2.173
0x4087b8   100.0   100.0   100.0          39    213   2.2089
0x4067d8   100.0   93.55    70.0          33    153   2.2663
0x4154f0   100.0   96.77    92.0          39    213   2.3416
0x40a23c   100.0   100.0   100.0          34    165    2.358
0x42fc70   100.0   81.63   100.0          47    189    2.429
0x40e0c8   100.0   100.0   100.0          17     74   2.4344 (read_write)
0x41ee58   100.0   72.13   51.52          40    245   2.4384 (pxe_opts)
0x428d08   100.0   70.83   96.97          34    204   2.5183
0x408dbc   100.0   66.67   67.86          59    241   2.6066
0x430658   100.0   64.91   88.89          35    210   2.8745
0x419e7c   100.0    75.0   100.0          63    264   2.9296
0x41f418   100.0    80.0   72.73          52    258    3.012
0x41773c   100.0   88.33   95.24          46    243   3.1392
0x4299e4   100.0   97.33   86.27          56    339   3.1529
0x4150d4   100.0   79.69   100.0          68    267   3.2115
0x418dcc   100.0    93.1   68.06          32    244   3.3027
0x41cc60   100.0   61.97   93.75          53    200   3.4978
0x409d0c   100.0   100.0   100.0          25    174   3.5168
0x41d8c0   100.0   87.04   89.47          52    226   3.5553 (lease_set_hostname)
0x406a0c   100.0   72.92   62.86          43    185   3.6349
0x416bc4   100.0    72.5   100.0          66    304   3.6795
0x417074   100.0   76.12   96.97          52    241   3.7236
0x42e7c0   100.0   58.62   73.68          84    337   3.7314
0x41e308   100.0   75.29   91.67          46    225   3.9259
0x42f168   100.0   72.37   95.92          51    311   4.0629 (send_ra)
0x407bc0   100.0   83.33    60.0          50    237   4.2421 (cache_add_dhcp_entry)
0x42dd10   100.0   85.92   88.37          56    332   4.3949
0x41dd18   100.0   99.08   86.96          57    382   4.4126
0x413df4   100.0   63.16   77.42          40    221   4.4252
0x41a250   100.0   90.57   83.72          38    252   4.5037
0x427718   100.0    80.0   65.71          36    265   4.8936
```

```
0x4189fc   100.0   73.08   88.89          44    248   4.9084
0x41d6c4   100.0    80.0   94.37          64    344   5.6152 (lease_set_hwaddr)
0x414160   100.0   44.26   81.82          48    194   6.4359
0x41c574   100.0   65.49   100.0          92    455   6.7027
0x409fc4   100.0   86.67   100.0          24    102    6.757 (questions_crc)
0x41abf0   100.0   83.87   98.33          76    394   7.3192
0x419508   100.0   59.46   84.62          79    307   8.0185 (check_dns_listeners)
0x429b64   100.0    60.0   96.55          36    262   8.0852
0x407744   100.0   78.99   85.48          89    455    9.62
0x415b84   100.0   84.96   95.92          81    427   9.883
0x407f5c   100.0   76.47   91.43          68    333   10.0952 (dump_cache)
0x42e124   100.0    50.0    50.0          35    186   10.602
0x41643c   100.0   77.07   90.67          96    486   13.9192 (forward_query)
0x429104   100.0   70.45   100.0          33    207   28.8268
0x425304   100.0   75.58   89.16         173    947   41.875 (create_helper)
0x409950   100.0    70.0   80.77          44    243   46.0342
0x426920   100.0   82.86   96.52         176    908   47.9203
0x414458   100.0   62.61   83.87         132    750   52.7695 (read_opts)
0x41bcfc   100.0    60.0   60.98          87    383   58.1208 (dhcp_read_ethers)
0x429f6c   100.0   73.62   91.21         193    935   58.5708
0x40a860   100.0   78.18   92.16         110    548   66.2842
0x407330   100.0   100.0   94.29          32    263   85.2826 (read_hostsfile)
0x4305fc   99.75   47.83   94.74         105    404   102.1399
0x417bec   99.75   51.43   97.44         100    404   113.5804
0x403040   100.0   72.79   90.22         283   1078   118.8713 (do_options)
0x41b480   100.0   72.15    87.5          58    326   129.6706
0x41ff08   100.0   79.26   90.88         414   2079   139.5511
0x423d48   100.0   47.69   100.0          97    381   345.7794 (iface_enumerate)
0x404028   100.0   87.57   72.04         347   1726   411.8559 (main)
0x413650   100.0   72.58   68.57          90    373   1758.0871
0x40b714   100.0   72.16   95.81         296   1473   9148.6134
-------------------------------------------------------------------------------
Disassembly Summary
-------------------------------------------------------------------------------
Instruction count:   47908
Unknown instrs   :       0
Function count   :     456
Function coverage:   77.25%
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Analysis Summary
-------------------------------------------------------------------------------
Esp precision   :   99.97%
Reads precision :    78.9%
Writes precision:   89.08%
Calls         :    2446
Analysis time   : 13421.38 secs
Iterations     :      11
-------------------------------------------------------------------------------
```

4.2.3 LLVM Infrastructure

Example output from `chx_read_bitcode.py ptr_add_one –dump`:

