AFRL-IF-RS-TR-2007-133
Final Technical Report
May 2007

EMBEDDED STATISTICAL PROFILING

Oasis Systems, Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-IF-RS-TR-2007-133 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
W. JOHN MAXEY Work Unit Manager	WARREN H. DEBANY, Jr. Technical Advisor, Information Grid Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2007	Final	Jan 05 – Jan 07

4. TITLE AND SUBTITLE

EMBEDDED STATISTICAL PROFILING

5a. CONTRACT NUMBER
FA8750-05-C-0022

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Michael Corley

5d. PROJECT NUMBER
4519

5e. TASK NUMBER
AK

5f. WORK UNIT NUMBER
03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Oasis Systems, Inc.
81 Hartwell Ave.
Lexington MA 02421

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFGB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-133

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-225

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Embedded Statistical Profiling is an initiative to support both offensive and defensive computer network applications. The basis for the effort comprises the development of an environment framework known as Simplified Protocol Capture (SIMPCAP). The environment comprises a collection of tools manifested from SIMPCAP including a SQLite based packet querying engine, virtual file abstraction for seamless multi-file processing, and a complete packet capture / decoding Application Programming Interface (API). Together, these tools automatically integrate with existing LIBPCAP based tools, resulting in a highly tunable and robust environment for 1st level and 2nd level forensic analysts working in network centric operations.

15. SUBJECT TERMS
SQLite, SIMPCAP, Statistical Profiling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			W. John Maxey
U	U	U	UL	17	19b. TELEPHONE NUMBER (*Include area code)*

Table of Contents

Introduction

This is the final status report in support of AFRL Embedded Statistical Processing (ESP). It is submitted in accordance with the deliverable requirements. This report describes the research initiatives pursued and associated software toolkit deliverables.

Section I (Overview)

Examining large quantities of network traffic data for statistical purposes is a difficult task. Packets of indeterminate size hurtling by at thousands to millions of packets per second (real time), or multi-gigabyte collection files containing millions to billions of packets to analyze provide a rich opportunity for streamlining and fine-tuning the analysis process. In the current (circa 2006) best practice approaches for large data-volume analysis, the normal strategy of throwing more horsepower at the problem seems to be a very sub-optimum one. A more successful approach has been to refine the goal to very specific statistic or objective, and then quickly sort through the data once to get that information for display or further analysis.

This approach works reasonably well when real-time and relatively simplistic results are the requirement, as in the case of the first large class of examination/monitoring functions, the real-time information assurance function - an alerting function that warns of immediate threats or problems to the network under observation. This is the case with the standard operations floor in regional network control centers (NCCs), where operators have a small set of screens to watch and react when some automated function turns an indicator from green to yellow (or red; sometimes accompanied by a klaxon or siren). Dedicated tools sifting through traffic data to locate specific trend changes (traffic rates, particular signature occurrences, catalogued patterns of behavior that trigger some alert) are ideal for this environment and provide the current best protection at the front end of network presence presented to the Internet. In the case of interest for this task, though, the streamlining approach does not provide sufficient flexibility to allow successful prosecution of the post-processing task of finding more subtle events in the network traffic stream. In the post-processing domain, an iterative approach is usually required prior to isolating and explaining appropriately an event of interest from the traffic samples. There is still, however, a keen interest in finding these subtle events in as timely a fashion as possible. This is the central problem of interest we investigated in this sub-task. Primary measures of effectiveness included speed and storage space requirements necessary to perform various standard network security analysis tasks under the constraint of large analysis datasets.

There are two conceptual approaches to post-processing network data. First, appropriate for smaller networks or capture sizes, is the development of algorithms that operate on the data directly each time a statistic or characteristic is to be calculated. This provides minimum overhead for storage of data – just the cost/space for the network capture itself – but maximizes the processing time per statistic desired, as the data has to be processed each time a statistic is desired. Further, given the current typical network tool based on LIBPCAP, the network data has to be processed serially from the beginning of the capture to the end to locate the desired packet structures to build the statistic.

The second approach is to pass through the network data once, and during that process calculate as many statistics as possible and store that information for later observation and use. This provides (ideally) the minimum processing time for the data, but maximizes the storage space necessary for the generation of the statistics. In addition, this approach assumes that the desired statistics (at least some large portion of them) are known in advance.

