

AFRL-IF-RS-TR-2005-394
Final Technical Report
December 2005



ANOMALY DETECTION IN DISPARATE COMPUTER NETWORKS

ACS Defense, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

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

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-394 has been reviewed and is approved for publication

APPROVED:

/s/
ANDREW J. KARAM
Project Engineer

FOR THE DIRECTOR:

/s/
WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2005	3. REPORT TYPE AND DATES COVERED Final May 04 – Nov 05	
4. TITLE AND SUBTITLE ANOMALY DETECTION IN DISPARATE COMPUTER NETWORKS			5. FUNDING NUMBERS C - FA8750-04-C-0039 PE - 61102F PR - 2311 TA - 00 WU - 01	
6. AUTHOR(S) Michael J. Corley				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) ACS Defense, Inc. Burlington MA 01803			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFGB 525 Brooks Road Rome NY 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2005-394	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Andrew J. Karam/IFGB/(315) 330-7290 Andrew.Karam@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT <i>APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.</i>			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) Two SIMPCAP based statistical applications for monitoring network status. Both are in the form of independent executable files. The first provides real-time and post mortem capability for visualizing in-bound and out-bound bandwidth usage. User selectable parameters include the network IP address to monitor, a flag to distinguish between real-time or off-line processing, and a field for selecting the sampling material interval for processing with RRD-tool. The second tool provides a capability for measuring in-bound and out-bound traffic by protocol for using selectable time intervals.				
14. SUBJECT TERMS SIMPCAP, anomaly, disparate, RRD-tool			15. NUMBER OF PAGES 16	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

1.0	Large Dataset Handling Research (Overview).....	1
2.0	Distributed Data-set Collection	4
3.0	Large Dataset Management System	4
4.0	Determining Network Status and Visualizations	7
5.0	SIMPCAP Extension Architecture	11

List of Figures

Figure 1: Multi-file Input.....	5
Figure 2: Example Virtual File Utility Usage	5
Figure 3: Virtual File Command Line Invocation	7
Figure 4: Bandwidth Monitor	9
Figure 5: Packets per Second (PPS by Protocol).....	10
Figure 6: Pluggable Toolkit	11
Figure 7: Packet Injection Engine (PDE)	12

1.0 Large Dataset Handling Research (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 2005) 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:

- Process the archived network data faster
- Save collected data/statistics in less space
- 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 “state space” representation concept

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 toolset includes an ability to search arbitrarily for packets of interest within large capture files, and this is paired with a secondary file which is built for each capture file relating file offset with time of capture within the file. This basic pair of tools can combine to perform the following valuable functions:

- 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).
- Compile statistics on repetitive time periods so that a bias-removal approach uses less data overall but still achieves a reasonable performance.
- Provide a capability to parallel process an archive (or group of archives) to achieve much higher throughput than traditional libpcap approaches.

Save Collected Data/Statistics in Less Space. The secondary storage necessary to evolve time-dependent statistics could be quite large. Longer-term statistics that show trends and provide for trend-removal and bias-removal functions to improve anomaly detection performance take up additional large volumes of secondary storage space. Raw secondary storage is actually pretty cheap. The real cost of the large data sets is not in the raw storage space itself, but in the timely access to the right data at the right time. Any compression or reduction in complexity of the network data needs to be paired with an ability to efficiently retrieve it and make use of it.

2.0 Distributed 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 of approximately 8.5 gigabytes and the NSIRC data-set totals in size of 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.

3.0 Large Dataset 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 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, 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 cannot 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 single 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.

