Towards a framework for the generation of enhanced attack and background network traffic for evaluation of network-based intrusion detection systems

Mr O C W Lo, Dr J R Graves, Prof W J Buchanan
owenlo@gmail.com
j.graves@napier.ac.uk
b.buchanan@napier.ac.uk

Centre for Mobile Computing and Security, Edinburgh Napier University, Edinburgh, UK

Abstract: There are a multitude of threats faced in computer networks such as viruses, worms, trojans, attempted user privilege gain, data theft and denial of service attacks. To combat such threats, multiple lines of defence are applied to a network including firewalls, malicious software scanners and intrusion detection systems (IDS). IDSs are generally considered a last line of defence for the detection of attacks; therefore, it is vital for users to assess how well an IDS will perform through means of testing. Although various methodologies have been proposed for the evaluation of IDSs in the past there is still no widely agreed upon standard.

A framework which is capable of carrying out an evaluation of network-based intrusion detection systems (NIDS) is presented in this paper. The paper shows that such a framework requires the need for both realistic real-time network traffic and meaningful metrics when carrying out an evaluation of IDSs. Automation of the testing process is also emphasised - which provides for ease-of-use and simplicity in repetition when carrying out an evaluation.

The framework is evaluated against the NIDS Snort in order to show its capabilities. Through the use of pre-existing programs and utilities, the aim of generating real-time attack traffic is achieved whilst benign background traffic is generated using static data sets. The metrics of efficiency, effectiveness, packet loss, CPU utilisation and memory usage are derived and, finally, the goal of automation is achieved by implementing the framework as a singular application. The results of the evaluation show that, whilst Snort is highly effective in the detection of attacks (true-positives), its main weakness is the dropping of network packets at higher CPU utilisations due to high traffic volume.

Finally, the conclusion to this paper illustrates that the main weakness with current IDS evaluation methodologies is in the approaches used in the generation of benign background traffic. Whilst using static data sets is viable, the main argument against such an approach is that an IDS under evaluation will not react to the traffic in a real-time manner. Furthermore, the use of synthetic traffic generators also has limitations due to the fact that such traffic may not accurately reflect traffic seen on a live network. This paper proposes that further research and development must be applied in the area of benign traffic generation in order to achieve the aim of providing real-time generation of background traffic which realistically mirrors real-life networks when carrying out an evaluation of IDSs.

Keywords: Intrusion Detection, Evaluation Framework, Attack Traffic, Background Traffic, Evaluation Metrics

1. Introduction & Background

Devices such as burglar alarms, smoke alarms, fire alarms and closed circuit television all fall under the category of real life, physical intrusion detection systems (Del Carlo *et al.*, 2003). The purpose of these devices is to monitor specific threats, and, if the threat occurs, then either produce some form of alert (such as emitting a high pitched noise in the case of fire and burglar alarms) or log the activity (in the case of closed circuit television cameras). Examples of threats include fires and trespassing of property. Consequences of these threats are high, therefore, it can be clear that such monitoring devices are highly important in functioning correctly, and perform the task they are intended to with absolute efficiency. In other words, such devices must perform to a certain standard.

Similar to real life, in the world of networking, we are faced with many threats such as viruses, worms, trojans, attempted user privilege gain, data theft and denial of service attacks. An intrusion detection system may be deployed to allow users to log the existence of such threats. However, unlike real life, physical intrusion detection systems such as the fire alarm, no standard exists for testing or evaluating how well an IDS performs. To quote the work of Mell *et al.* (2003), they state that "while intrusion detection systems are becoming ubiquitous defences in today's networks, currently we have no comprehensive and scientifically rigorous methodologies to test the effectiveness of these systems (Mell, *et al*., 2003)." In other words, a standard accepted methodology is still a requirement in the evaluation of IDSs.