For both of these approaches, as data volumes increase, performance falls off rapidly. For the first case, the requirement to serially process through the data set each time a statistic is gathered becomes problematic, often lasting many minutes for each statistic. For the second case, the multi-dimensional aspect of many of the statistics (for example, time-to-live and time stamp versus source and destination IP address) multiplies storage requirements significantly such that only very typical statistics are gathered because of limitations in storage and retrieval of the calculated data.

In either case, the ability to rapidly find anomalous behavior in large network captures is limited significantly. Solutions to this problem can be summarized generically as follows:

1. Process the archived network data faster
2. Save collected data/statistics in less space
3. Retrieve and compile the statistics more efficiently

Additionally, the interactive nature of the analysis process needs to be incorporated into the optimization process; an analyst needs to interact with the network data and statistics in a heuristic fashion with minimal distracting tasks along the way to finding the statistic or characteristic of interest. Finally, the reporting and consolidation of an analysis session's results needs to be appropriate for rapid dissemination and understanding. The task of translating the detected anomaly into a scheme suitable for incorporation into the first-line alerting function (single-pass speedy processing of a particular anomaly) is not addressed.

Solution alternatives and the SQLite engine

It is fairly easy to say "just process the data faster" as one way of speeding up the anomaly detection process, but not quite as easy to actually put that theory into practice. There are too many "speeds" to take into account. First, we are trying to detect a process occurring in the midst of normal network traffic, which in itself is not very well behaved, so the dividing line is not usually clear between normal and abnormal traffic. The speed then is dependent on the variety and types of features that can be used to describe the differences in the traffic characteristics which would allow the detection and isolation of anomalous behavior. There are easily dozens of common protocols, each with possibly dozens of values of interest in just the header content, and also with an indeterminate number of values/features available in the data portion of each packet, leading to numbers of potential features well into the multiple hundreds. Trying to get a contemporary pattern recognition tool to stabilize with a few tens of features is possible, but several hundreds of features are currently not tenable. Then, the "speed" of the algorithms is dependent on the highest dimensionality of the set of data to be analyzed. If, for instance, we are comparing a small enclave (say, a class C network) to a large enclave (a class B

network), then the possible interactions between a large set of outside addresses (for illustration purposes, a class B network) with each enclave would have to track (2^8*2^16), or 2^24 possible interactions for the smaller network and (2^16*2^16), or 2^32 possible interactions for the larger class B network (or roughly 10^23 to 10^31 relations). In real scenarios, the external activity is somewhat more restricted than a full class B network, but in one example enclave with a few hundred hosts across five class C subnets, the average interactions across internal and external addresses (as a two-dimensional x-y chart) still had on the order of 10^5 interactions over the course of a few days. In the example data provided for cross-domain traffic (between .mil and .com domains), there were on the order of 10^7 interactions occurring in a very limited time set of data. The very real implication is that a general data management approach to large-volume captures would need to routinely handle feature set interactions on the order of 10^3 features by 10^5 interactions, or 10^8 observation types for a small network, and potentially up to 10^20 for large network entities, such as cross-domain routes. With this order of magnitude in mind, we now look at the possible ways to make the problem more feasible. Process Archived Network Data Faster. Just adding CPU horsepower is not a clear advantage; the volume of the active memory space required (10^3 for small enclave IP addresses, by 10^5 for Internet IP addresses, by 10^3 features to track) could easily be multiple gigabytes for a small to moderate enclave, and that is before any statistical calculations take place. Then, the current best-practice tool basis (LIBPCAP) processes files serially, from top to bottom, each time a statistic or feature is desired. Processing a large file faster may not be as important as processing the right part of the file at the right time. To this end, the SIMPCAP/SQLite toolset (included as deliverables with this report) provides a capability to perform SQL queries over network traffic and other disparate (and proprietary formatted) data repositories. These include but are not limited to data repositories containing Snort/Firewall logs, WHOIS internics, radio antenna tower data, and any other form of binary and/or text data semantically appropriate for analysis.

The SQLite extensions for performing network capture queries are accomplished using the SIMPCAP engine, which is the basis for interfacing with network trace files and virtual file system. This basic pair of tools can combine to perform the following valuable functions:

1. Gather statistics on particular time groups in a file, such as looking for the least-frequent IP pairs in the highest density time periods in a file (to look for the clever “needle in the haystack” intruder).
2. Compile statistics on repetitive time periods so that a bias-removal approach uses less data overall but still achieves a reasonable performance.
3. Provide a capability to parallel process an archive (or group of archives) to achieve much higher throughput than traditional libpcap approaches.