Typical unix command line:

```
%>  
%>  
%> cat input_files.txt  
2003_02_17_file1.pcap  
2003_02_17_file2.pcap  
2003_02_17_file3.pcap  
2003_02_17_file4.pcap  
.  
.  
.  
2003_02_17_file300.pcap  
  
%>  
%>
```

These are the requested LIBPCAP files.
Denoted by input file: **input_files.txt**

Figure 1: Multi-file Input

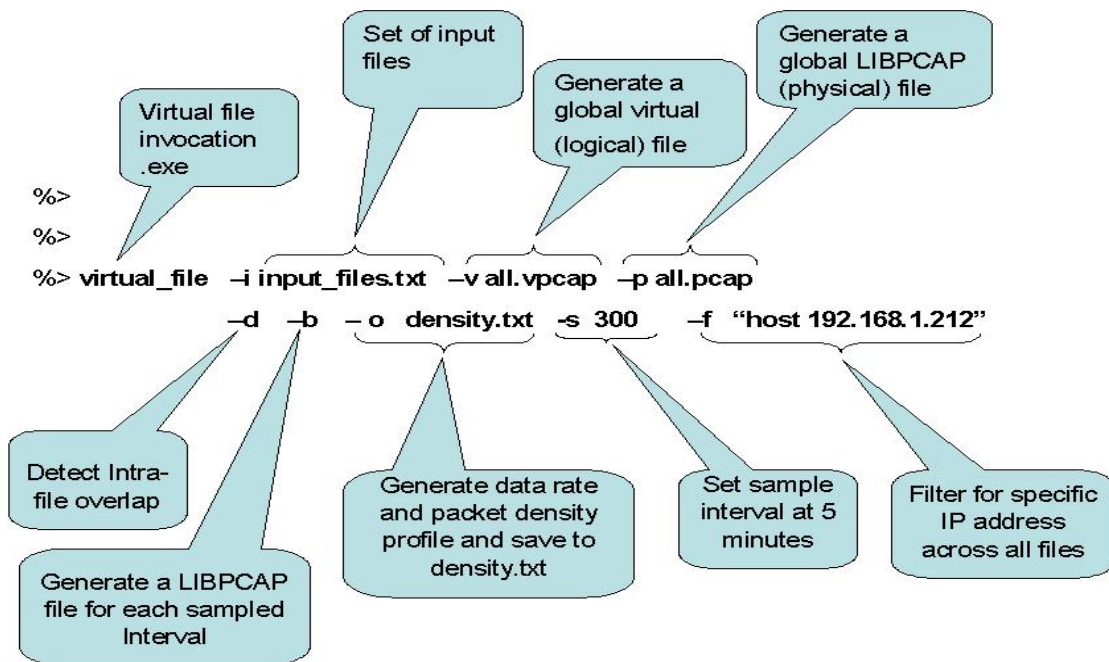


Figure 2: Example Virtual File Utility Usage

The structure of a 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 LIBPCAP files processed for inclusion into the virtual file. In addition, the header stores certain basic attributes about each file. The body of is comprised of timestamp versus byte offset values for each file where the values are determined based on the sampling rate and resolution of each file. This is handy in cases where analysis requires rapid look 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 files 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; and 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 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 file may indicate data redundancy, human error in the collection process, and perhaps even a valid anomaly. In any case, such uncertainties should be caught prior to analysis so other statistical measures 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 packages as Excel or GNU plot. Figure 3, (below) illustrates the command line invocation of the win32 executable. Listed is a print out of all options and associated command line usage.

```

E:\simpcap_4.0.1>virtual_file_win32.exe
VirtualFile Utility Version: 1.5
USAGE: virtualfile -i input.txt | -l input.pcap [-v vfile.vpcap -p pcap_file.pcap -f BPF expression -e | -c -U -s -d -b -o out_file.txt]
-e --sorts input files by ending timestamp (default)
-c --sorts input files by captured time duration
-l --takes input from a single (.pcap) file: input.pcap
-f --takes BPF expression as input (wrapped in "")
-i --takes input from a text file: input.txt containing paths to (.pcap) file(s)
-v --generates a virtual file (.vpcap) (logical concatenation of all file(s)) with output filename: vfile.vpcap
-p --generates a standard (.pcap) file (physical concatenation of all file(s)) with output filename: pcap_file.pcap
-d --detects level 1 and 2 overlapping time regions between files: builds separate (.pcap) for all detected regions
    ignored when only one (.pcap) file is present
-s --sample interval (in sec.) valid range: (0,0) = R (<= 86400,0 == 1 day) (default == 60)
-b --generates a representative (.pcap) file for each sampled interval: e.g. 1 day of traffic --> -s 3600(one hour) == 24 .pcap file(s)
-o --outputs packet rate and data rate statistics for the given sample interval to: out_file.txt
-U --verbose: turns off command line printing options(default == on)
--NOTE: Full process report written to: vfile_log.log

