The Challenges in Traffic and Application Modeling for Intrusion Detection System Benchmarking [1]

Technical Report CSTR 030600

Hilmi Gunes Kayacik

Abstract: *The lack of test data in intrusion detection system (IDS) research is a known problem. An ideal IDS Benchmark should provide a wide variety of data and extensive documentation for better analysis. This technical report summarizes the efforts relevant to IDS benchmarking and aims to determine a direction to our benchmarking efforts.*

1. Introduction

Like all crucial systems, intrusion detection systems should be tested thoroughly to determine and eliminate its weaknesses. In case of data driven systems such as machine learning and anomaly based IDSs, "training" data, which characterizes the normal use of the environment is employed to adapt the system to a particular environment. Finding data to test and train intrusion detection systems has many challenges. An IDS is (or should be) tested on a mixture of normal usage and attacks to determine its detection capabilities as well as its false alarms on normal behavior. Generating attacks and testing detection rate is fairly easy compared with the normal usage modeling. Attack traffic is sufficient to test signature based IDSs because IDS simply matches attack signatures with the input data. Therefore normal usage can be simplified as the background noise from which IDS detects the signals (attacks). On the other hand, modeling normal usage gains importance when the IDS learns the normal usage from the input data. In that case, IDS tries to discriminate attacks from the normal behavior. In data driven systems, learnt model is as good as the input provided. Therefore, normal usage can't be simplified as background noise. Although application usage or traffic modeling is not a new research area, previous efforts (that are unrelated with IDSs) aim to generate application traffic that will be similar to real traffic from network management perspective. Therefore these efforts focus on network layer aspects of the simulation such as the simulating congestions, modeling utilization, burstiness and so on. This technical report focuses on traffic modeling and significant IDS benchmarking efforts. For more information regarding the details, readers are encouraged to refer to the references.

User behaviour can be simulated in two ways: (1) Record & replay and (2) Analyze & modelling. In record & replay approach, the sessions of a real user is recorded and later replayed on the test bed. In addition to employing *tcpdump* to record sessions on the network, *Expect* [3] can be employed (and extended) to record the sessions on the host.

[1] In this technical report, the term "benchmark" and "benchmarking" implies the collected data and the methodology employed during the process.

This solution is easy but does not provide much insight about the users or the network. Additionally, in real world, elastic protocols such as TCP behaves differently on congested networks. Therefore the replay may not be representative [5]. Second approach is to analyze the user behaviour on live data and try to model his/her behaviour rather than capturing the his/her live sessions. Section 2 of this report summarizes the efforts on modeling traffic and user behaviour. Moreover, IDS benchmark efforts explained in Sections 5, 6 and 7 tackle the problem in their own way.

Simulating attacks are easier than user behaviour, because resources such as exploit databases and security mailing lists provide detailed information about the discovered attacks. Usually simulating an attack can be as easy as reading the attack documentation, writing (or finding) its code and deploying the attack to the appropriate target. Appropriate target means, if the attack targets a specific version of the FTP server, that particular FTP server should be installed on the target machine. IDS benchmarks explained in Sections 5, 6 and 7 more or less use this approach to generate attacks. One alternative solution might be deploying a honeypot to attract attackers to a target machine and record all the session and traffic data on the target machine (of which the majority is attack). Honeypot approach will be discussed in Section 3.

2. Traffic Modeling (Analytic Approach)

In traffic modeling approaches, main goal is to generate a workload similar to real world traffic. Example uses of the simulated traffic are mostly related with network management and engineering tasks such as testing QoS of a network protocol or reliability of a router. As far as the content is concerned, only source and destination make use of the packet content (i.e. 7th layer of OSI: Application layer). For a router working at the network (3rd OSI) layer, packet content is irrelevant. Therefore many traffic generation approaches treat the content as random bytes. On the other hand, in order to have a useful traffic model for IDS testing, both application and traffic modeling should not be overlooked. Modeling traffic (or application) involves analysis of the live data.