```
Magic:     186106078
Version:    0
Offset:    20
Size:      1816
cpu type:   7
header data: 3737142082
================================================================================
1 Enter block 13 (IDENTIFICATION_BLOCK) with blockcode size 5 and wordcount 7
================================================================================
   IDENTIFICATION: APPLE_5_=44.4.7=.6_4
   EPOCH: 0
End block 13
================================================================================
10 Enter block 8 (MODULE_BLOCK) with blockcode size 3 and wordcount 442
================================================================================
   1: [1]
   ============================================================================
   12 Enter block 0 (BLOCKINFO) with blockcode size 2 and wordcount 19
   ============================================================================
      None
      None
      None
   End block 0
   ============================================================================
   34 Enter block 10 (PARAMATTR_GROUP_BLOCK) with blockcode size 3 and wordcount 166
   ============================================================================
      ENTRY:
      ENTRY:0,correctly-rounded-divide-sqrt-fp-math,false,disable-tail-calls,
      false,less-precise-fpmad,false,no-frame-pointer-elim,true,
      no-frame-pointer-elim-non-leaf,no-infs-fp-math,false,no-jump-tables,false,
      no-nans-fp-math,false,no-signed-zeros-fp-math,false,no-trapping-math,false,
      stack-protector-buffer-size,8,target-cpu,penryn,target-features,+cx16,+fxsr,
      +mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87,unsafe-fp-math,false,use-soft-float,
      false
      ENTRY:
   End block 10
   ============================================================================
   202 Enter block 9 (PARAMATTR_BLOCK) with blockcode size 3 and wordcount 2
   ============================================================================
      2: [1, 2]
      2: [3, 2]
   End block 9
   ============================================================================
   206 Enter block 17 (TYPE_BLOCK_NEW) with blockcode size 4 and wordcount 13
   ============================================================================
      NUMENTRY: 11
      i32
      i32*
      i32* (i32*)
      i32* (i32*) *
      i8
      i8*
      i8**
      i32 (i32,i8**)
      i32 (i32,i8**) *
      METADATA
      void
   End block 17
   TRIPLE: i386-apple-macosx10.12.0
   DATALAYOUT: e-m:o-p:32:32-f64:32:64-f80:128-n8:16:32-S128
   FUNCTION: type: i32* (i32*) ; paramattr: 1
```

```
FUNCTION: type: i32 (i32,i8**) ; paramattr: 2
SOURCEFILE_NAME: ptr_add_one.c
VALUE_SYMTAB_OFFSET: 442
================================================================================
260 Enter block 11 (CONSTANTS_BLOCK) with blockcode size 4 and wordcount 6
================================================================================
  SETTYPE: i32
  i32 1
  i32 0
  i32 2
End block 11
================================================================================
268 Enter block 22 (METADATA_KIND_BLOCK) with blockcode size 3 and wordcount 104
================================================================================
  KIND:  0  dbg
  KIND:  1  tbaa
  KIND:  2  prof
  KIND:  3  fpmath
  KIND:  4  range
  KIND:  5  tbaa.struct
  KIND:  6  invariant.load
  KIND:  7  alias.scope
  KIND:  8  noalias
  KIND:  9  nontemporal
  KIND: 10  llvm.mem.parallel_loop_access
  KIND: 11  nonnull
  KIND: 12  dereferenceable
  KIND: 13  dereferenceable_or_null
  KIND: 14  make.implicit
  KIND: 15  unpredictable
  KIND: 16  invariant.group
  KIND: 17  align
  KIND: 18  llvm.loop
  KIND: 19  type
  KIND: 20  section_prefix
  KIND: 21  absolute_symbol
End block 22
================================================================================
374 Enter block 15 (METADATA_BLOCK) with blockcode size 4 and wordcount 45
================================================================================
  STRINGS: 3,4,85,178,6,0  NumRegisterParametersPIC LevelApple LLVM version 9.0.0
                                                          (clang-900.0.39.2)
  VALUE:   0   2
  VALUE:   0   3
  VALUE:   0   4
  NODE: 4,1,5
  NODE: 4,2,6
  NODE: 3
  NAME: llvm.module.flags
  NAMED NODE: 6,7
  NAME: llvm.ident
  NAMED NODE: 8
End block 15
================================================================================
421 Enter block 21 (OPERAND_BUNDLE_TAGS_BLOCK) with blockcode size 3 and wordcount 11
================================================================================
  OPBUNDLETAG: deopt
  OPBUNDLETAG: funclet
  OPBUNDLETAG: gc-transition
End block 21
================================================================================
434 Enter block 12 (FUNCTION_BLOCK) with blockcode size 4 and wordcount 2
================================================================================
  DECLAREBLOCKS: 1
  function-code-43: 1,0,1,4
  INST_RET: 1
End block 12
```

```
  ================================================================================
  438 Enter block 12 (FUNCTION_BLOCK) with blockcode size 4 and wordcount 2
  ================================================================================
    DECLAREBLOCKS: 1
    INST_RET: 4
  End block 12
  ================================================================================
  442 Enter block 14 (VALUE_SYMTAB_BLOCK) with blockcode size 4 and wordcount 9
  ================================================================================
    FN:1: main (at offset: 438)
    FN:0: f (at offset: 434)
  End block 14
End block 8
Block 13: IDENTIFICATION_BLOCK
================================================================================
filename: ptr_add_one.c
================================================================================
Constants
--------------------------------------------------------------------------------
i32 1
i32 0
i32 2

Types
--------------------------------------------------------------------------------
  0: i32
  1: i32*
  2: i32* (i32*)
  3: i32* (i32*) *
  4: i8
  5: i8*
  6: i8**
  7: i32 (i32,i8**)
  8: i32 (i32,i8**) *
  9: METADATA
 10: void

Function signatures
--------------------------------------------------------------------------------
FUNCTION: type: i32* (i32*) ; paramattr: 1
FUNCTION: type: i32 (i32,i8**) ; paramattr: 2

Value symbol table
--------------------------------------------------------------------------------
FN:1: main (at offset: 438)
FN:0: f (at offset: 434)

Functions
================================================================================

LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
RETURN: %2
--------------------------------------------------------------------------------

LLVM Function main: i32 (i32,i8**)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
RETURN: i32 0
--------------------------------------------------------------------------------
```

4.2.4 LLVM Test Cases

4.2.5 Addition

4.2.5.1 add_int_arg