```

Figure 3: Virtual File Command Line Invocation

4.0 Determining Network Status and Visualizations

Two SIMPCAP based statistical utilities have been developed for monitoring and visualizing network status. Both are in the form of independent executable files. The first provides real-time and post mortem capability for visualizing in-bound and out-bound bandwidth usage. User selectable parameters include the network IP address to monitor, a flag to distinguish between real-time or offline processing and a field for selecting the sampling interval for processing with the RRD-tool. Figure 4 illustrates the command usage and corresponding visualization. The second tool provides a capability for measuring in-bound and out-bound traffic by protocol for using selectable time intervals. Figure 5 illustrates the command usage and corresponding visualization. Both tools taken together constitute the fundamental SIMPCAP pluggable framework for defining independent solutions for specific monitoring requirements. In this case, the user has the capability to compare in/out-bound bandwidth usage with in/out-bound protocol over arbitrary time segments.

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 too formulate an effective model for determining normal versus anomalous behavior. Because of these limitations, this effort as chosen not to provide a specific targeted set of applications to analyze specific scenarios. For example by

providing the tailored tools needed for identifying a particular class of DDOS (Distributed-Denial of Service) attacks. 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, rendering 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 analysis in an ever changing environment. SIMPCAP and its pluggable framework provide such a facility. Figures 4 and 5 illustrate the two basic examples of the SIMPCAP based analysis tools provided. Taken together, such custom tools comprise the basis for the pluggable toolkit framework. Figure 6 illustrates the concept.

```
Shell - Konsole
Session Edit View Bookmarks Settings Help

corleym[54]:
corleym[54]:
corleym[54]: gcc -o band_monitor example_11_rrd_band_monitor.c -lpcap
corleym[55]: band_monitor ~corleym/testdata/needs/01mar-out.dmp 156.30.0.0 255.255.0.0 0
895x672
corleym[56]:
```