Traffic analysis starts by finding the invariant characteristics. Invariants are defined as "some facet of Internet behaviour shown to hold over a wide variety of environments and to be observable by any observer."[9] Invariants are also characterized by the probability distribution they fit[2]. Invariant characteristics can be selected from any layer[3]. For example on the network layer, interarrival time of the packets; on the transport layer, TCP connection time; and on the application layer, sizes of the requested web pages could be derived. Further information about the invariants on different layers and their probability distributions can be found in [9,10].

[2] For instance, file sizes follow Pareto distribution. Employing statistical methods to determine the probability distributions is discussed in [13]

[3] In this report, the term "layer" refers to the OSI layers.

The mathematical analysis, which is based on several invariants from different layers, will produce a more accurate model of the real network traffic. On the other hand it also creates the challenge of combining all the invariants to generate the final output. A solution to this problem is covered in [11]. Fortunately, traffic generators such as *GenSyn* [5] and *Surge* [11] employ user behaviour models to generate synthetic but realistic traffic. *GenSyn* employs the user modeling approach and use the workstation's built-in network protocol stack to generate the traffic. This is important because different vendors can implement the same protocol with differences. GenSyn models the user behaviour in a semi-Markov State model, while the communication systems are not modelled. The equipment constraints and the protocol behaviour are automatically included through the linking of the user modelling to the built in protocol stack [5]. User states are divided into two categories: stochastic state space for user behaviour and communication state space for users that are waiting for a response. Further information on state spaces can be found at [5]. *Surge* [11] is another traffic generator that aims to test the resources of the servers such as buffers and file systems by creating HTTP traffic. User equivalent concept in *Surge* simulates the user behaviour including thinking time or waiting for the page to appear on the screen (called OFF time). This is important because, ignoring OFF times of a user can destroy the self-similarity[4] of the network traffic [11]. Workload is characterized by invariants such as file sizes, request sizes, popularity of a page, embedded references in the page and temporal locality (caching). These invariants are then combined to produce HTTP traffic. This is not a trivial task and solutions adopted in Surge are explained in [11] with details.

Although these traffic generators provide a way to generate fairly representative traffic, they are limited to the protocols such as HTTP or FTP where file (or generally data) transfer is essential. In case of interactive applications such as telnet, user modeling becomes more important. There are numerous ways to model user behaviour:

> Markov models: Finite State Automata where user behaviour or the actions than can be taken are expressed in states. The next action (state) is determined by the current state and a fixed probability. [16]
> Petri Nets [8]: Unlike Markov models, this is a memoryful approach where users next action is based on the satisfied conditions. Discouragement, satisfaction, exhaustion can be modelled.
> Selecting behaviours according to the user selection [12].
> Psychological and sociological user profiles [12].

3. Honeypots

The honeypot approach tries to attract hackers to a controlled but vulnerable system to gather information about their methods. Honeypots should allow hackers to continue their activities on the system as long as possible without detecting the honeypot or harming other networks. One of the honeypot organizations called the HoneyNet project "is a

[4] One of the important characteristics of the network traffic is self-similarity. It implies that the object will look like itself regardless of the magnification [10]. It has a heavy mathematical treatment therefore left out of this report. Detailed explanation can be found in [10, 13].

security-research organization dedicated to learning the black hat community's tools, tactics and motives and then share the lessons learned." [14] Some key advantages of honeypots are [14], collecting small amount of high value data and capturing hacker activities without his/her detection. Disadvantages of honeypots are limited view of the network and security risk, in case the hackers compromise the honeypot machine. There is no pre-packaged software for honeypots, therefore honeypot should be considered as an architecture design. [4] details the design issues of honeypots such as:

> Setting data control: Hackers should be able to break into the honeypots, but should not be able to use them to attack other systems. To control the in/outbound traffic, a layer 2 bridge or an IDS (e.g. Snort) can be employed.
> Data Capture: Collected data can not be kept on the honeypot because hackers can cover their tracks after the break in. Honeynet Project [14] provides various ways to manage the captured data.

Honeypot approach in [14] is proven to capture attacks and provide data for further analysis.