```
c
------------------------------------------------------------------------------
int f(int x, int y) {
  return x+y;
}
------------------------------------------------------------------------------

x86
------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x8(esp,,1)    eax = arg.0008 (= arg.0008_in)
    0x4f1  [   0 ]  add eax, 0x4(esp,,1)    eax := (eax + arg.0004) (= (arg.0008_in
                                                          + arg.0004_in))
    0x4f5  [   0 ]  ret                  return (eax (= (arg.0008_in + arg.0004_in)))
------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32,i32)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = ADD %1, %0
RETURN: %3
------------------------------------------------------------------------------

mips
------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                 return (a0_in + a1_in)
  0x4005c4  [   0 ]  addu $v0, $a0, $a1      v0 := (a0 + a1) (= (a0_in + a1_in))
------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = ADD %0, %1
RETURN: %3
------------------------------------------------------------------------------
```

4.2.5.2 add_int_const

```
c
------------------------------------------------------------------------------
int f(int x) {
  return x+42;
}
------------------------------------------------------------------------------
```

```
x86
------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  add eax, 0x2a          eax := (eax + 42) (= (arg.0004_in + 42))
    0x4f4  [   0 ]  ret                  return (eax (= (arg.0004_in + 42)))
------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
------------------------------------------------------------------------------
i32 42
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ADD %0, 42:i32
RETURN: %2
------------------------------------------------------------------------------

mips
------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return (a0_in + 42)
  0x4005c4  [   0 ]  addiu $v0, $a0, 42      v0 := (a0 + 42) (= (a0_in + 42))
------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
------------------------------------------------------------------------------
i32 42
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ADD %0, 42:i32
RETURN: %2
------------------------------------------------------------------------------
```

4.2.5.3 add_int_one

```
c
------------------------------------------------------------------------------
 int f(int x) {
  return x+1;
}
------------------------------------------------------------------------------

x86
------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  add eax, 0x1           eax := (eax + 1) (= (arg.0004_in + 1))
    0x4f4  [   0 ]  ret                  return (eax (= (arg.0004_in + 1)))
------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
```

```
%2 = ADD %0, 1:i32
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [    0 ]  <ret>                 return (a0_in + 1)
  0x4005c4  [    0 ]  addiu $v0, $a0, 1       v0 := (a0 + 1) (= (a0_in + 1))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ADD %0, 1:i32
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.5.4 add_int_two

```
c
--------------------------------------------------------------------------------
 int f(int x) {
  return x+2;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [    0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [    0 ]  add eax, 0x2             eax := (eax + 2) (= (arg.0004_in + 2))
    0x4f4  [    0 ]  ret                    return (eax (= (arg.0004_in + 2)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ADD %0, 2:i32
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [    0 ]  <ret>                 return (a0_in + 2)
  0x4005c4  [    0 ]  addiu $v0, $a0, 2       v0 := (a0 + 2) (= (a0_in + 2))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ADD %0, 2:i32
```

```
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.6 Branch

4.2.6.1 br_ge_const

```
c
--------------------------------------------------------------------------------
void f(int *p, int n) {
  if (n >= 42) {
    p[0] = 0;
  } else {
    p[1] = 0;
  }
}
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  cmp 0x8(esp,,1), 0x29
    0x4f6  [   0 ]  jg 0x500              if (arg.0008_in > 41) goto 0x500
--------------------------------------------------------------------------------
    0x4f8  [   0 ]  mov 0x4(eax), 0x0       arg.0004_in[4] = 0
    0x4ff  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------
    0x500  [   0 ]  mov (eax), 0x0         arg.0004_in[0] = 0
    0x506  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
i32 41
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SGT  %1, 41:i32
BR i1 %3, label %4, label %5

; <label>:4:
STORE i32 0, %0, align 4
BR label %7

; <label>:5:
%6 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %6, align 4
BR label %7

; <label>:7:
RETURN
--------------------------------------------------------------------------------
```

```
mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  slti    $a1, $a1, 42         a1 := 1 if (a1 < 42) (= (a1_in < 42))
                                                    else 0
  0x4005c4  [   0 ]  bne     $a1, $zero, 0x4005d4   if (a1 <> 0) (= (a1_in < 42))
                                                 then goto 0x4005d4
  0x4005c8  [   0 ]  <nop>
--------------------------------------------------------------------------------
  0x4005cc  [   0 ]  <ret>                      return v0_in
  0x4005d0  [   0 ]  sw      $zero, ($a0)         a0_in[0] := 0
--------------------------------------------------------------------------------
  0x4005d4  [   0 ]  <ret>                      return v0_in
  0x4005d8  [   0 ]  sw      $zero, 0x4($a0)      a0_in[4] := 0
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SLT  %1, 42:i32
BR i1 %3, label %4, label %6

; <label>:4:
%5 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %5, align 4
BR label %7

; <label>:6:
STORE i32 0, %0, align 4
BR label %7

; <label>:7:
RETURN
--------------------------------------------------------------------------------
```

4.2.6.2 br_ge_zero

```
c
--------------------------------------------------------------------------------
void f(int *p, int n) {
  if (n >= 0) {
    p[0] = 0;
  } else {
    p[1] = 0;
  }
}
```

```
x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  cmp 0x8(esp,,1), 0x0
    0x4f2  [   0 ]  js 0x4ff              if (arg.0008_in < 0) goto 0x4ff
--------------------------------------------------------------------------------
    0x4f4  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f8  [   0 ]  mov (eax), 0x0         arg.0004_in[0] = 0
    0x4fe  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------
    0x4ff  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x503  [   0 ]  mov 0x4(eax), 0x0       arg.0004_in[4] = 0
    0x50a  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SLT  %1, i32 0
BR i1 %3, label %4, label %6

; <label>:4:
%5 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %5, align 4
BR label %7