The SQLite toolkit engine is comprised of an architectural modification to an open source SQL querying engine in which the grammar parser component has been extended to include an “import” keyword. The import keyword empowers the analyst with a means to describe the semantic nature and format layout for virtually any type of file format. SIMPCAP is used for the component to interface with network trace files and virtual file subsystem.
This implies that the analysis can very readily issue queries over libpcap formatted trace files and virtually any other type of textual or binary data. Example scenarios for

effective system usage are present in the SQLite manual (included along with this report).

Section II. Data-set Collection

A distributed data-set repository has been constructed for conducting analysis. These constitute data-sets made available through other AFRL efforts in addition to local enclave collections performed on demand. The repository includes network traffic from two operational military sectors, NEADS (North East Air Defense Sector) and NSIRC (National Security Incident Response Center). The NEADS traffic is composed of approximately seventeen (17) days of complete twenty-four hour traffic segments. This data is pertinent for testing of protocol level activity as expected in enterprise and enclave networks. The NSIRC traffic is composed of approximately five-hundred (500) segments averaging about one-half second (1/2 sec.) per file. The time segment for the NSIRC traffic totals to approximately five (5) minutes. This is clearly a much larger traffic volume and will be pertinent to testing of protocol level actively as expected in Wide Areas Nets (WANs) and World-Wide Distributed (WWW) Nets. In addition, this set is the basis for testing and analysis for development of the management system for large order traffic volumes. The NEADS data-set totals in size to approximately 8.5 gigabytes and the NSIRC data-set totals in size to approximately 5 gigabytes. In addition, local collections from the NGCS (Next Generation Cyber Security) Lab have been performed for additional support of testing. This data was generally captured as needed, and also provided the testing point for live network processing. In all cases, the data is unclassified and uploaded to the NGCS NAS (Network Attached Storage) Server.

Section III. Data-set Management System

Extensive effort into various design processes for high volume data set management techniques have led to the development of a Virtual File System (VFS), which enables an analyst to logically relate, and sift through multiple disparate data-sets by building virtual (logical) representations that describe user specified criterion relationships between the data-sets. Using this approach, analysts interact with the virtual file in very similar fashion to standard processing.

The virtual file is a SIMPCAP based facility that enables the analyst to specify filters and extract statistics and potentially common attributes across very large disparate sets of network traffic. The major features include but are not limited to time based sampling, packet density and data rate profiling, BPF filtering, and intra-file overlap detection. Time based sampling enables a user to sort, and reconstruct and analyze files over specified time segments. Within a large scale network environment, there often arises scenarios in which there are many capture files that are either too large or so small they can't be managed effectively with current tools. A good example is when a forensic analyst has a repository of three-hundred (300) disparate capture (trace) files, and he or she knows very little about the attributes. With currently available tools the analyst would likely perform analysis on each individual file, packet-by-packet (as with ethereal, TCP-dump, etc.), or perhaps even create a labor intensive batch process that performs a singe rigid operation. With the virtual file utility, the analyst need only specify the files of interest and the criteria desired such as a BPF filter and/or a sample interval, etc. Figures 1 and 2 illustrate the typical usage of the utility.

The structure of the virtual file is comprised of a header and a body. The header structure contains information pertaining to sampling rate, number of samples per file, and all of the LIBCAP file(s) processed for inclusion into the virtual file. In addition, the header stores certain basic attributes about each file. The body is comprised of timestamp versus byte offset values for each file, where the values are determined based on the sampling rate and resolution. This is handy in cases where analysis requires rapid look up of arbitrary time segments, or some type of parallel processing or handling of packet content in one or more contexts. As a more general utility, the virtual file provides pointer access to standard capture file in a distributed storage system. That is, LIBPCAP files can be scattered locally on a single host or on many hosts over a large scale network. The virtual file abstraction provides seamless access to all files as if a single capture entity. This option depicted by the "-v" option in Figure 2. Intra-file overlap detection provides a means to discover overlapping time segments between files. This is often useful for sorting out data-sets from large scale distributed WANS in cases where the analyst may know little or nothing about individual file attributes. Overlapping files may indicate traffic from separate sensors. In such a case, it would useful to specify a BPF filter to extract traffic that has common attributes in some context. One example might include filtering all traffic originating from a particular subnet, IP or set of IP addresses. This feature is illustrated in Figure 2 by "-f" option. Overlapping files may indicate data redundancy, human error in the collection process, and perhaps even a valid anomaly. In any such case, uncertainties should be caught prior to analysis so other statistical measures are not skewed leading to potential false alarms. Data rate and packet density profiling are options that provide statistics for the actual data rate and number of packets observed on a per sample interval basis, and for the entire analyzed set of files. These figures are very common indicators used for determining network status.
Additionally, the output facilities for these options export formatted text which provides sufficient flexibility for visualization with well known packages as Excel or GNU plot. Figure 2 illustrates the command line invocation of the win32 executable. Listed is a print out of all options and associated command line usage.