4. A Practical Benchmarking Approach

The simplest way (for the evaluator) to evaluate an intrusion detection system is to deploy the system on a live network without making any benchmarking plans or developing any test cases. Detection rate is determined on the attacks that are injected to the network and false alarm rate is observed on the live network data. Along with its obvious advantage (easy to perform) it has many disadvantages. Due to disk space limitation and privacy issues, live data, which is employed in the IDS benchmark, is not stored for further use. Therefore, IDS performance analysis suffer from "on the fly" nature of the benchmark. Moreover, the environment on which the IDS works is not sufficiently characterised. That is, models of the network traffic and the applications are either ill defined or neglected. Lack of plans and methodology also affects the performance analysis where false alarm and detection rates are expressed with sketchy metrics and without any details of the data employed. Additionally, due to the difficulty of generating many complicated attacks, benchmarking focuses on only a few attacks. Results from the IDS benchmark always remain questionable.

5. A Software Platform for Testing IDS [1]

One of the initial benchmarking efforts that aimed towards controlled evaluation involves developing test cases for normal and attack sessions where each test case consists of a sequence of commands. *Expect* [3] is the most commonly used software to create automated text based command sessions. In this methodology, emphasis was on generation and synchronization of attack scripts. Synchronization is necessary when the attack might involve multiple steps, each depending on the previous step. As an additional reason for synchronization, deterministic execution is claimed to be required for repeatable benchmarking. Normal behaviour was considered as the background noise; hence no parameters were defined except the level of the background traffic (workload). As indicated in their paper, this leads to a limitation in case of benchmarking anomaly

systems where definition of "normal" plays an important role. One of the strengths of [1] is its comprehensive evaluation methodology. Many criteria are employed in the evaluation such as:

> Intrusion identification: Detection rate on attacks
> Background noise: False alarms on normal traffic
> Resource Usage: Disk usage data collected
> Smokescreen noise: Mixing attack commands with legitimate ones (e.g. `attack_command1; ls; attack_command2; pwd;` so on)
> High volume data: Extensive amount of data
> Intense data: Fast data
> Load: IDS shares CPU with other programs

Their results indicate that increasing stress conditions such as introducing noise or increasing data rate affects the detection of the evaluated IDS. According to their paper [1], authors can be contacted to request the scripts that they employed.

6. IBM IDS Workbench Research [2]

One of the design principles of this benchmark is to generate wide variety of normal usage patterns (which was not emphasized in [1]). The focus of this benchmark is limited to FTP service. Three ways to generate normal usage is discussed. These are:

> Using finite state automata to model all possible actions that a user can take. (Good, if number of commands is limited.)
> Using test cases developed by OS designers. It is a good approach since a test case would contain all kinds of usage patterns that the designers expect from the users. However test cases may not represent the normal usage.
> Recording and later replaying the live data. This approach is employed in the benchmark.

Exploit database maintained in IBM is employed to generate attacks. Both normal usage and attacks are generated by *expect* scripts. All the IDS evaluated are host-based systems. Both client machines and servers (running IDSs) are controlled by a workbench controller, which provides a user interface to control the benchmark, sets up the IDSs, tunes and triggers the normal behaviour. The controller seems to interact with the IDSs where this might be a shortcoming if workbench controller needs to be extended or re-configured for each new IDS that are added to the test bed.

7. MIT Lincoln Labs. DARPA 99 Data [4]

The main design goal of DARPA IDS datasets is to provide a comprehensive IDS benchmark[5] that can be shared by researcher to test and train IDSs. Unlike other approaches, which focus on specific aspects of benchmarking, this evaluation adopts a wider approach and simulates the daily use of a military network for 5 weeks. Recorded

[5] In [4], prior approaches are criticized to use small number of attacks and little background traffic.

traffic in *tcpdump* format and audit logs were made public for off-line IDS testing and training. One of the significance of DARPA benchmark is, it employed a modified Linux kernel to make the traffic look like it is originating from thousands of hosts. Unfortunately, this is achieved at the network layer, all the traffic originating from the modified Linux machine have the same MAC address, therefore except the IP addresses, the traffic does not really look like it is originating from thousands of hosts. This proves to be a problem during replay, because all IDSs except the ones working on promiscuous mode will neglect the traffic because the traffic is destined to another machine (according to the MAC address on the packet).