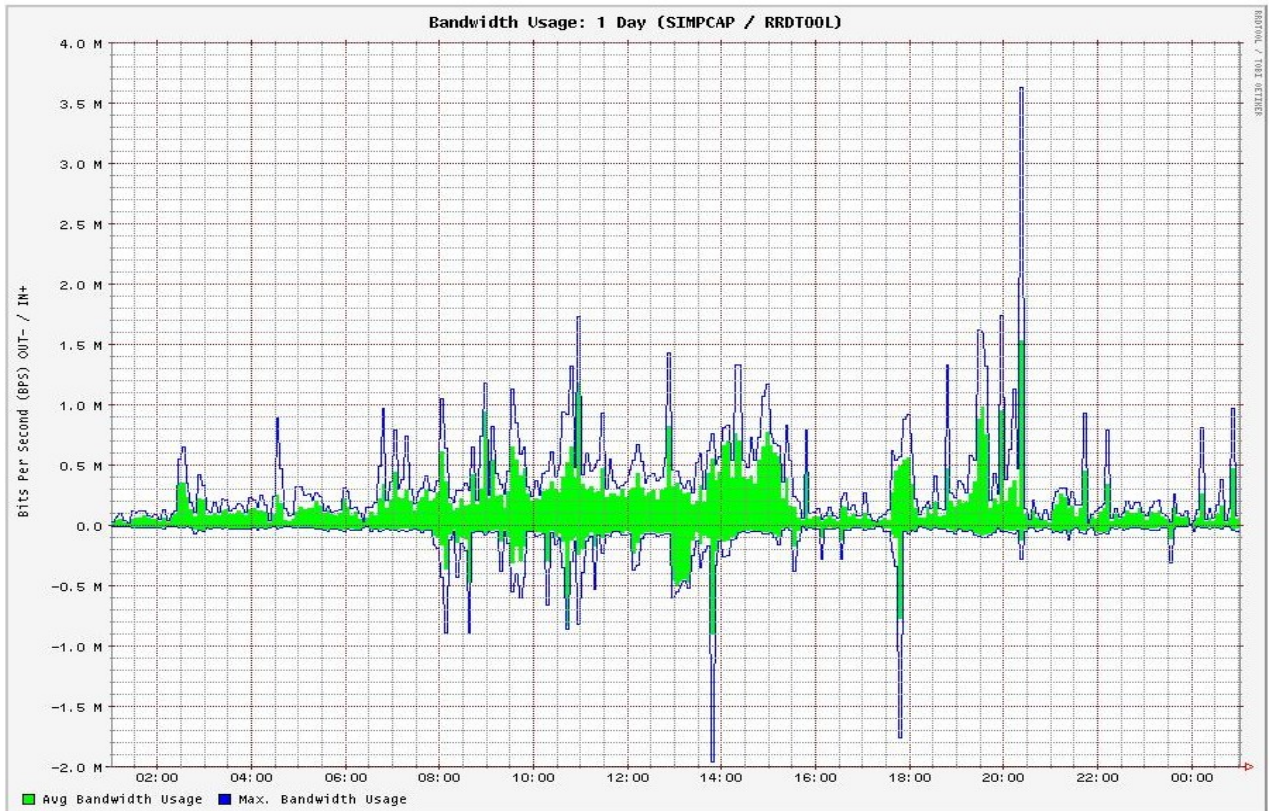


Figure 4: Bandwidth Monitor


```
Shell - Konsole
Session Edit View Bookmarks Settings Help

895x672
corley@79: gcc -o port_monitor example_12_rrd_band_by_protocol.c -lpcap
corley@80: port_monitor ~corley/testdata/needs/01mar-out.dmp 135.5.0.0 255.255.0.0 1
895x672
895x672
895x672
895x672
```