There are three main categories of IDS in use. They are widely referred to as host-based IDS (HIDS), network-based IDS (NIDS) and distributed-based IDS (DIDS). Although all three are equally important, it is apparent that the most widely used category of these devices are NIDSs. Therefore, the focus of this paper will be on this category. It is proposed that an evaluation of IDSs must consist of the inclusion of both attack and background traffic along with meaningful metrics for evaluation. Attack traffic allows us to see how well an IDS detects threats, whilst background traffic will allow us assess the accuracy in an IDSs detection method. By defining the correct metrics during the evaluation process we may then come to a meaningful conclusion.

Based on this hypothesis, this paper aims to present a framework which attempts to include all three requirements which have been listed to provide for an effective evaluation of NIDSs. A framework is proposed which allows for the evaluation of NIDSs using realistic attack network traffic and meaningful metrics of evaluation. Generation of benign network traffic is also taken into consideration. The framework presented is used to carry out a black box evaluation of the NIDS known as Snort (Sourcefire, 2009) and, based on the experiment findings, an analysis of the effectiveness of this framework is provided. Finally, this paper concludes by highlighting the areas of research which are currently most required in regards to the evaluation of IDSs.

2. Related Works

The framework presented in this paper has been developed based on existing methodologies, which include: the DARPA evaluation (Lippmann *et al.*, 2000), LARIAT evaluation (Rossey *et al.*, 2002), and TRIDENT evaluation (Sommers *et al.*, 2004). The first methodology employs an offline evaluation method whilst the latter two employ a real-time testing method. The overall goal of these evaluations was to provide a concise methodology for the evaluation of IDSs. As highlighted, there are three main requirements in the evaluation of IDSs: inclusion of attack traffic, inclusion of background traffic and meaningful metrics of evaluation. It was found that the existing methodologies, summarised in the following sub sections, have all attempted to meet this concept.

2.1 DARPA Evaluation

Developed by MIT Lincoln Labs, the main goal in the DARPA evaluation was to carry out a non biased measurement on the performance of various anomaly-based IDSs along with producing an evaluation data set which could be used by others in testing their IDSs (McHugh, 2000). Lincoln Lab set up a test network and, through the use of programs, to emulate a large number of workstations, and scripts, created synthetic background traffic mixed with attack traffic at certain periods of time (Brugger *et al.*, 2007). The traffic is then captured with a packet capture tool and saved as a data set. This data set could then be played back against the IDS under evaluation to assess its performance.

With the use of static data sets, repeatability in experiments is easily achieved and, furthermore, the data set is easy to obtain and free to download. However, some criticism has been made in regards to this evaluation, including the fact that the network traffic used may not be considered realistic enough to reflect a real-life network (Mahoney *et al.*, 2003). Furthermore, it can be stated that the presentation of the evaluation results may not be completely meaningful since only one form of metric is used to determine the performance of the IDS under test (McHugh, 2000) known as the ROC curve.

2.2 LARIAT Evaluation

The Lincoln Adaptable Real time Information Assurance Test bed (LARIAT) was designed as a follow up to the 1999 DARPA evaluation (Athanasiades *et al.* 2003) and as described by the authors of this evaluation "two design goals were established for LARIAT: (1) support real-time evaluations and (2) create a deployable, configurable and easy-to-use test bed (Rossey *et al.*, 2002)."

In this evaluation, an emphasis on automation, and ease-of-use is made. The LARIAT evaluation is implemented on top of a Java applet named NetworkDirector (Rossey *et al*., 2002). This acts as a *wrapper* in which a GUI interface is built and allows users to interact through the selection of menus and buttons rather than having to manually input commands.

Using the NetworkDirector, user input is only required in the initial stages of this evaluation methodology, in which they select specific "profiles" of both attack and background traffic. Upon selecting a profile, the system will automatically configure the emulated network and begin the process of the testing the IDS under evaluation.

Although this evaluation methodology is highly sophisticated, the main limitation is that the LARIAT evaluation is part of a United States Government funded project and not available for public use (Athanasiades *et al.*, 2003).