; <label>:6:
STORE i32 0, %0, align 4
BR label %7

; <label>:7:
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  bltz    $a1, 0x4005d0   if (a1 < 0) (= (a1_in < 0)) then
                                        goto 0x4005d0
  0x4005c4  [   0 ]  <nop>
--------------------------------------------------------------------------------
  0x4005c8  [   0 ]  <ret>                  return v0_in
  0x4005cc  [   0 ]  sw      $zero, ($a0)    a0_in[0] := 0
--------------------------------------------------------------------------------
  0x4005d0  [   0 ]  <ret>                  return v0_in
  0x4005d4  [   0 ]  sw      $zero, 0x4($a0)  a0_in[4] := 0
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SLT  %1, i32 0
BR i1 %3, label %4, label %6
```

```
; <label>:4:
%5 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %5, align 4
BR label %7

; <label>:6:
STORE i32 0, %0, align 4
BR label %7

; <label>:7:
RETURN
--------------------------------------------------------------------------------
```

4.2.7 Comparison

4.2.7.1 ge_arg_arg (mips only)

```
c
--------------------------------------------------------------------------------
int f(int x, int y) {
  return x >= y;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  slt $v0, $a0, $a1       v0 := 1 if (a0 < a1) (= (a0_in < a1_in))
                                               else 0
  0x4005c4  [   0 ]  <ret>                 return (a0_in >= a1_in)
  0x4005c8  [   0 ]  xori $v0, $v0, 1        v0 := (v0 xor 1) (= ((a0_in < a1_in) xor 1))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = ICMP SGE  %0, %1
%4 = ZEXT %3 to i32
RETURN: %4
--------------------------------------------------------------------------------
```

4.2.7.2 ge_arg_const (mips only)

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x >= 42;
}
--------------------------------------------------------------------------------
```

```
mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  slti $v0, $a0, 42      v0 := 1 if (a0 < 42) (= (a0_in < 42)) else 0
  0x4005c4  [   0 ]  <ret>                return (a0_in >= 42)
  0x4005c8  [   0 ]  xori $v0, $v0, 1       v0 := (v0 xor 1) (= ((a0_in < 42) xor 1))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ICMP SGE  %0, 42:i32
%3 = ZEXT %2 to i32
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.7.3 ge_arg_zero (mips only)

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x >= 0;
}
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  nor $v0, $zero, $a0     v0 := (0 bnor a0) (= (0 bnor a0_in))
  0x4005c4  [   0 ]  <ret>                return (a0_in >= 0)
  0x4005c8  [   0 ]  srl $v0, $v0, 31       v0 := (v0 / 0x80000000) (= ((0 bnor a0_in)
                                                                / 0x80000000))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = ICMP SGE  %0, i32 0
%3 = ZEXT %2 to i32
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.8 Mixed expressions

4.2.8.1 mixed_mul_plus

```
c
--------------------------------------------------------------------------------
```

```
int f(int x, int y) {
  return (3 * x) + ( 4 * y);
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  lea eax, (eax,eax,2)    eax = (eax + (2 * eax)) (= (arg.0004_in
                                                    + (2 * arg.0004_in)))
    0x4f4  [   0 ]  mov edx, 0x8(esp,,1)    edx = arg.0008 (= arg.0008_in)
    0x4f8  [   0 ]  lea eax, (eax,edx,4)    eax = (eax + (4 * edx)) (=
                                           ((3 * arg.0004_in) + (4 * arg.0008_in)))
    0x4fb  [   0 ]  ret                    return (eax (= ((4 * arg.0008_in)
                                             + (3 * arg.0004_in))))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
i32 4
i32 3
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = MUL 4:i32, %1
%4 = MUL 3:i32, %0
%5 = ADD %3, %4
RETURN: %5
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  sll     $v0, $a0, 1    v0 := (a0 * 2) (= (2 * a0_in))
  0x4005c4  [   0 ]  addu    $v0, $v0, $a0   v0 := (v0 + a0) (= ((2 * a0_in) + a0_in))
  0x4005c8  [   0 ]  sll     $a1, $a1, 2    a1 := (a1 * 4) (= (4 * a1_in))
  0x4005cc  [   0 ]  <ret>                 return ((3 * a0_in) + (4 * a1_in))
  0x4005d0  [   0 ]  addu    $v0, $v0, $a1   v0 := (v0 + a1) (= ((3 * a0_in)
                                                    + (4 * a1_in)))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
i32 3
i32 4
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = MUL 3:i32, %0
%4 = MUL 4:i32, %1
%5 = ADD %3, %4
RETURN: %5
--------------------------------------------------------------------------------
```

4.2.9 Multiplication

4.2.9.1 mul_int_arg (mips only)

```
c
--------------------------------------------------------------------------------
int f(int x, int y) {
  return x*y;
}
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  mult $a0, $a1          (hi,lo) := (a0 * a1) (= (a0_in * a1_in))
  0x4005c4  [   0 ]  mflo $v0              v0 := (a0_in * a1_in)
  0x4005c8  [   0 ]  <ret>                 return (a0_in * a1_in)
  0x4005cc  [   0 ]  <nop>
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = MUL %0, %1
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.9.2 mul_int_const (mips only)

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x*42;
}
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  sll $v0, $a0, 1        v0 := (a0 * 2) (= (2 * a0_in))
  0x4005c4  [   0 ]  sll $a0, $a0, 3        a0 := (a0 * 8) (= (8 * a0_in))
  0x4005c8  [   0 ]  subu $a0, $a0, $v0      a0 := (a0 - v0) (= (6 * a0_in))
  0x4005cc  [   0 ]  sll $v0, $a0, 3        v0 := (a0 * 8) (= (48 * a0_in))
  0x4005d0  [   0 ]  <ret>                 return (42 * a0_in)
  0x4005d4  [   0 ]  subu $v0, $v0, $a0      v0 := (v0 - a0) (= (42 * a0_in))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
```

```
%2 = MUL 42:i32, %0
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.9.3 mul_int_three

