Recent Advances in Network Intrusion Detection System Tuning

Joel Sommers
University of Wisconsin-Madison
jsommers@cs.wisc.edu

Vinod Yegneswaran
University of Wisconsin-Madison
vinod@cs.wisc.edu

Paul Barford
University of Wisconsin-Madison
pb@cs.wisc.edu

***Abstract*— We describe a traffic generation framework for online evaluation and tuning network intrusion detection systems over a wide range of realistic conditions. The framework integrates both benign and malicious traffic, enabling generation of IP packet streams with diverse characteristics from the perspective of (*i*) *packet content* (both header and payload), (*ii*) *packet mix* (order of packets in streams) and (*iii*) *packet volume* (arrival rate of packets in streams). We begin by describing a methodology for benign traffic generation that combines payload pools (possibly culled from traces of live traffic) with application-specific automata to generate streams with representative characteristics. Next, we describe a methodology for malicious traffic generation, and techniques for integration with benign traffic to produce a range of realistic workload compositions. We realize our traffic generation framework in a tool we call Trident, and demonstrate its utility through a series of laboratory-based experiments using traces collected from our departmental border router, the DARPA Intrusion Detection Evaluation data sets provided by Lincoln Lab, and a suite of malicious traffic modules that reproduce a broad range of attacks commonly seen in today's networks. Our experiments demonstrate the effects of varying packet content, mix, and volume on the performance of intrusion detection systems.**

I. Introduction

Malicious traffic in the Internet is growing at an alarming rate both in terms of volume and diversity. This motivates the need for methods and tools that can be used to assess and tune the capabilities of network intrusion detection systems (NIDS) to a wide range of both malicious and benign traffic. Standard methods for NIDS testing include the use of canonical packet traces for *offline* tests or traffic generation systems for *online* evaluation in a controlled laboratory setting. Regardless of the approach, a standardized, comprehensive test suite for NIDS enables rule sets and configurations to be tuned to meet projected demands, and detailed comparisons between different systems.

The landmark work by McHugh [1] introduced a set of requirements for NIDS test traffic streams. A summary of these requirements is that tests must be conducted with a diverse set of representative packet flows (including packet content) of both benign and malicious traffic. A natural approach for addressing representativeness in both flows and content is to take empirical traces from real networks for offline analysis. However, this approach is often considered impractical due to standard privacy concerns and the difficulty in accurate labeling of individual packets as benign or malicious. The most notable exceptions are the well known DARPA data sets developed at Lincoln Lab in 1998–1999 for offline NIDS testing [2], [3]. The authors of those studies went to great lengths to create software robots that mimicked user behavior as a means for gathering empirical trace data. While this work has since come under some criticism, it remains the largest publicly available data set for offline NIDS testing, and has been used in many studies.

Another approach to addressing the challenge of robust NIDS testing is to generate traffic streams synthetically. In principle, this process can result in traces for offline tests or in live streams for online tests. While many traffic generators have been developed for specific network systems tests, none of them address the problem of robust NIDS testing in particular. Perhaps most importantly, the synthetic generation of diverse, representative benign traffic (including payload content) has not been well addressed.

Our goal in this work is to create tools and a test methodology for evaluating and tuning the growing number of stateful, protocol-aware intrusion detection systems, with a secondary aim of meeting the test requirements outlined in [1]. These objectives guided our design of a collection of tools, called Trident, which can be used to generate packet traces for traditional offline evaluations, and can also be used in controlled laboratory settings to assess the online performance characteristics of NIDS or other network systems (*e.g.* firewalls). The capabilities of Trident include:

- The ability to generate representative benign traffic streams, including payloads,
- The ability to construct and generate new types of malicious traffic,
- The ability to modulate the mixture of benign and malicious test traffic,
- The ability to modulate the volume of both benign and malicious test traffic,
- The ability to modulate temporal arrival processes of both benign and malicious test traffic.

To the best of our knowledge, no existing toolset provides this combination of capabilities, and we show that they enable a unique and important range of tests for NIDS.

One of the most important features of Trident, and something that distinguishes it from simple malicious traffic generators such as [4], is that it includes representative benign traffic. Trident uses handcrafted automata-based representations of popular network services to generate a wide range of protocol-

compliant packet streams. The packets (headers and payload) within the streams are extracted from traces that have been carefully groomed to remove malicious content. We outline three strategies for trace grooming in § III. A more complete discussion is found in [5].