2.3 Trident Evaluation

In the Trident evaluation, the main aim of this work to allow for a variable mix between background and attack traffic (Sommers *et al.*, 2006). In other words, users may specify certain percentages of attack traffic and background traffic. As an example, users may specify a test with 10% attack traffic and 90% benign traffic to see whether the high level of background traffic will have any detrimental effect in the IDSs ability in detecting attacks.

This is achieved using the Malicious Traffic Composition Environment (MACE), which is a tool used for generating attacks on a network (Sommers *et al.*, 2004). MACE works by implementing the Python programming language in order to both write and invoke network attacks. Additionally, a flow level background traffic generator named Harpoon is used to create both TCP and UDP packets (Sommers & Barford, 2004).

Both MACE and Harpoon were developed during separate periods of time but are used in conjunction for the Trident framework (Figure 1). The usage of MACE and Harpoon allows for a fine control between benign and attack traffic along with mixing them in a realistic manner.

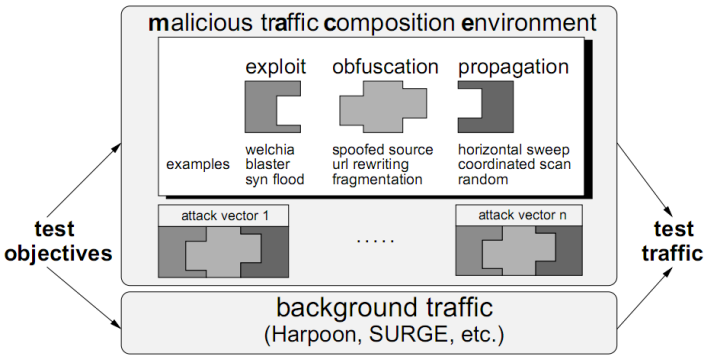


Figure 1 – Trident Framework (Sommers *et al.*, 2004)

One main critique should be made against this evaluation method, which is in regards to MACE. Although a wide taxonomy of attacks has been defined, it should be noted that these are attacks have been specifically coded and crafted for researchers to evaluate IDSs. In other words, what this evaluation may end up assessing is how effective an IDS is in detecting attacks generated by MACE but not attacks from programs found widely on the internet.

3. Framework Design and Methodology

Figure 2 provides an overview of the framework. Three main components are developed for this framework: the attack traffic component, background traffic component and evaluation metrics component.

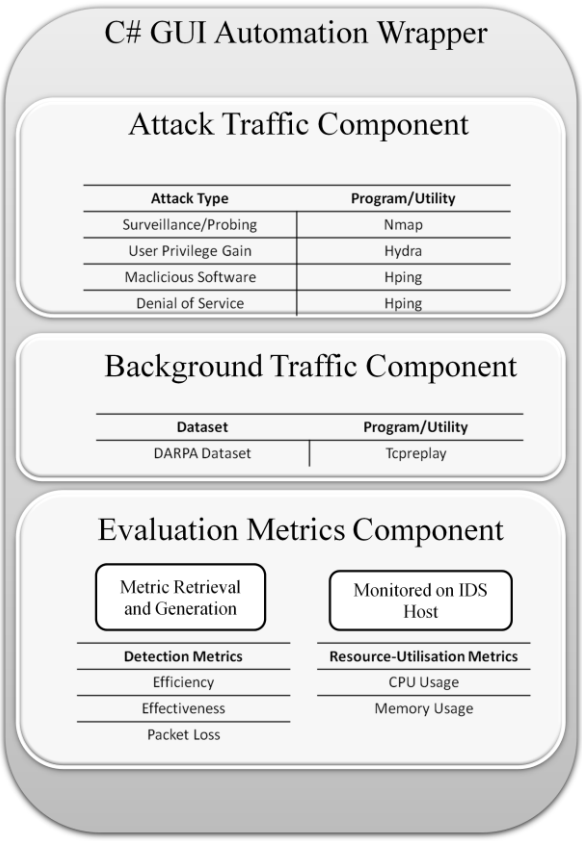


Figure 2 – Framework Overview

The attack traffic component allows us to invoke four main categories of attacks in real-time. The framework implements existing programs and utilities in order to achieve this goal. The background traffic component is used for the invocation of benign network traffic. This is achieved using a traffic playback tool using static data sets. The framework is capable of generating both forms of network traffic in conjunction with each other. The evaluation metrics component allows us to assess the performance of the IDS under evaluation based on measurable parameters.