Section IV. Statistical Profiling

Determining network status for anomalous behavior is an ever changing model. Computer networks ranging from distributed world-wide nets (WANs) to small enclaves are always evolving and therefore the status from the perspective of the analyst changes as well. Everyday new machines and devices join and are removed, and new services (such IP telephony) are constantly emerging while old servicing are upgraded or go offline. The rate of change and dynamic nature of content in these systems make it especially difficult to formulate an effective model for determining normal versus anomalous behavior. Because of these limitations, this effort has chosen not to provide a specific targeted set of applications to analyze specific scenarios. For instance, suppose a suspected attack was of an IP SPOOF class, or DDOS attack which made use of vulnerabilities in a newly deployed IP telephony service. Such a class of DDOS attacks often has a completely different set of signatures from the former, rending previous detection capabilities virtually useless. This effort has therefore sought to provide the analyst with a toolkit that provides flexible and simplistic, yet powerful facilities for custom pluggable analysis in an ever changing environment. Figure 5 illustrates the basic concept of pluggable analysis.

Section V. SIMPCAP Extension Architecture

SIMPCAP has been integrated with LIBPCAP comprising a super-set of the native API. The current version is 8.0, and is a candidate for an open source release. The latest edition to the system incorporates facilities for random seeking and modified binary searching for LIBPCAP based Ethernet save-files. The integration enables SIMPCAP facilities to directly interact with the state of LIBPCAP run-time primitives. This was a requirement to achieve true random access. The fundamental component enabling non-linear access (developed in an earlier version of SIMPCAP) is called the packet detection engine (PDE). The PDE exports two parameters to all client derived facilities. The first is a packet criteria specification. This enables the user to specify the number of static packet field attributes used for identification of packets. The second is a confidence figure that is passed to indicate the number of static fields that must be matched for a successful identification. These parameters are intended for future use in appropriately handling malformed and/or corrupted packets, and also for adjusting to system performance needs. Figure 4 illustrates the basic hierarchy.