We demonstrate the capabilities and utility of Trident through a series of tests on live systems in a controlled laboratory environment. We begin by populating Trident with two different benign traffic traces. Next, we create a set of attack modules for malicious traffic commonly seen in today's networks, as explained in § IV. We use the combination of benign traces and attack modules to assess the behavior of two popular NIDS over a range of traffic volume and packet diversity (content and mix). Our experiments are based on a set of test hypotheses and protocols designed for each system type. The results show that Trident easily exposes an important range of behavior in our test systems. In particular, we show how NIDS performance can be sensitive to the mix of benign application payloads. We also show that the relative proportions of malicious flows to all traffic has a very clear impact on NIDS performance and resulting alarm quality. We further show that while traffic volume has a clear effect on NIDS packet loss, its effect on alarm quality is system dependent. The key implication of our results is that Trident is well suited for evaluating NIDS or other network systems that are protocol-aware and stateful (maintain connection state for detecting anomalous or malicious activity spanning multiple packets or connections). Our results also suggest that Trident would be very useful for tuning NIDS rule sets and the host systems on which they run.

II. Related Work

Several tools exist for generating purely malicious traffic, including [4], [6], [7], [8]. Trident is also related to the Metasploit and Exploitation Framework projects that provide libraries of common modern attacks [9], [10]. Efforts toward generation of both benign and malicious traffic streams include [11], [12] and a commercial product from Skaion [13]. Trident differs from these systems in its approach to benign traffic generation and the flexibility that it provides in controlling the volume, mix and content of produced traffic streams.

A study following on McHugh's critique was performed by Mahoney and Chan [14]. The authors conducted an evaluation of *anomaly-based* NIDS with an enhanced version of the DARPA data set created by injecting benign traffic from a single host (their department web server). Our work, while somewhat similar, differs in the following ways: (i) our target systems are much broader than anomaly-based NIDS; (ii) our goal is to provide a flexible and extensible framework for recreating a wide range of attack scenarios by modulating the mix of malicious and benign traffic, control of traffic volumes, and inclusion of representative benign payload contents; (iii) we consider the problem of separating potentially malicious traffic from benign traffic based on protocol knowledge and statistical properties of the traffic instead of relying on a firewall or manual grooming.

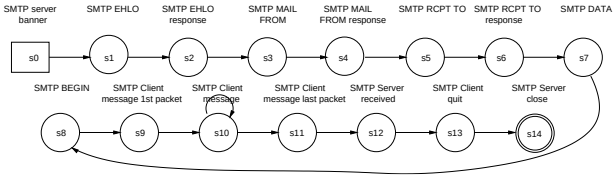
III. Constructing a Benign Trace

One of the most important aspects of NIDS evaluation is a thorough assessment of the system's propensity to generate alarms in the absence of malicious traffic (false positives). The quantity of false positives is intrinsically tied to both the NIDS under test and the nature of benign traffic in the test environment. Therefore, one of the essential aspects of NIDS evaluation for any network is a benign traffic workload that features the spectrum of characteristics that are *typical* or *expected* for that network. While one might be able to readily capture a collection of packet traces from the network over an appropriate period of time, the difficulty arises from the fact that we expect these traces to contain a mixture of both benign and malicious traffic. So, the question becomes *how to identify and isolate the benign traffic*. In many ways, this is precisely the intrusion detection problem.

There are several possibilities for populating Trident with benign traffic payloads. We describe three strategies below. While other techniques may be possible, we believe that these strategies are reasonable, effective and cover a large portion of the design space.