On top of these three components, a graphical user interface coded in C# has been developed to “wrap” the framework into one single application. This allows for both ease-of-use and automation when carrying out the evaluation process.

3.1 Attack and Background Traffic Component Design

The Attack Traffic Component is implemented using a wide variety of existing tools and utilities available on the internet. It is believed that this technique allows for the most realistic method for the generation of attack traffic. The main advantage of this design choice is that real-time attacks may be carried out against an IDS. However, limitations – in terms of security - may be found if one was to attempt to evaluate an IDS against malicious software (worms, viruses and trojans for example). Therefore, malicious software threats are crafted using the utility Hping3 rather than using real executables of viruses. Table 1 provides taxonomy of the attacks which the framework is capable of carrying out.

Table 1 – Framework Attack Taxonomy

Category	Utility Used To Generate Attack
Surveillance/Probing	Nmap
User Privilege Gain	Hydra
Malicious Software	Hping3
Denial of Service	Hping3

Regarding the Background Traffic Component, achieving realistic and controlled generation of benign traffic is still very much an open issue. The closest available tool which will allow for realistic generation of background traffic is found in the work of Sommers *et al.* (2005) using Harpoon. Unfortunately, it was dismissed as not being realistic enough due to using the same port in every connection by Corsini (2009)**.** Furthermore, Harpoon is difficult to use as it requires a greater effort during both the installation and configuration process. NetFlow logs may ease the configuration process but in the case of them not being available, users must manually create their own topologies and configurations.

The framework presented in this work overcomes these issues through the use of static data sets for playback of background traffic using Tcpreplay, as developed by Turner & Bing (2002). Two main justifications exist for this choice. Firstly, the ease of repeatability will be achieved in playing back the traffic since the data set will notr change in each test. It will also allow for variable playback speeds in order to assess whether the IDS's performance, with regards to the metrics defined, will be impacted when exposed to high traffic volumes. The limitation present in this decision relates to the fact that the only publicly available data sets are those released by DARPA, meaning any evaluation carried using this framework at this point in time is restricted to this data set unless the user was to provide their own.

3.2 Evaluation Metrics Component Design

Knowing whether or not an attack was logged does not provide enough information when performing a concise evaluation of an IDS. For example, if an IDS were to raise a high number of false positives, we know that it is not very accurate in the detection of attacks but we do not know the reason for this occurrence. Therefore, we require a certain set of evaluation metrics which will provide meaning to the evaluation. This framework employs two main categories of evaluation metrics: detection-related metrics and performance metrics.

The detection related metrics include efficiency and effectiveness, which are derived from the work of Sommers *et al.* (2005), along with packet loss, derived from Graves *et al.* (2006). Efficiency is a measure of false-positives whilst effectiveness is a measure of false-negatives. In other words, we can assess whether the IDS under evaluation will trigger alerts against background traffic (false-positives) and whether it will miss real attacks (false-negatives). Figure 3 demonstrates the formulae used to work out each metric. In both equations, the calculated result closest to 1 is always better.

$$\text{Efficiency} = \frac{\text{TruePositives}}{\text{AllAlarms}}$$

$$\text{Effectiveness} = \frac{\text{TruePositives}}{\text{AllPositives}}$$

Figure 3 - Formula for working out Efficiency and Effectiveness Metric (Sommers *et al*., 2005)

Monitoring of packet-loss is important as it allows for assessing whether the IDS can successfully monitor all traffic even in high throughput. In the category of performance metrics, CPU usage and Memory usage of the system in which the IDS resides are

monitored. This allows us to see whether higher resource utilisation results in any detrimental effect to the IDS (such as detection of attacks). Specifically, a relation between detection metrics versus resource utilisation metrics can be established. Table 2 provides a summary of the evaluation metrics used in this framework.