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x*3;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  lea eax, (eax,eax,2)    eax = (eax + (2 * eax)) (= (arg.0004_in
                                                    + (2 * arg.0004_in)))
    0x4f4  [   0 ]  ret                  return (eax (= (3 * arg.0004_in)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
Local constant definitions
--------------------------------------------------------------------------------
i32 3
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = MUL 3:i32, %0
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  sll $v0, $a0, 1        v0 := (a0 * 2) (= (2 * a0_in))
  0x4005c4  [   0 ]  <ret>                 return (3 * a0_in)
  0x4005c8  [   0 ]  addu $v0, $v0, $a0      v0 := (v0 + a0) (= ((2 * a0_in) + a0_in))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
i32 3
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = MUL 3:i32, %0
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.9.4 mul_int_two

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x*2;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  add eax, eax           eax := (eax + eax) (= (arg.0004_in
                                                    + arg.0004_in))
    0x4f3  [   0 ]  ret                  return (eax (= (2 * arg.0004_in)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = MUL 2:i32, %0
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return (2 * a0_in)
  0x4005c4  [   0 ]  sll $v0, $a0, 1        v0 := (a0 * 2) (= (2 * a0_in))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = MUL 2:i32, %0
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.10 Phi Expressions

4.2.10.1 phi_ge_zero

```
c
--------------------------------------------------------------------------------
int f(int *p, int n) {
  int x;
  if (n > 0) {
    x = p[0] + 1;
  } else {
    x = p[1];
  }
  return x;
}
```

```
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  cmp 0x8(esp,,1), 0x0
    0x4f2  [   0 ]  jle 0x4fe             if (arg.0008_in <= 0) goto 0x4fe
--------------------------------------------------------------------------------
    0x4f4  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f8  [   0 ]  mov eax, (eax)         eax = arg.0004_in[0] (= arg.0004_in[0]_in)
    0x4fa  [   0 ]  add eax, 0x1           eax := (eax + 1) (= (arg.0004_in[0]_in + 1))
    0x4fd  [   0 ]  ret                   return (eax (= (arg.0004_in[0]_in + 1)))
--------------------------------------------------------------------------------
    0x4fe  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x502  [   0 ]  mov eax, 0x4(eax)       eax = arg.0004_in[4] (= arg.0004_in[4]_in)
    0x505  [   0 ]  ret                   return (eax (= arg.0004_in[4]_in))
--------------------------------------------------------------------------------

LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SLE  %1, i32 0
BR i1 %3, label %4, label %7

; <label>:4:
%5 = getelementptr inbounds i32, %0, i32 1
%6 = LOAD i32, %5, align 4
BR label %10

; <label>:7:
%8 = LOAD i32, %0, align 4
%9 = ADD %8, 1:i32
BR label %12

; <label>:10:
%11 = PHI i32  [ %6, %4 ]  [ %9, %7 ]
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  blez    $a1, 0x4005d4   if (a1 <= 0) (= (a1_in <= 0))
                                                then goto 0x4005d4
  0x4005c4  [   0 ]  <nop>
--------------------------------------------------------------------------------
  0x4005c8  [   0 ]  lw      $v0, ($a0)     v0 := a0_in[0]_in
  0x4005cc  [   0 ]  <ret>                 return (a0_in[0]_in + 1)
  0x4005d0  [   0 ]  addiu   $v0, $v0, 1    v0 := (v0 + 1) (= (a0_in[0]_in + 1))
--------------------------------------------------------------------------------
  0x4005d4  [   0 ]  <ret>                 return a0_in[4]_in
  0x4005d8  [   0 ]  lw      $v0, 0x4($a0)   v0 := a0_in[4]_in
--------------------------------------------------------------------------------
```

```
mips to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 4
%3 = ICMP SLE  %1, i32 0
BR i1 %3, label %4, label %7

; <label>:4:
%5 = getelementptr inbounds i32, %0, i32 1
%6 = LOAD i32, %5, align 4
BR label %10

; <label>:7:
%8 = LOAD i32, %0, align 4
%9 = ADD %8, 1:i32
BR label %12

; <label>:10:
%11 = PHI i32  [ %6, %4 ]  [ %9, %7 ]
RETURN
--------------------------------------------------------------------------------
```

4.2.11 Pointer expressions

4.2.11.1 ptr_add_arg

```
c
--------------------------------------------------------------------------------
int *f(int *p, int n) {
  return p + n;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x8(esp,,1)    eax = arg.0008 (= arg.0008_in)
    0x4f1  [   0 ]  shl eax, 0x2           eax = (4 * eax) (= (4 * arg.0008_in))
    0x4f4  [   0 ]  add eax, 0x4(esp,,1)    eax := (eax + arg.0004) (= ((4
                                                 * arg.0008_in) + arg.0004_in))
    0x4f8  [   0 ]  ret                  return (eax (= ((4 * arg.0008_in)
                                                        + arg.0004_in)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = getelementptr inbounds i32, %0, %1
RETURN: %3
--------------------------------------------------------------------------------
```

```
mips
----------------------------------------------------------------------------
  0x4005c0  [    0 ]  sll $v0, $a1, 2        v0 := (a1 * 4) (= (4 * a1_in))
  0x4005c4  [    0 ]  <ret>                 return (a0_in + (4 * a1_in))
  0x4005c8  [    0 ]  addu $v0, $a0, $v0      v0 := (a0 + v0) (= (a0_in + (4 * a1_in)))
----------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*,i32)
----------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = getelementptr inbounds i32, %0, %1
RETURN: %3
----------------------------------------------------------------------------
```