- **NIDS-based Strategies:** The first strategy is to use a standard NIDS such as Snort (that is well known to generate a large number of false alarms with its full rule set, but also detects true attacks accurately) to groom a packet trace taken at a local site. We argue that this approach is problematic since a portion of the connections that will be identified as malicious (and therefore removed) are likely to be those that are of "highest interest" in the sense that they are benign packets that trigger alarms (*i.e.*, false positives).
- ***Synthetic Generation Strategies:*** A second strategy is to use synthetic traffic generated using software robots that emulate user behavior. The idea is to craft the robot to ensure that it only creates connections with known good hosts (either local or remote). This data is then used as a basis for further expansion of the trace through synthetic generation of packets (as in the DARPA data set). While this strategy is clearly limited in terms of representativeness from an application mix and payload perspective, it may be appropriate for certain environments. A further benefit of this method is that since the base trace is generated by robots, it may enable trace data sets to be shared.
- ***Trust-based Strategies:*** The third strategy is to groom a packet trace taken at a local site using connection heuristics (*e.g.*, failure rates or scanning characteristics). This approach exploits the differences in connection characteristics of benign versus malicious sources based on a model of malicious connection behavior. This technique is attractive because it is based on transport level characteristics, does not require knowledge of application semantics, and is not biased by a particular system (NIDS independence). We posit that a trust-based grooming strategy results in a set of packets labeled as benign that have a higher opportunity for uncovering false positive behavior. It is, however, limited in that it might miss targeted attacks by sophisticated adversaries that have connec-

(a) HTTP service automaton.



(b) SMTP service automaton.

Fig. 1. Service automata for two common application protocols.

tion characteristics sufficiently similar to benign users. In [5], we examine the trust-based strategy by defining the specifics of a trust-based grooming methodology and evaluating its performance using a packet trace from our department border router.

IV. The Trident System

In this section we describe how application-specific benign traffic streams and malicious traffic streams are generated within the Trident framework.

A. Benign Traffic Generation

Several studies of IDS performance, including many papers that use the DARPA data set, consider detection characteristics from an offline evaluation (*e.g.,* [15]). As we show in the results of our laboratory experiments, the dynamic characteristics of traffic streams, notably the mix and volume, can have huge impact on NIDS performance and provide key insights for system tuning. Therefore, one of the challenges of benign traffic generation is to dynamically generate diverse traffic streams based on knowledge obtained from a limited set of traces.

In Trident, we adopt protocol-aware emulation of traffic based on payload interleaving. Payload interleaving is our term for dynamic construction of flows through random selection of packets from payload pools corresponding to particular states in a service automaton, as we describe below. This method supports the generation of synthetic traffic streams with realistic application level headers and payloads. We describe the components of our protocol-aware emulation scheme below.

• **Automata Generation.** At the heart of our benign traffic generation system is a collection of automata with states that describe classes of packets observed in a specific service. In our preliminary exploration of the feasibility of this approach, we use basic automata to describe the most popular services seen in our test data sets described in § IV-C. We do not claim completeness of these automata or suggest that they exercise all classes of NIDS. We use them as examples to demonstrate the utility of our methods and to show how they can accommodate flexible recreation of a broad class of protocols.

Our automata describe each service through a three phase abstraction that is typical of most network protocols. The first stage, *prologue*, describes the application-level client server handshake. The next stage is *dialog*, in which client and server exchange data. The final stage is *epilogue*, in which the participants agree to gracefully tear down the connection. Each stage in the conversation could involve several states in the automata and the final stage is optional in some protocols such as HTTP. We created automata that model the packet exchange protocols for HTTP, SMTP, DNS, Telnet, FTP and SSH (*i.e.,* the most popular services from our data sets). Our automata-based abstractions for HTTP and SMTP are shown in Figures 1(a) and 1(b). Pipelined HTTP requests are currently not supported but should be an easy extension.

A weakness of a protocol-aware automata-based system such as ours is that the effectiveness of the evaluation is related to the quality of the automata. It is our hope that a library of automata will be developed by the research community over time.

• **Payload Classification.** The raw traces classified as benign (*i.e.*, trusted or trust neutral) are given as input to the payload classification module which we call `payload-gen`. The purpose of `payload-gen` is to classify packets in the trace into various pools that correspond to particular states of different service automata. In this step, packets generated in the same application state, but from different flows, are aggregated into the same pool. This aggregation does not preserve packet ordering from any individual flow.

• **Payload Sanitization.** Following classification, payloads are discarded or modified to ensure that they do not violate a simple set of requirements. Discard is appropriate if the original payload suffered truncation during packet capture or the payload does not match a valid automata state. Modification is generally done to simplify service automata definition and processing, and to avoid generation of false alarms that would result simply as an effect of interleaving. Our current approach is to be aggressive in these normalization steps. For example, with HTTP, we remove `Connection`, `Content-length`, and `Transfer-encoding` headers from server responses, since a NIDS could conceivably use these items to monitor a connection in progress. Since we wish to arbitrarily use client and server payloads without maintaining elaborate state or dynamically rewriting payloads, we remove the echoed address from the server response since it is not required for correct protocol operation. An effect of our sanitization is that we may underreport the levels of false alarms generated by the NIDS that we evaluate.