Section VI. Figures

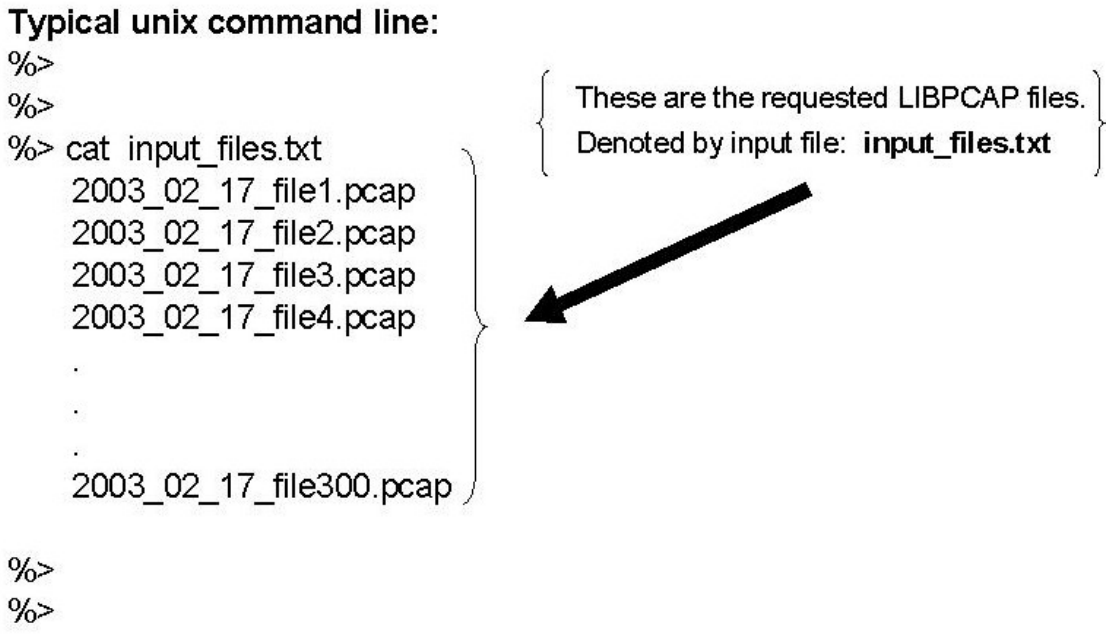

Figure 1: Virtual File Structure

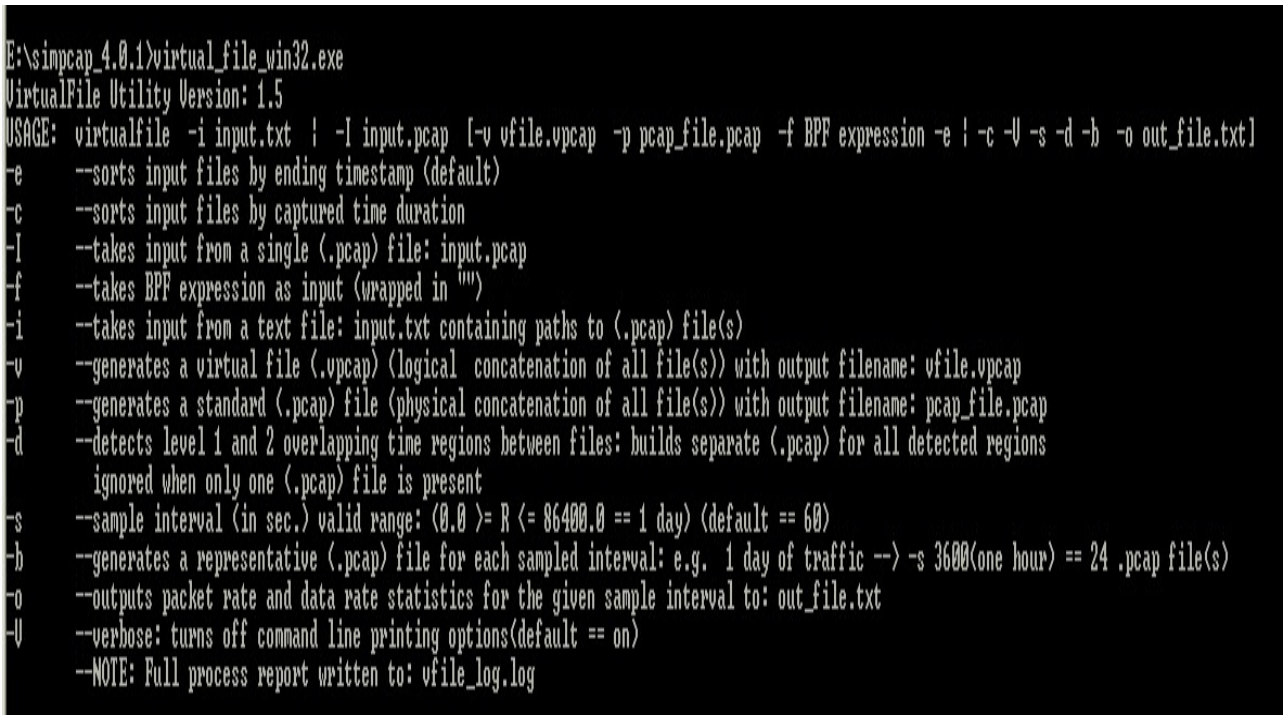