4.2.11.2 ptr_add_const

```
c
----------------------------------------------------------------------------
int *f(int *p) {
  return p + 42;
}
----------------------------------------------------------------------------

x86
----------------------------------------------------------------------------
    0x4ed  [    0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [    0 ]  add eax, 0xa8          eax := (eax + 168) (= (arg.0004_in + 168))
    0x4f6  [    0 ]  ret                   return (eax (= (arg.0004_in + 168)))
----------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*)
----------------------------------------------------------------------------
i32 42
----------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
RETURN: %2
----------------------------------------------------------------------------

mips
----------------------------------------------------------------------------
  0x4005c0  [    0 ]  <ret>                 return (a0_in + 168)
  0x4005c4  [    0 ]  addiu $v0, $a0, 168     v0 := (a0 + 168) (= (a0_in + 168))
----------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*)
----------------------------------------------------------------------------
i32 42
----------------------------------------------------------------------------
```

```
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.11.3 ptr_add_one

```
c
--------------------------------------------------------------------------------
int *f(int *p) {
  return p + 1;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  add eax, 0x4           eax := (eax + 4) (= (arg.0004_in + 4))
    0x4f4  [   0 ]  ret                   return (eax (= (arg.0004_in + 4)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                 return (a0_in + 4)
  0x4005c4  [   0 ]  addiu $v0, $a0, 4       v0 := (a0 + 4) (= (a0_in + 4))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.11.4 ptr_load_const

```
c
--------------------------------------------------------------------------------
int f(int *p) {
  return p[42];
}
```

```
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov eax, 0xa8(eax)      eax = arg.0004_in[168]
                                              (= arg.0004_in[168]_in)
    0x4f7  [   0 ]  ret                   return (eax (= arg.0004_in[168]_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
%3 = LOAD i32, %2, align 4
RETURN: %3
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return a0_in[168]_in
  0x4005c4  [   0 ]  lw $v0, 0xa8($a0)      v0 := a0_in[168]_in
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
%3 = LOAD i32, %2, align 4
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.11.5 ptr_load_one

```
c
--------------------------------------------------------------------------------
int f(int *p) {
  return p[1];
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov eax, 0x4(eax)       eax = arg.0004_in[4] (= arg.0004_in[4]_in)
    0x4f4  [   0 ]  ret                   return (eax (= arg.0004_in[4]_in))
```

```
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
%3 = LOAD i32, %2, align 4
RETURN: %3
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return a0_in[4]_in
  0x4005c4  [   0 ]  lw $v0, 0x4($a0)       v0 := a0_in[4]_in
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
%3 = LOAD i32, %2, align 4
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.11.6 ptr_load_zero

```
c
--------------------------------------------------------------------------------
int f(int *p) {
  return p[0];
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov eax, (eax)         eax = arg.0004_in[0] (= arg.0004_in[0]_in)
    0x4f3  [   0 ]  ret                  return (eax (= arg.0004_in[0]_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = LOAD i32, %0, align 4
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                              return a0_in[0]_in
```

```
  0x4005c4  [    0 ]  lw      $v0, ($a0)                 v0 := a0_in[0]_in
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = LOAD i32, %0, align 4
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.11.7 ptr_store_const

```
c
--------------------------------------------------------------------------------
void f(int *p) {
  p[42] = 0;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [    0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [    0 ]  mov 0xa8(eax), 0x0      arg.0004_in[168] = 0
    0x4fb  [    0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
STORE i32 0, %2, align 4
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [    0 ]  <ret>                 return v0_in
  0x4005c4  [    0 ]  sw $zero, 0xa8($a0)     a0_in[168] := 0
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
STORE i32 0, %2, align 4
RETURN
```

```
--------------------------------------------------------------------------------
```

4.2.11.8 ptr_store_const_arg

```
c
--------------------------------------------------------------------------------
void f(int *p, int c) {
  p[42] = c;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov edx, 0x8(esp,,1)    edx = arg.0008 (= arg.0008_in)
    0x4f5  [   0 ]  mov 0xa8(eax), edx      arg.0004_in[168] = edx (= arg.0008_in)
    0x4fb  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = getelementptr inbounds i32, %0, i32 42
STORE %1, %3, align 4
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                 return v0_in
  0x4005c4  [   0 ]  sw $a1, 0xa8($a0)       a0_in[168] := a1 (= a1_in)
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*,i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = getelementptr inbounds i32, %0, i32 42
STORE %1, %3, align 4
RETURN
--------------------------------------------------------------------------------
```

4.2.11.9 ptr_store_const_const

```
c
--------------------------------------------------------------------------------
```

```
void f(int *p) {
  p[42] = 43;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov 0xa8(eax), 0x2b     arg.0004_in[168] = 43
    0x4fb  [   0 ]  ret                   return (eax (= arg.0004_in))
--------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*)
--------------------------------------------------------------------------------
i32 43
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
STORE i32 43, %2, align 4
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  li $v0, 43            v0 := 43
  0x4005c4  [   0 ]  <ret>                return 43
  0x4005c8  [   0 ]  sw $v0, 0xa8($a0)      a0_in[168] := v0 (= 43)
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*)
--------------------------------------------------------------------------------
i32 43
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 42
STORE i32 43, %2, align 4
RETURN
--------------------------------------------------------------------------------
```

4.2.11.10 ptr_store_one

```
c
--------------------------------------------------------------------------------
void f(int *p) {
  p[1] = 0;
}
--------------------------------------------------------------------------------
```

```
x86
------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov 0x4(eax), 0x0       arg.0004_in[4] = 0
    0x4f8  [   0 ]  ret                  return (eax (= arg.0004_in))
------------------------------------------------------------------------------

x86 to LLVM Function f: void (i32*)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %2, align 4
RETURN
------------------------------------------------------------------------------

mips
------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return v0_in
  0x4005c4  [   0 ]  sw $zero, 0x4($a0)      a0_in[4] := 0
------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 1
STORE i32 0, %2, align 4
RETURN
------------------------------------------------------------------------------
```

4.2.11.11 ptr_store_zero