Expect scripts for attacks and normal usage is prepared prior to benchmarking. At the beginning of each day, attack scripts are placed on the attacker machines and scheduled to run appropriate times during the simulation. Thousands of normal usage sessions in *expect* scripts were administered in the same fashion. The victims (i.e. attack targets) are the real machines running different operating systems. Simulated machines are not attacked except the probes (such as ipsweep). John McHugh's critique paper [6] addresses the shortcomings of the DARPA bench marking. In summary, shortcomings are as follows:

> Simulated normal usage is said to be similar to the normal usage of a real network. However no further data (statistics that are used to describe normal usage and measures) are provided to prove this claim except a few things such as average number of words in a mail[6]. This also raises many questions about how the normal usage is modeled.
> Data rate in the recorded traffic is suspiciously low. It takes only hours to replay one the day traffic, which is supposed to contain the traffic of thousands of hosts.
> Internet traffic is not well behaved; On the other hand, DARPA traffic is found to be too clean and to have idiosyncrasies that can lead to detection. Further investigation on simulation artefacts and ways to solve the problem is explained in [7]. A simple analysis we performed on KDD 99 dataset also confirms that real network data is more diverse than the simulated traffic in DARPA 98 dataset.
> Benchmark is criticized to have insufficient validation. Experimenters should frequently check whether the normal usage scripts really generate modeled normal usage or not. Success of the attacks should also be checked. Lack of usage models and statistics seem to be the main reason for the lack of validation.
> Unlike most of the real world networks, simulation network has a flat architecture, where all hosts on the simulated network are connected to the same hub.
> Attack taxonomy is from attacker's view. A better taxonomy suggested is based on the layer(s) that the attack manifests itself. For example probe attacks can be detected by inspecting the packets whereas a buffer overflow attack is likely to be detected by maintaining session state.
> Evaluation methodology that is employed on DARPA benchmark has problems. For example ROC (receiver operating curve) is not the best way to evaluate IDS. ROC curves are useful if the IDS can produce different false alarm and detection

[6] Details are said to be classified.

rates on different configurations. In some IDSs (like ours) ROC will only contain one point.

In [15], Air Force Research Laboratory (AFRL) employed the normal traffic from DARPA 98 benchmark. AFRL found DARPA 98 dataset inadequate for the job mainly because it was impossible to repeat the DARPA 98 benchmark even with the expect scripts that contains normal usage sessions. Therefore AFRL decided to develop a distributed traffic generation technique, which automatically generates sessions. In this approach, user behaviour is kept in a database. A master controller manages slave controllers to start sessions with particular properties (how many sessions to start). Slave controller generates application sessions, which satisfy the conditions passed to the slave controller by the master. Each application has an automata (like DARPA 98 & 99 datasets), which initiates and conduct various services. This mechanism allows benchmark to run unattended. Further details can be found in [15].

8. Conclusions

In this technical report generating data to evaluate IDSs is discussed with relevant literature survey and emphasis on normal usage modeling. Insufficient analysis and modelling are the common shortcomings of all IDS benchmarks. This also leads to insufficient documentation, which makes it very difficult to analyse IDS test results. Suggested goals for our IDS benchmark project are:

Whenever possible, synthetic data should be characterized by models and parameters.

Progress should be documented frequently. An effort as large as DARPA 99 benchmark would involve extensive analysis for modeling. Additionally, details on how the test bed is configured should be documented with details.

Benchmark should support both real time (where evaluator sets up the test bed) and off line (evaluator uses published data) evaluation. Some IDSs interact with their environments a lot (e.g. A distributed IDS which monitors various host resources as well as the network). Therefore, benchmark should be repeatable with the published documentation.

Although the coverage of the benchmark can be limited to one protocol and a few attacks at the initial stages, final benchmark should contain various protocols and various attacks.

At all stages, benchmark and the generated data should be validated. [6] suggests deploying commercial IDS on the test bed for validation. In our case, Snort can be deployed.

Evaluation process is another important issue. If possible, along with the benchmark data, one of our IDSs (and maybe a commercial IDS) should be evaluated on the benchmark data to establish a baseline for analysis of the test results.

9. Summary of the IDS Benchmarks

A tabular comparison of the three benchmarks is given on the following Table.