• **Content Aware Traffic Generation via Harpoon.** We wrote a new traffic generation plug-in for Harpoon [16] to execute the application state machines and transmit sanitized payloads. Control of state machine processing is done on Harpoon clients. Harpoon servers simply respond to requests to send a certain number of packets from specified application payload pools.

The exchange of application-layer payloads according to a given service automaton is done using standard user-level

sockets, *i.e.*, above the transport layer. Benign traffic streams produced by Trident are therefore targeted at intrusion detection systems focused on layers above the transport layer. Presently, we do not explicitly account for non-malicious transport-layer anomalies that may have been present in a trace captured in a live network, such as misconfigurations or implementation bugs. Our malicious traffic generators, described next, include exploits that target both network and transport layers, as well as higher layers.

B. Attack Traffic Generation

• **General attack traffic creation.** MACE is a modular attack composition framework that consists of three primary components: (*i*) exploit, (*ii*) obfuscation, and (*iii*) propagation, as well as a number of functions to support interpretation, execution, and exception handling of attack profiles. For this work, we extended the existing set of exploits in MACE from 5 to 21 attacks [4] and enhanced its ability to modulate attack volumes. A taxonomy of available MACE exploits is found in [5]. Our objective is not to provide a complete attack database for intrusion testing, but to provide a spectrum of attacks that exercise NIDS in sufficiently diverse ways and to support a set of basic building blocks that can be used to create additional (and perhaps as yet unseen) attack vectors.

• **DARPA attack recreation.** The DARPA data set provides a collection of 58 different attack instances. A catalog of these attacks is found in [5]. To extend the utility of Trident, we added the capability to dynamically replay these attacks. We began by developing a tool called `split-darpa` to distill labeled attacks from the mixed traces. Due to possible inaccuracies in the labeling, `split-darpa` was able to automatically isolate 56/58 attacks in the data set. Next, we developed the ability to perform dynamic replay of attack traces with the tool `attack-replay`. One of the key aspects of this effort is that for TCP attacks, reassembly of payloads is done before sending the packets through TCP sockets. All state is maintained at the client and appropriate server responses are fed to the server through an out-of-band control channel in a timely manner. For UDP and ICMP packets, the traffic is transmitted through raw sockets.

C. Test Methodology

The objective of laboratory-based experiments reported in § V is to demonstrate the utility of Trident by evaluating the effectiveness of specific NIDS configurations along dimensions of packet diversity (content and mix) and traffic volume.

Test Setup. The NIDS we evaluated in our experiments were Bro (version 0.9a8) and Snort (version 2.3.0). For Bro, we used the default `brolite.bro` policy, and for Snort, we used the default `snort.conf`. We included a third NIDS configuration consisting of Snort (version 2.3.0) with a recent snapshot of signatures from Bleeding Snort [17]. Each NIDS ran on a separate workstation with a 2 GHz Intel Pentium 4 processor, 1 GB of RAM, and Intel/PRO 1000 network cards. FreeBSD 5.1 was installed on each machine[1].

The three NIDS hosts were connected to a Cisco 6500 enterprise switch/router. Harpoon, MACE, and attack-replay traffic generators were also connected to this switch, which was configured in such a way that the three NIDS received all traffic sent between the traffic generation hosts. We used two large (2^{16}) address spaces as "internal" and "external" networks, configuring interface aliases on each traffic generation host.

We ran three sets of experiments. The first set of experiments was designed to establish a baseline of alarm behavior for each NIDS. We first generated low (5 Mb/s) rates of benign traffic using the DARPA data set and a sanitized packet trace from our department border router (CSL trace). We traced all traffic during these experiments and used the traces in an offline manner to produce a baseline set of alarms generated by the three NIDS configurations. Similarly, for each exploit produced by MACE and each exploit from the DARPA data set, we generated a baseline set of alarms for the three NIDS configurations.