Figure 2: Win32 Virtual File Utility Executable

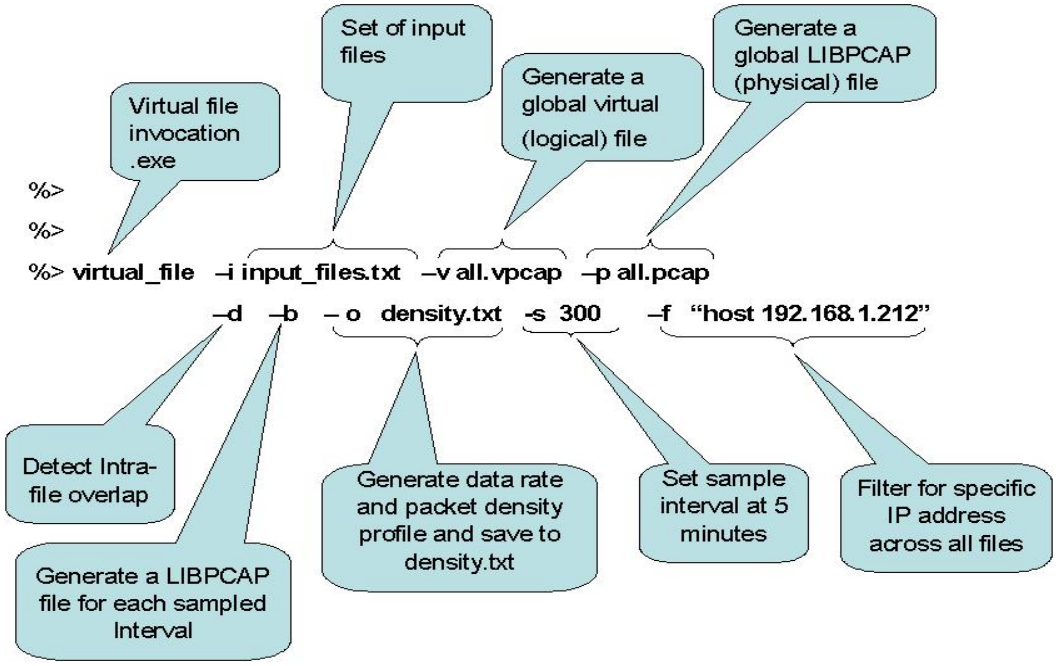

Figure 3: Virtual File Utility (Example Usage)