Table 2- Evaluation Metrics used in Framework

Detection Metrics	**Description**
Efficiency	True-Positives / All Alarms
Effectiveness	True-Positives / All Positives
Packet Loss	The number of packets lost, as reported by the IDS
Resource Utilisation Metrics	
CPU Usage	Percentage of CPU used
Memory Usage	Percentage of memory used

4. Experiment Design

The Attack Traffic Component of the framework is capable of carrying out surveillance/probing, user privilege gain, malicious software and DoS attacks. In each test scenario of this experiment, each and every single attack that the framework is capable of carrying out will be invoked against a target machine. Background traffic will run in conjunction with the attacks using the Background Traffic Component. The Monday Week 1 1998 DARPA data set (M. L. Laboratory, 1998) is used in the playback of background traffic.

The only variation we apply to each instance of running the experiment will be the playback speed of background traffic. By carrying out the experiment in this way, we can assess whether the volume of traffic on the network will have any detrimental effects to Snort's detection abilities. Thus, a dynamic evaluation of the NIDS is achieved.

Each test run (a total of six) was carried out by invoking attack and background traffic. After a test run finished, the Evaluation Metrics component was used to automatically report the efficiency, effectiveness and packet loss metrics based on the logs produced by the IDS.

Along with noting the detection metrics, the resource utilisation metrics of CPU Utilisation and Memory usage are also logged. As we are running the experiment via a Linux environment, the *top* command was used to retrieve the resource utilisation results. Figure 4 provides a schematic of the experiment whilst Table 3 outlines the different traffic playback speeds for each test run.

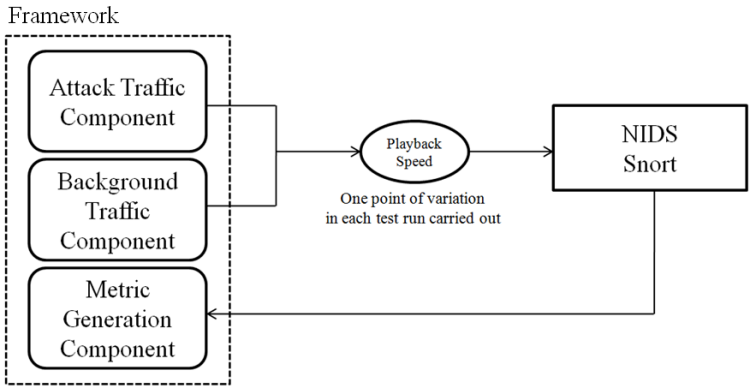


Figure 4 – Schematic of Experiment

Table 3 – Traffic Playback Speeds

Test Run	Playback Speed (Mbps)
1	20
2	40
3	60
4	80
5	100
6	120

4.1. Snort Configuration

The NIDS Snort is a signature-based IDS, therefore, custom rules must be crafted in order for it to detect network attacks. Furthermore, as we are applying background traffic in this evaluation, arbitrary rules must also be setup in order to assess whether they will be triggered when under evaluation (resulting in false-positives). To achieve this goal, the use of the Vulnerability Research Team (VRT) rule set provided by Sourcefire (2009) was used for the arbitrary rules, whilst custom rules were generated manually. A sample of a custom rule is shown in Figure 5 which is used for the detection of FTP brute force attacks carried out by the framework.

```
alert tcp any any -> $HOME_NET 21 (msg:"FTP Brute force Attack
Detected"; flow:  to_server,established; content:"PASS";  threshold:
type threshold, track by_src, count 3, seconds 1; sid:002;)
```