In the second set of experiments, we altered the mix of flow volumes between benign traffic generated by Harpoon, and malicious traffic generated by MACE (CSL) or attack-replay (DARPA). To effect different mixes, we kept the benign traffic level constant at about 20 Mb/s, while introducing different levels of malicious flows. The specific mixes we used were 100% benign flows, 90% benign flows, 50% benign flows, and 10% benign flows. Below, we refer to these test setups as mix100, mix90/10, mix50/50, and mix10/90, respectively.

In the third set of experiments, we used three different levels of traffic volumes. We tuned Harpoon to generate roughly 20 Mb/s, 40 Mb/s, or 60 Mb/s for each of the CSL and DARPA data sets. For each traffic volume level, we tuned MACE or attack-replay to produce approximately 2 Mb/s, 4 Mb/s, and 6 Mb/s of aggregate attack traffic, respectively. Below, we refer to these tests exploring the effect of traffic volumes as vol20, vol40, and vol60.

We ran each experiment for 15 minutes. On each NIDS host, we measured CPU and memory usage every 5 seconds using `vmstat`. We also took note of packet drops reported by each NIDS upon shutdown.

Evaluating Results. Using the results of running the malicious traffic in the baseline experiments, we constructed representations of the types and number of alarms at each NIDS we expected to observe for each exploit. For the second and third sets of experiments, we recorded the number of times an individual exploit was executed by MACE or attack-replay. We then were able to compare the alarms produced by each NIDS with what we would expect, given the specific exploits launched over the duration of the test. We wrote a script to automatically process this representation of expected alarms along with the actual log files produced by a NIDS during

[1]On each host we modified the kernel parameters `debug.bpf_bufsiz` to 4194304 and `debug.bpf_maxbufsize` to 8388608 as suggested in the Bro documentation. Snort presumably can benefit from this change as well so we applied the change to each NIDS host.

a given test. The script reported the set of alarms produced by the NIDS, along with a frequency of occurrence, and the expected number of occurrences. We used these counts to generate relative alarm efficiency and effectiveness values [18]. Efficiency is defined as $Efficiency = \frac{TruePositives}{AllAlarms}$, and is a measure of false positive occurrence, where a value of 1 means that there are no false positives and a value of 0 means that all alarms are false positives. Effectiveness is defined as $Effectiveness = \frac{TruePositives}{AllPositives}$ is a measure of false negatives, where a value of 1 means there are no false negatives (*i.e.*, all alarms that should have occurred did occur).

V. Laboratory-based IDS Evalution

We illustrate the utility of our framework through results of an experimental evaluation of Bro, Snort and Bleeding Snort performance under varying traffic content, mix and volume. While these results substantiate the effectiveness of the measurement tools and the potential of the methodology, they should be interpreted with the following two caveats:

1) The results are limited by the representativeness and diversity of our protocol automata.
2) Our goal is to conduct black-box evaluation of NIDS performance. So we do not perform in-depth analysis of behavioral causalities.

As a result, *these results are not intended to be used as a head-to-head comparison of the systems or their rule sets*. However they are valuable in that they demonstrate effects of varying benign and malicious traffic content, mix and volume, and establish the feasibility of our approach.

Baselining Benign Traffic / Evaluating effect of interleaving. An important question is how payload interleaving, the process of random selection of packets from individual payload pools based on the states in each service automaton, affects alarm characteristics. In particular, it is important to demonstrate that no legitimate alarms are introduced due to data consistency issues.

We counted the number of *unique* alarms produced by the three NIDS setups in both offline and online configurations using the CSL and DARPA data sets. Results are found in [5]. For the offline setup, we ran each NIDS configuration using each trace after grooming, but prior to payload classification and sanitization so that the original flows (including IP and TCP headers) were left intact. In the online setup, we used Trident to generate flows using the groomed, classified, and sanitized traces. We saw that the number of unique alarms is consistently less for the online test. This effect is caused by the conservative nature of our sanitization process and by the fact that our laboratory tests are run in a relatively simple environment. Furthermore, the set of alarm types generated in the online tests is a subset of the alarm types produced in the offline setup. Alarms unique to the offline tests are most often related to transport, addressing, and routing (*i.e.*, layers 3 and 4) and the common alarms are application-related.