```
c
------------------------------------------------------------------------------
void f(int *p) {
  p[0] = 0;
}
------------------------------------------------------------------------------

x86
------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  mov (eax), 0x0         arg.0004_in[0] = 0
    0x4f7  [   0 ]  ret                  return (eax (= arg.0004_in))
------------------------------------------------------------------------------


x86 to LLVM Function f: void (i32*)
------------------------------------------------------------------------------
DECLAREBLOCKS: 1
STORE i32 0, %0, align 4
```

```
RETURN
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return v0_in
  0x4005c4  [   0 ]  sw $zero, ($a0)        a0_in[0] := 0
--------------------------------------------------------------------------------

mips to LLVM Function f: void (i32*)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
STORE i32 0, %0, align 4
RETURN
--------------------------------------------------------------------------------
```

4.2.11.12 ptr_sub_arg

```
c
--------------------------------------------------------------------------------
int *f(int *p, int n) {
  return p - n;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x8(esp,,1)    eax = arg.0008 (= arg.0008_in)
    0x4f1  [   0 ]  shl eax, 0x2           eax = (4 * eax) (= (4 * arg.0008_in))
    0x4f4  [   0 ]  mov edx, 0x4(esp,,1)    edx = arg.0004 (= arg.0004_in)
    0x4f8  [   0 ]  sub edx, eax           edx := (edx - eax) (= (arg.0004_in
                                                          - (4 * arg.0008_in)))
    0x4fa  [   0 ]  mov eax, edx           eax = edx (= (arg.0004_in
                                                          - (4 * arg.0008_in)))
    0x4fc  [   0 ]  ret                  return (eax (= (arg.0004_in
                                                          - (4 * arg.0008_in))))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = SUB i32 0, %1
%4 = getelementptr inbounds i32, %0, %3
RETURN: %4
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  sll     $v0, $a1, 2                v0 := (a1 * 4)
                                                            (= (4 * a1_in))
```

```
  0x4005c4  [    0 ]  <ret>                              return (a0_in - (4 * a1_in))
  0x4005c8  [    0 ]  subu    $v0, $a0, $v0               v0 := (a0 - v0)
                                                      (= (a0_in - (4 * a1_in)))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = SUB i32 0, %1
%4 = getelementptr inbounds i32, %0, %3
RETURN: %4
--------------------------------------------------------------------------------
```

4.2.11.13 ptr_sub_const

```
c
--------------------------------------------------------------------------------
int *f(int *p) {
  return p - 42;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [    0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [    0 ]  sub eax, 0xa8          eax := (eax - 168) (= (arg.0004_in - 168))
    0x4f6  [    0 ]  ret                  return (eax (= (arg.0004_in - 168)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
i32 -42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 -42
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [    0 ]  <ret>                 return (a0_in - 168)
  0x4005c4  [    0 ]  addiu $v0, $a0, -168    v0 := (a0 + -168) (= (a0_in - 168))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
i32 -42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 -42
```

```
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.11.14 ptr_sub_one

```
c
--------------------------------------------------------------------------------
int *f(int *p) {
  return p - 1;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  sub eax, 0x4           eax := (eax - 4) (= (arg.0004_in - 4))
    0x4f4  [   0 ]  ret                   return (eax (= (arg.0004_in - 4)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
i32 -1
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 -1
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                 return (a0_in - 4)
  0x4005c4  [   0 ]  addiu $v0, $a0, -4      v0 := (a0 + -4) (= (a0_in - 4))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32* (i32*)
--------------------------------------------------------------------------------
i32 -1
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = getelementptr inbounds i32, %0, i32 -1
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.12 Subtraction

4.2.12.1 sub_int_arg

```
c
--------------------------------------------------------------------------------
```

```
int f(int x, int y) {
  return x-y;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  sub eax, 0x8(esp,,1)    eax := (eax - arg.0008)
                                               (= (arg.0004_in - arg.0008_in))
    0x4f5  [   0 ]  ret                  return (eax (= (arg.0004_in - arg.0008_in)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = SUB %0, %1
RETURN: %3
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return (a0_in - a1_in)
  0x4005c4  [   0 ]  subu $v0, $a0, $a1      v0 := (a0 - a1) (= (a0_in - a1_in))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32,i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%3 = SUB %0, %1
RETURN: %3
--------------------------------------------------------------------------------
```

4.2.12.2 sub_int_const

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x-42;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  sub eax, 0x2a          eax := (eax - 42) (= (arg.0004_in - 42))
    0x4f4  [   0 ]  ret                  return (eax (= (arg.0004_in - 42)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
```

```
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 42:i32
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return (a0_in - 42)
  0x4005c4  [   0 ]  addiu $v0, $a0, -42     v0 := (a0 + -42) (= (a0_in - 42))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
i32 42
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 42:i32
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.12.3 sub_int_one

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x-1;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  sub eax, 0x1           eax := (eax - 1) (= (arg.0004_in - 1))
    0x4f4  [   0 ]  ret                   return (eax (= (arg.0004_in - 1)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 1:i32
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                return (a0_in - 1)
  0x4005c4  [   0 ]  addiu $v0, $a0, -1      v0 := (a0 + -1) (= (a0_in - 1))
```

```
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 1:i32
RETURN: %2
--------------------------------------------------------------------------------
```

4.2.12.4 sub_int_two

```
c
--------------------------------------------------------------------------------
int f(int x) {
  return x-2;
}
--------------------------------------------------------------------------------

x86
--------------------------------------------------------------------------------
    0x4ed  [   0 ]  mov eax, 0x4(esp,,1)    eax = arg.0004 (= arg.0004_in)
    0x4f1  [   0 ]  sub eax, 0x2           eax := (eax - 2) (= (arg.0004_in - 2))
    0x4f4  [   0 ]  ret                   return (eax (= (arg.0004_in - 2)))
--------------------------------------------------------------------------------

x86 to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 2:i32
RETURN: %2
--------------------------------------------------------------------------------

mips
--------------------------------------------------------------------------------
  0x4005c0  [   0 ]  <ret>                 return (a0_in - 2)
  0x4005c4  [   0 ]  addiu $v0, $a0, -2      v0 := (a0 + -2) (= (a0_in - 2))
--------------------------------------------------------------------------------

mips to LLVM Function f: i32 (i32)
--------------------------------------------------------------------------------
DECLAREBLOCKS: 1
%2 = SUB %0, 2:i32
RETURN: %2
--------------------------------------------------------------------------------
```