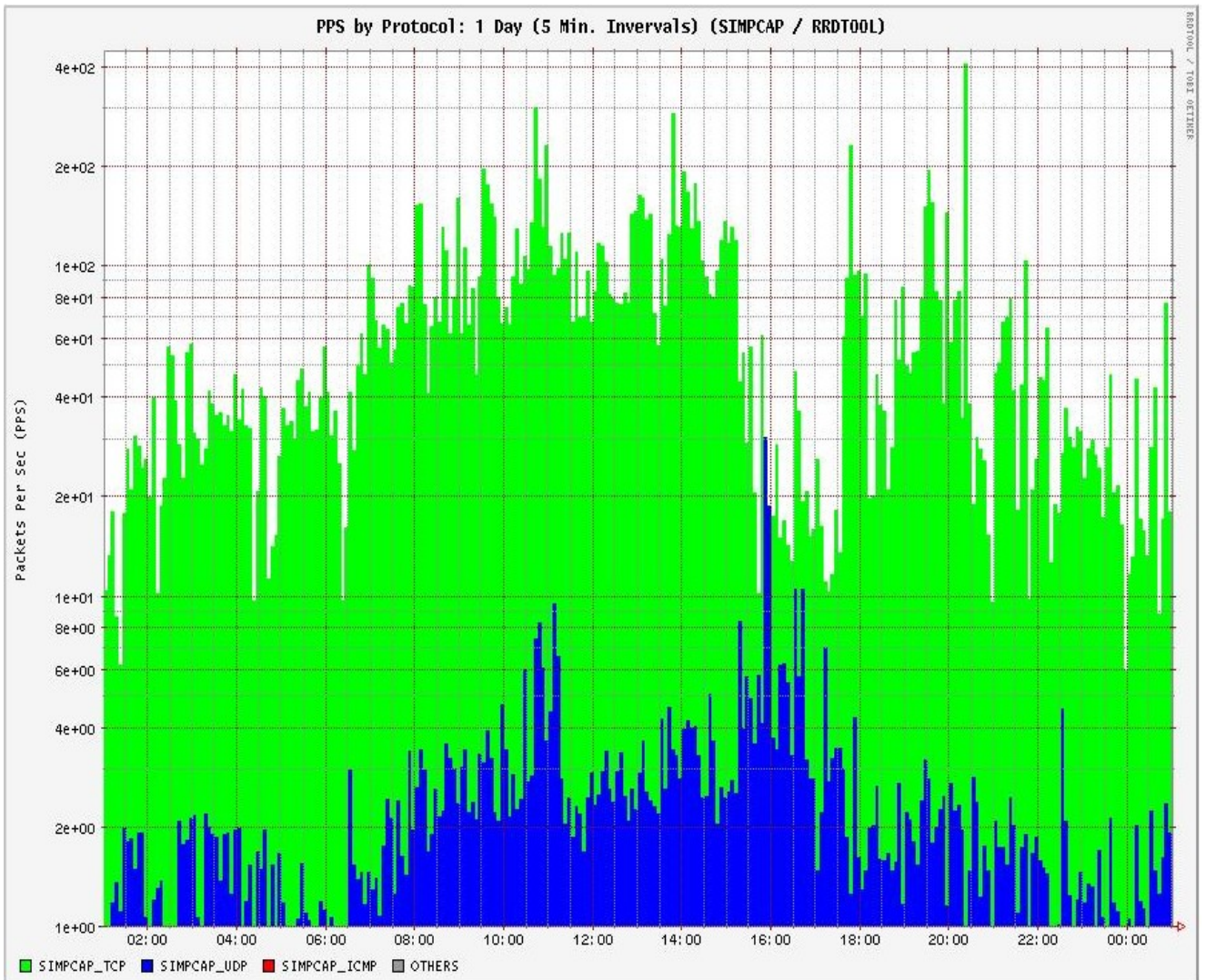


Figure 5: Packets per Second (PPS by Protocol)

Toolkit Approach for Statistical Profiling

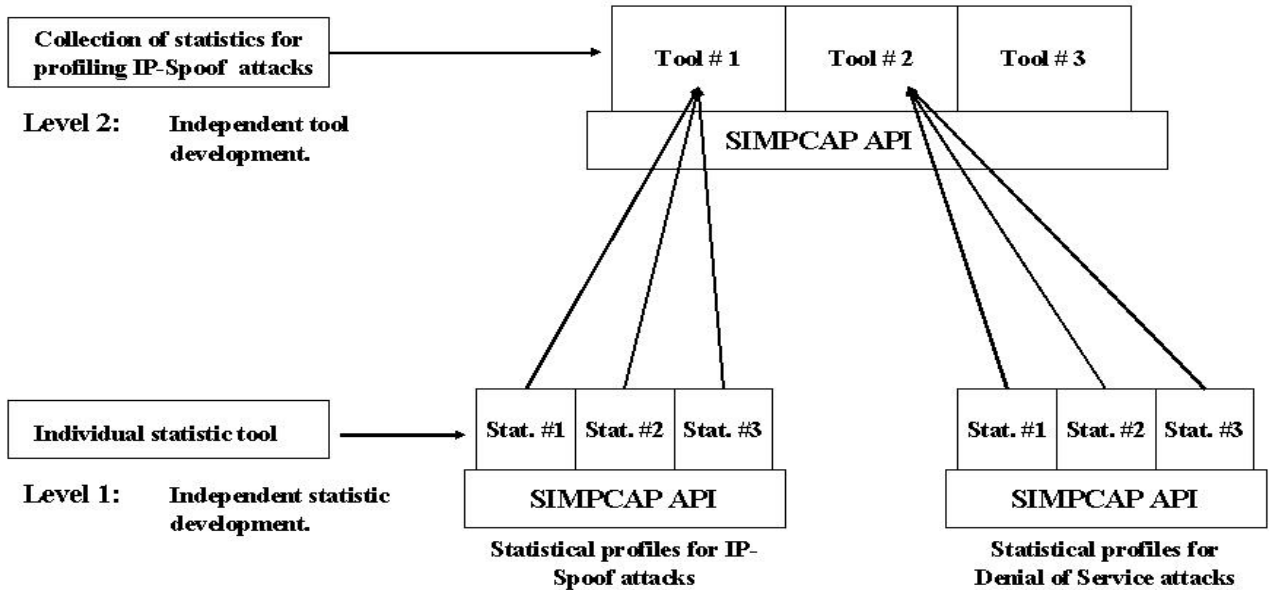


Figure 6: Pluggable Toolkit

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.