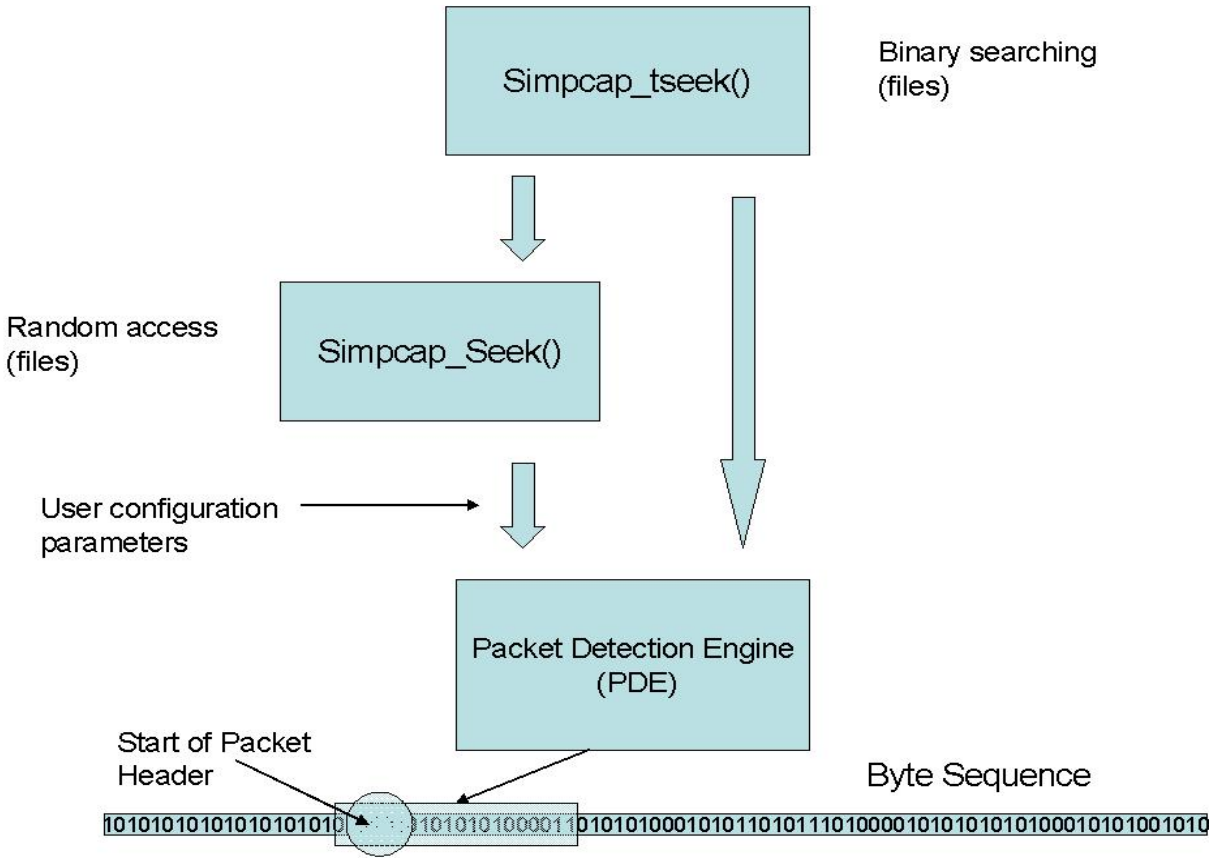

Figure 4: SIMPCAP Extension Architecture

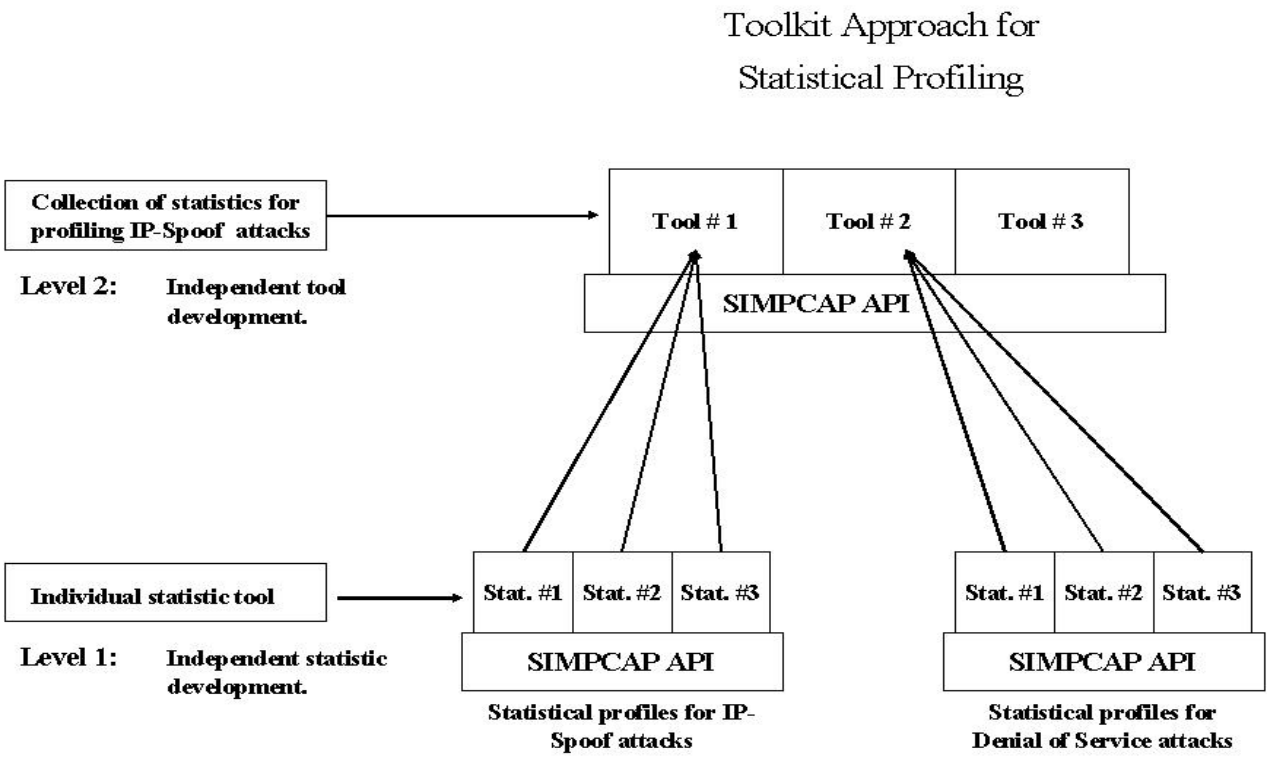

Figure 5: Pluggable Toolkit

As shown above, in level one (1) the user defines custom SIMPCAP scripts tailored for specific analysis. In level two (2), the user combines multiple individual tools, as developed in level 1, to comprise a suite of tools for analyzing an entire class of anomaly scenarios. As in the example above, the user developed a toolkit for a variety of disparate DDOS and IP SPOOF class attacks.