	UC Davis	**IBM Test Bed**	**DARPA**
Objective:	Develop a methodology for IDS testing.	Develop a workbench with heterogeneous environment and Wide variety of normal usage.	Construct a publicly available test data, extend the attack and normal use coverage.
Emphasis on:	1. Test case selection and testing methodologies. 2. Concurrent intrusions.	1. Ways of generating normal use. 2. Workbench Controller	1. Virtual Hosts 2. Finite state automata employed for normal usage.
Benchmark Type:	Real Time	Real Time	Off Line
Software Platform:	Expect	Expect	Expect
Test Bed consists of:	One machine, IDS	Client, Server, Controller, IDS	Servers and Virtual Hosts
Administration:	Manual. Everything is performed on one host.	Workbench Controller for UI, configuration, analysis	Manual. 1 hour of every day is spent on maintenance
Evaluated IDS:	NSM	Three host based IDS (names are not given)	Various GOTS, Research, Commercial IDS
Normal Usage Derived By:	Record & Replay	Record & Replay	Modelling & Analysis (Details are classified)
Normal Usage Parameters:	Level of background noise	Server Load	Traffic per service Traffic Level We sites visited Telnet content Hourly Traffic variation Email frequency and sizes
Execution:	Synchronized	Workbench controlled	Flexible
Service:	Telnet, FTP	FTP	HTTP, Mail, FTP, IRC, Telnet, X, SQL, DNS, Finger, SNMP, time
# Attacks	7	9	80
Data Provided:	None	None	Tcpdump files Audit files File system dumps
Pros:	1. Good testing methodology	1. Good design principles	1. The *only* available dataset with fairly good attack documentation.
Cons:	1. Limited coverage 2. Lack of normal usage definition	1. Limited coverage 2. Controller – IDS interaction?	1. Outdated. 2. Not documented. 3. Not repeatable. 4. Other flaws discussed in Section 6.

References

[1] PUKETZA, N. J., ZHANG, K., CHUNG, M., MUKHERJEE, B., AND OLSSON, R. A. 1996. A methodology for testing intrusion detection systems. *IEEE Trans. Softw. Eng. 22*, 10, 719–729. http://seclab.cs.ucdavis.edu/papers.html

[2] DEBAR, H., DACIER, M., WESPI, A., AND LAMPART, S. 1998. An experimentation workbench for intrusion detection systems. Res. Rep. RZ 2998 (#93044) (Sept.). Research Division, IBM, New York, NY.

[3] Expect Web Site , http://expect.nist.gov/

[4] Darpa 99 Report, http://www.ll.mit.edu/IST/ideval/pubs/2001/TR-1062.pdf

[5] GenSyn – A Generator of Synthetic Internet Traffic used in QoS Experiments, Poul E. Heegaard.

[6] Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory, John McHugh, ACM Transactions on Information and System Security, Vol. 3 No.4 November 2000.

[7] An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection, Mathew V. Mahoney, Philip K. Chan. TR CS-2003-02

[8] Models of the behavior of People Searching the Internet: A Petri Net Approach, Paul B. Kantor, Ragnar Nordlie

[9] A Short Tutorial on Fractals and Internet Traffic, Thomas B. Fowler

[10] Fractals, Heavy-Tails and the Internet, Martin J. Fischer, Thomas B. Fowler.

[11] Generating Representative Web Workloads for Network and Server Performance Evaluation, Paul Barford and Mark Crovella.

[12] Characterization of Internet Traffic and User Classification: Foundations for the Next Generation of Network Emulation Rigoberto Chinchilla, John Hoag, David Koonce, Hans Kruse, Shawn Osterman, Yufei Wang

[13] Characterization of Internet Traffic, S-38.149 Postgraduate Course in Teletraffic Theory, Esa Hyytiä

[14] The Honeynet Project: Trapping the Hackers, Lance Spitzner, IEEE Security & Privacy.

[15] Air Force Intrusion Detection System Evaluation Environment, Terrence G. Champion, Robert S. Durst

[16] Traffic Source Modeling, Helmut Hlavacs, Gabriele Kotsis, Christine Steinkellner Technical Report No. TR-99101