5.0 SIMPCAP Extension Architecture

SIMPCAP has been integrated with LIBPCAP comprising a super-set of the native API. The current version is 7.1 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 in 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 7 illustrates the basic hierarchy.

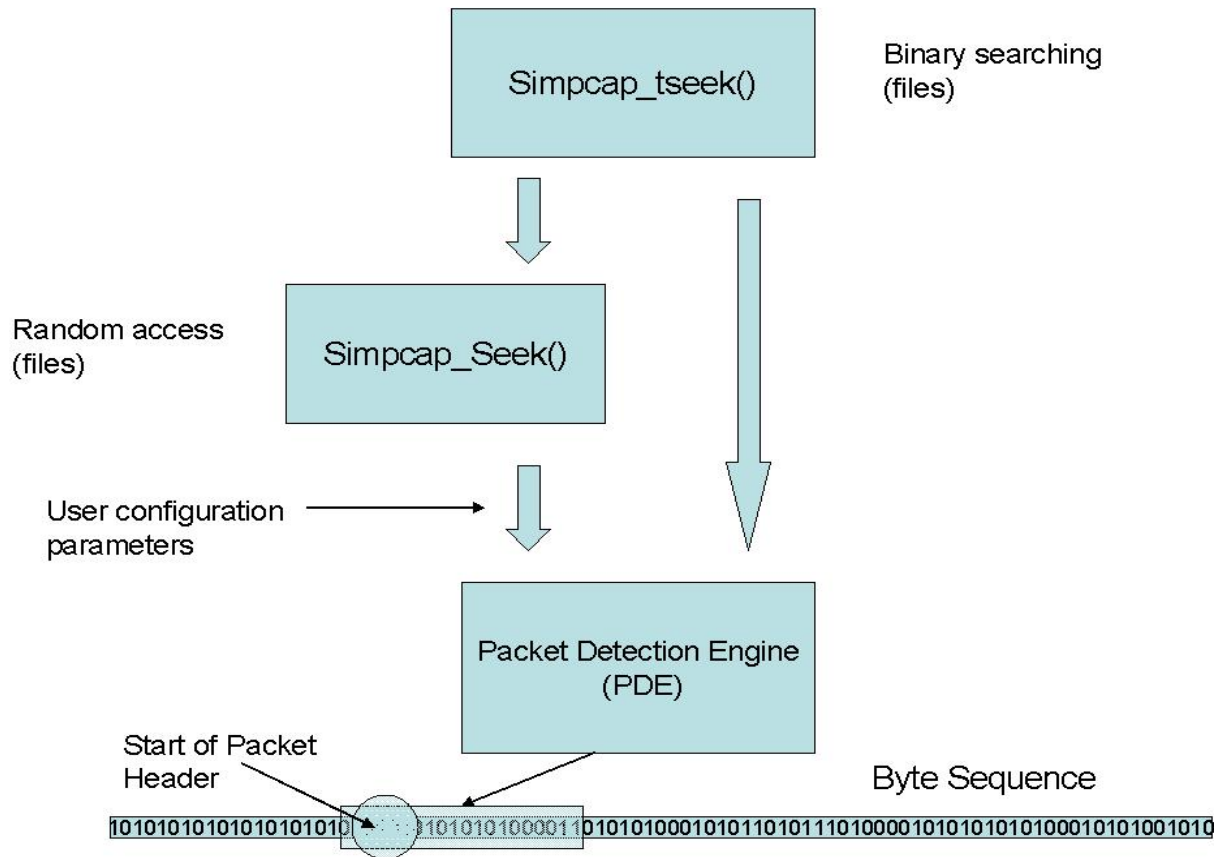


Figure 7: Packet Injection Engine (PDE)

As illustrated, the PDE directly enables non-linear seeking (Simpcap_Seek()) , and a combination of simpcap_seek() and the PDE provides non-linear searching. All client designs need only inherit from simpcap_tseek() to suit custom time based processing needs.