Figure 5 – Sample of Snort Custom Rule

4.2. Test Bed Description

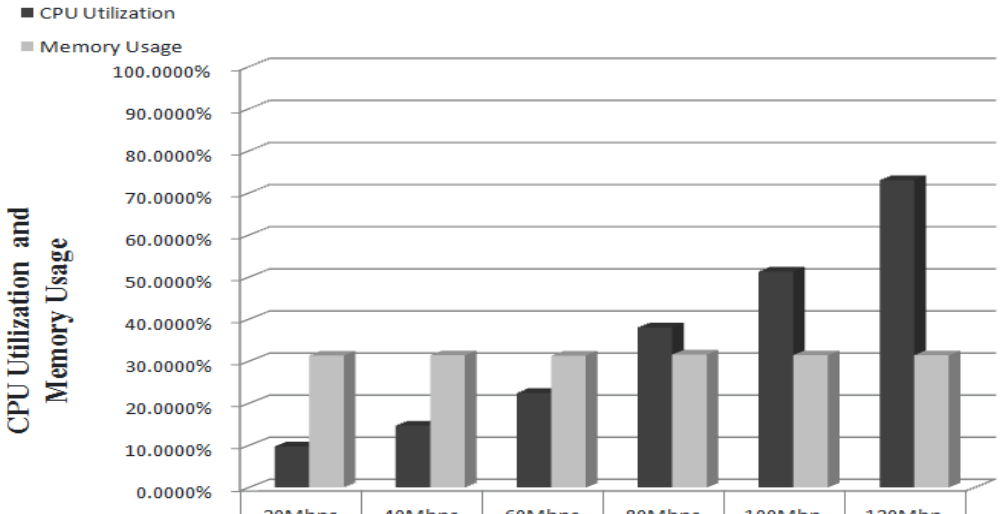
Virtual machines are used to create a private virtual network in order to conduct the experiment. The software VMware (version 6.0.2) (VMWare, 2009) has been used for this purpose. Three virtual machines are required: one machine to run the application, one machine to run Snort and one machine to act as the target of attacks. All three machines are running the Xubuntu Distribution of Linux using kernel 2.6.27. The three machines are connected to a virtual switch which is created from VMWare. The specifications for the three machines are presented in Table 4.

Table 4 – Specifications of Virtual Machines

Machine Name	Operating System	CPU (shared)	Memory
VM 1	Xubuntu Kernel: 2.6.27	Intel Core2 Quad Q6600 @ 2.40 GHz	512 MB
VM 2	Xubuntu Kernel: 2.6.27	Intel Core2 Quad Q6600 @ 2.40 GHz	512 MB
VM 3	Xubuntu Kernel: 2.6.27	Intel Core2 Quad Q6600 @ 2.40 GHz	256 MB