Baselining Malicious Traffic. We compiled summaries of the alerts generated by Bro, Snort, and Bleeding Snort for each instance of the 21 MACE and 52 DARPA attacks, details are found in [5]. We used this data to derive expected alarm types and frequencies for categorizing alarms, *e.g.*, as false positives. Certain exploits in these tables highlight some of the main differences between Bro and Snort, *i.e.*, Bro is generally concerned with stateful monitoring of connections and applications, while Snort is oriented toward detecting specific conditions in individual packets (such as the presence of a particular string). For example, a specific string in the **winnuke** exploit in generates 10 alarms in Snort and Bleeding Snort but does not trigger any Bro alarms. Similarly, unexpected SMTP state produced during the **mailbomb** exploit triggers 450 alarms in Bro, but 0 in Snort and 60 in Bleeding Snort.

Evaluating effect of Traffic Mix. Tests using a range of benign and malicious traffic mixes demonstrate a remarkable diversity in the behavior of Bro, Snort, and Bleeding Snort. Figure 2 shows CPU utilization and packet loss results for the CSL/MACE data set. Results for the DARPA/attack-replay data set are found in [5]. We see that as the overall flow composition becomes dominated by malicious flows, CPU utilization shows a clear increasing trend. We also see a difference caused by the application mix of benign traffic flows. For the CSL data set with no malicious flows (mix100), we see that Bleeding Snort consumes more CPU time than Bro, but that the opposite occurs for the DARPA data set with no malicious flows.

Packet loss behavior between the CSL and DARPA traffic is quite different for both Bro and Snort/Bleeding Snort. For both Snort variants, there is always some occurrence of packet loss. (We are currently investigating the cause of this high level of packet loss even with relatively low CPU utilization, which is consistent with previously reported experiments [4].) On the other hand, Bro appears quite resilient to dropping packets until it consumes all CPU resources, at which point loss becomes endemic.

Figure 3 shows alarm effectiveness and efficiency over the range of benign and malicious flow mixes for the CSL/MACE data set. Results for the DARPA/attack-replay data set are found in [5]. We see that for the CSL data set alarm effectiveness drops as the traffic mix becomes dominated by malicious flows (false negatives increase). This effect is mainly caused by packet drops experienced by Bro and both Snort variants. False positives are relatively low across all mixes for the CSL data set, with the fewest false positives coming when most of the traffic is malicious (mix10/90).

Evaluating Effect of Volume. To understand the effect of volume in NIDS performance, we conducted experiments with traffic rates of 20, 40 and 60 Mbps. In these experiments, the mix of malicious to benign traffic was fixed at 10/90. Figure 2 shows the packet drops and CPU utilization for the three systems under different volumes for the CSL/MACE data set. On all systems, an increase in traffic volume directly affects CPU utilization. Interestingly, while Bro seems quite resilient to packet drops with the CSL data set, the DARPA data set, which is dominated by HTTP traffic, has a substantial impact on Bro performance (see [5]). Snort's drop rates seem to degrade less intensely with volume for this data set.

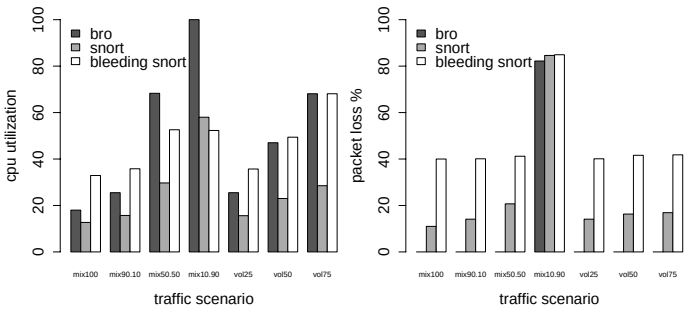


Fig. 2. CPU utilization (left) and packet loss (right) measurements for Bro, Snort, and Bleeding Snort on CSL traffic setup.

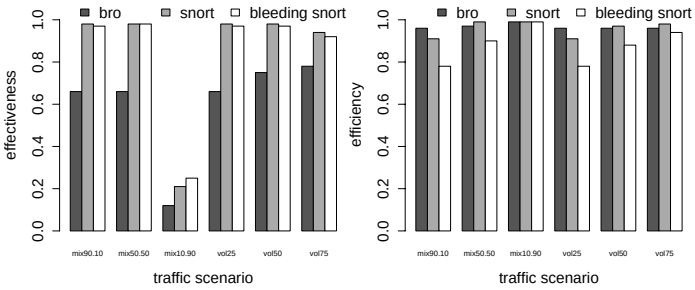


Fig. 3. Alarm effectiveness (left) and efficiency (right) for Bro, Snort, and Bleeding Snort on CSL traffic setup.