5.0 CONCLUSION

Detailed provenance tracking provides information that can be critical to the rapid understanding of information and privacy leaks. To date, however, the overhead and complexity of obtaining such information has hampered the development of systems that can deliver this information. ClearScope, with its combination of split device/server design and effective compiler optimizations, enables, for the first time, the ability to collect the information required to build a complete, byte-level provenance web that tracks the complete path each byte follows through the system. Experience using ClearScope on the Adups FOTA malware highlights the benefits that this information can deliver in this context; performance results highlight the performance benefits that its compiler optimizations can deliver.

6.0 References

[1] Samak, M., Kim, D., and Rinard, M., "Synthesizing Replacement Classes," in *47th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*

[2] Achour, Sara and Rinard, Martin, "Time Dilation and Contraction for Programmable Analog Devices with Jaunt," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, New York, NY, USA, ASPLOS '18, pp. 229–242, URL http://doi.acm.org/10.1145/3173162.3173179

[3] Gordon, Michael, Eikenberry, Jordan, Eden, Anthony, Perkins, Jeff, and Rinard, Martin, *Precise and Comprehensive Provenance Tracking for Android Devices*, Technical report, MIT/CSAIL, October 2019, https://hdl.handle.net/1721.1/122968

[4] Cito, Jürgen, Rubin, Julia, Stanley-Marbell, Phillip, and Rinard, Martin, "Battery-aware Transformations in Mobile Applications," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ACM, New York, NY, USA, ASE 2016, pp. 702–707, URL http://doi.acm.org/10.1145/2970276.2970324

[5] Shen, Jiasi and Rinard, Martin C., "Using Active Learning to Synthesize Models of Applications That Access Databases," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ACM, New York, NY, USA, PLDI 2019, pp. 269–285, URL http://doi.acm.org/10.1145/3314221.3314591

[6] Rinard, Martin C., Shen, Jiasi, and Mangalick, Varun, "Active Learning for Inference and Regeneration of Computer Programs That Store and Retrieve Data," in *Proceedings of the 2018 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, ACM, New York, NY, USA, Onward! 2018, pp. 12–28, URL http://doi.acm.org/10.1145/3276954.3276959

[7] Enck, William, Gilbert, Peter, Chun, BG, and Cox, LP, "TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones," in *OSDI*

[8] Sun, Mingshen, Wei, Tao, and Lui, John C.S., "TaintART: A Practical Multi-level Information-Flow Tracking System for Android RunTime," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, New York, NY, USA, CCS '16, pp. 331–342, URL http://doi.acm.org/10.1145/2976749.2978343

[9] Bell, Jonathan and Kaiser, Gail, "Phosphor: Illuminating Dynamic Data Flow in Commodity Jvms," in *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, ACM, New York, NY, USA, OOPSLA '14, pp. 83–101, URL http://doi.acm.org/10.1145/2660193.2660212

[10] LLC, KryptoWire, "Kryptowire Discovers Mobile Phone Firmware that Transmitted Personally Identifiable Information (PII) Without User Consent or Disclosure," https://www.kryptowire.com/adups\s\do6(s)ecurity\s\do6(a)nalysis.html, 2016

[11] Stavrou, Angelos, personal communication

[12] Android, "Using Binder IPI," https://source.android.com/devices/architecture/hidl/binder-ipc, 2017

[13] Cousot, Patrick and Cousot, Radhia, “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints,” in Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *POPL*, ACM, pp. 238–252

[14] Karr, Michael, “Affine Relationships Among Variables of a Program,” *Acta Inf.*, **6**, 1976, pp. 133–151

[15] Cousot, Patrick and Halbwachs, Nicolas, “Automatic Discovery of Linear Restraints Among Variables of a Program,” in Alfred V. Aho, Stephen N. Zilles, and Thomas G. Szymanski, editors, *POPL*, ACM Press, pp. 84–96

[16] Balakrishnan, Gogul and Reps, Thomas W., “DIVINE: DIscovering Variables IN Executables,” in Byron Cook and Andreas Podelski, editors, *VMCAI*, Springer, volume 4349 of *Lecture Notes in Computer Science*, pp. 1–28

[17] Corporation, Pendragon Software, “CaffeineMark 3.0,” `http://www.benchmarkhq.ru/cm30/`, 2016

[18] ADUPS, “adups fota,” , 2016, URL `http://www.adups.com/index.php`

[19] Kryptowire, “Kryptowire Discovers Mobile Phone Firmware That Transmitted Personally Identifiable Information (PII) Without User Consent Or Disclosure,” , 2017, URL `https://www.kryptowire.com/adups\s\do6(s)ecurity\s\do6(a)nalysis.html`

[20] Apuzzo, Matt and Schmidt, Michael S., “Secret Back Door in Some U.S. Phones Sent Data to China, Analysts Say,” *New York Times*, November 2016

7.0 List of Symbols, Abbreviations, and Acronyms

AOSP	**Android Open Source Platform**	**1, 3, 5, 56, 107**
APT	Advanced Persistent Threat	iii, 7, 107
ART	Android Runtime	i, 3, 4, 6, 20, 21, 54, 55, 56
CDM	Common Data Model	iii, 81, 98, 100, 101, 103
CTS	Compatibility Test Suite	i, 5, 60
JNI	Java Native Interface	4, 40, 41, 55, 57
PII	Personally-identifiable Information	106
TC	Transparent Computing	2, 3, 73, 81