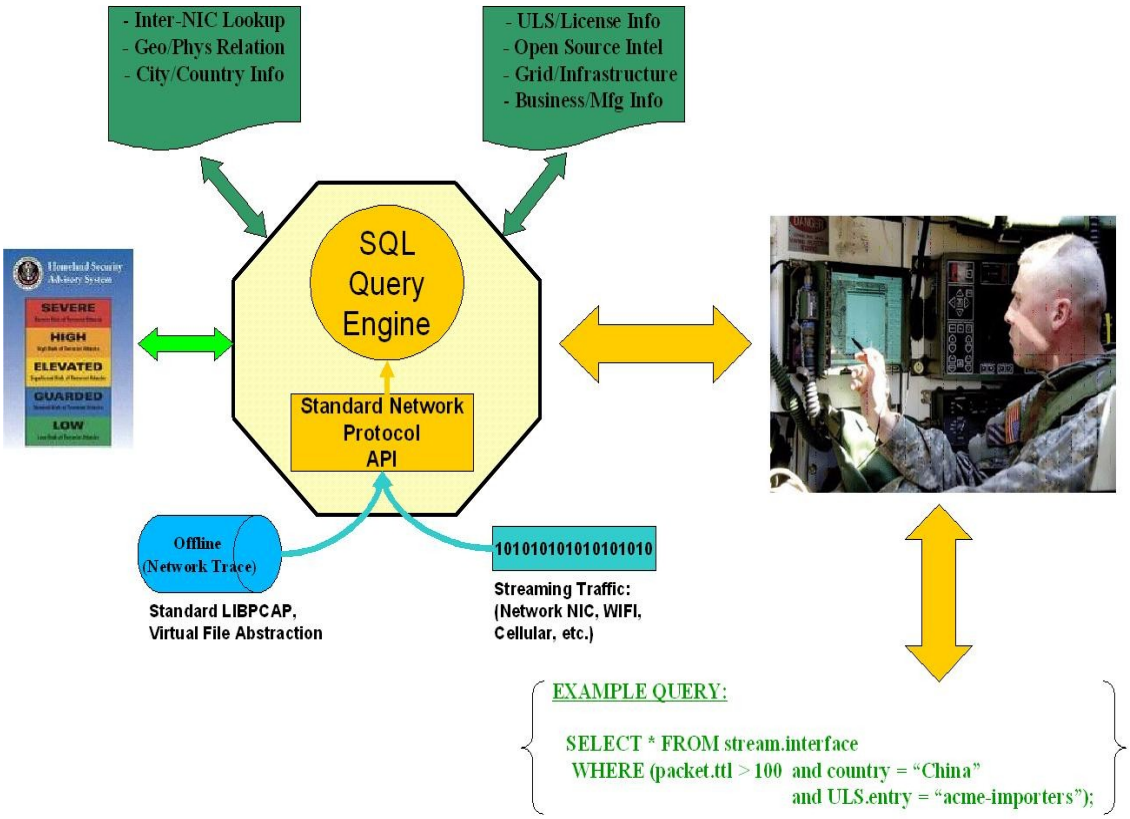

Figure 6: SQLite Concept

Appendix A.

1) Figure List

2) Definitions

NCC - Network Control Center
LIBPCAP - standard packet capture driver
IP - Internet Protocol
SQLite - open source, light weight query engine extended to handle network packet based queries.
SIMPCAP - (Simplified Protocol Capture) - a set of extensions to the standard packet capture driver, adding the capability for automated protocol decoding, non linear capture, and virtualization of the packet capture file apparatus
WHOIS - regional network registry for managing internet protocol address allocations.
CPU - Central Processing Unit
NEADS - North East Air Defense Sector
NSIRC - National Security Incident Response Center
AFRL - Air Force Research Laboratory
WWW - World wide web
NGCS - Next Generation Cyber Security
NAS - Network Attached Storage
VFS - Virtual File System
BPF - Berkeley Packet Filter
WAN - Wide Area Network
DDOS - Distributed Denial Of Service
API - Application Programming Interface
PDE - Packet Detection Engine

3) List of Deliverable Software and Users Manuals

All Software is located on the enclosed CD. Please view file README.txt on the CD for complete detail

1) SIMPCAP (Simplified Protocol Capture) toolkit
    1.1) Simpcap Users Manual: file: simpcap_manual.txt

2) SQLite (SQL query engine) toolkit
    2.1) SQLITE manual: file: sqlite_manual.txt