In Figure 3 we show the effectiveness and efficiency of the systems in the volume experiments for the CSL/MACE data set. Efficiency and effectiveness do not seem to be highly correlated with volume of traffic although counts of alarms do increase with volume. Second, while we do not show the results here, it seems that Snort's signature set has been to tuned to detect DARPA attacks much better than Bro. Bro generates few alarms on the DARPA data, and as a result even though Bro produces few false alarms its efficiency and effectiveness are poor. For the CSL traffic, Bro's efficiency and effectiveness are significantly better. Another observation is that the Bleeding Snort rule set, which is a superset of the Snort rule set, seems to have lower efficiency and effectiveness. As we would expect, the volume of baseline alerts in Bleeding Snort is higher than for Snort. As a result, Bleeding Snort has a greater chance of missing true alarms during packet drops but also a larger chance for false positives, which decreases effectiveness.

VI. Conclusion

In this paper, we describe an online traffic generation framework for robust evaluation and tuning of network intrusion detection systems. The objective of our work is to create a system capable of generating realistic, diverse streams of both benign and malicious traffic. We describe a methodology for creating benign traffic that is based on protocol specific automata and includes the use of packet payloads culled from live traces. We argue that this approach enables highly representative testing and will be quite useful for security administrators seeking to tune systems for their own environments. We realize our traffic generation framework in a tool set we call Trident. We demonstrate the utility of Trident through a set of experiments on open-source NIDS conducted in a controlled laboratory setting. The results of these experiments indicate that content, mix and volume have tremendous effect on NIDS performance and demonstrate Trident's capability to expose a range of diverse behavior on modern NIDS.

References

[1] J. McHugh, "Testing Intrusion Detection Systems: A critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, November 2000.

[2] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating Intrusion Detection systems: 1998 DARPA Off-line Intrusion Detection Evaluation," in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 1998.

[3] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)*, Toulouse, France, October 2000.

[4] J. Sommers, V. Yegneswaran, and P. Barford, "A Framework for Malicious Workload Generation," in *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference*, Taormina, Italy, October 2004.

[5] ——, "Toward Comprehensive Traffic Generation for Online IDS Evaluation," University of Wisconsin-Madison, Tech. Rep. 1525, October 2005.

[6] "THOR: A Tool to Test Intrusion Detection Systems by Variations of Attacks," http://thor.cryptojail.net/, 2005.

[7] D.Mutz, G. Vigna, and R. Kemmerer, "An Experience Developing and IDS Simulator for Black-box Testing of Network Intrusion Detection Systems," in *In Proceedings of ACSAC*, Las Vegas, NV, December 2003.

[8] Nessus, http://www.nessus.org, 2005.

[9] "Metasploit project," http://www.metasploit.com, 2005.

[10] Exploitation Framework, http://www.securityforest.com/wiki/-index.php/Exploitation_Framework, 2005.

[11] S.-S. Hong and S. F. Wu, "On Interactive Traffic Replay," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)*, Seattle, WA, September 2005.

[12] L. Rossey, R. Cunningham, D. Fried, J. Rabek, R. Lippman, J. Haines, and M. Zissman, "LARIAT: Lincoln Adaptable Real-Time Information Assurance Testbed," in *Proceedings of IEEE Aerospace Conference*, Big Sky, Montana, March 2002.

[13] "Skaion's Traffic Generation System (TGS)," http://www.skaion.com/products/index.html, 2005.

[14] M. Mahoney and P. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Intrusion Detection," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)*, Pittsburgh, PA, September 2003.

[15] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)*, Toulouse, France, October 2000.

[16] J. Sommers and P. Barford, "Self-Configuring Network Traffic Generation," in *Proceedings of ACM SIGCOMM Internet Measurement Conference*, Taormina, Italy, October 2004.

[17] "BleedingSnort," http://www.bleedingsnort.com, 2005.

[18] S. Staniford, J. Hoagland, and J. McAlerney, "Practical Automated Detection of Stealthy Portscans," *Journal of Computer Security*, vol. 10, no. 1-2, 2002.