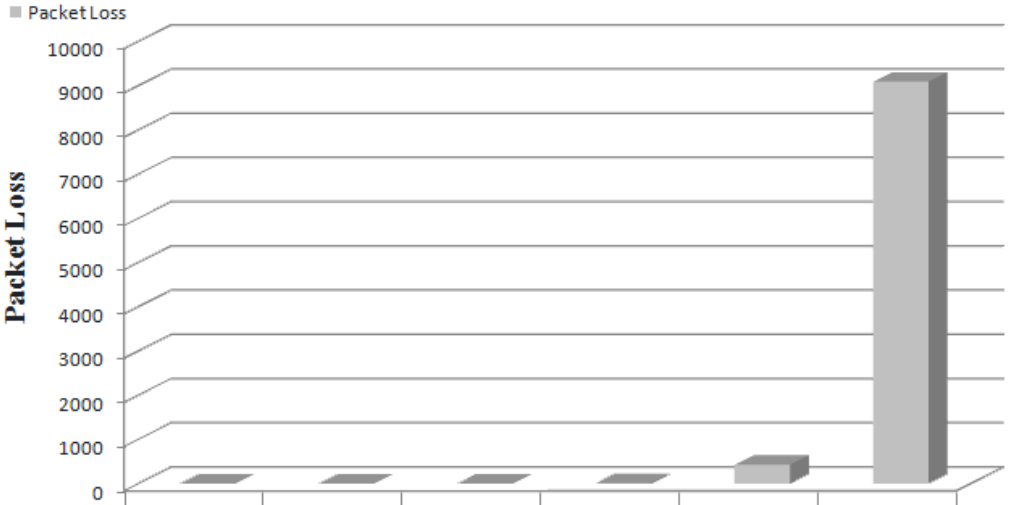
5. Results

Figure 6 presents the CPU and memory usage at different playback speeds, whilst Figure 7 provides the pack loss results. The efficiency and effectiveness results are presented in Figure 8.



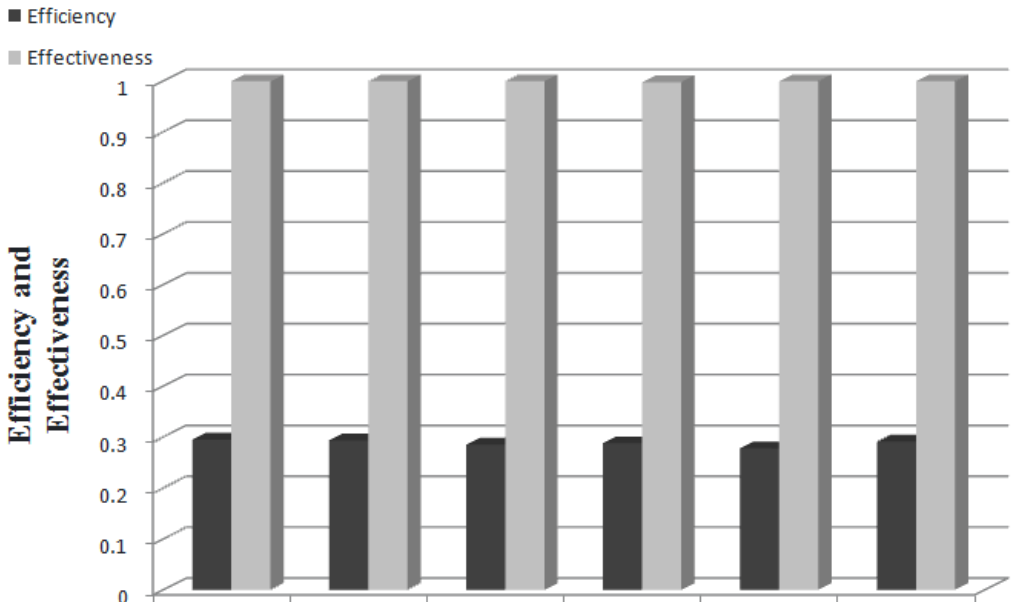
	20Mbps	40Mbps	60Mbps	80Mbps	100Mbps	120Mbps
CPU Utilization	9.7400%	14.7067%	22.4333%	38.0000%	51.3000%	73.0000%
Memory Usage	31.35%	31.41%	31.33%	31.58%	31.42%	31.41%

Figure 6 – CPU Utilisation and Memory Usage Results



	20Mbps	40Mbps	60Mbps	80Mbps	100Mbps	120Mbps
Packet Loss	0	0	0	13	428	9091

Figure 7 – Packet Loss Results



	20Mbps	40Mbps	60Mbps	80Mbps	100Mbps	120Mbps
Efficiency	0.2953	0.294	0.2854	0.2882	0.2783	0.2914
Effectiveness	1	1	1	0.9983	1	1

Figure 8 – Efficiency and Effectiveness Results

6. Analysis

In each of the figures provided, a comparison is drawn between the metrics defined against the variation in traffic throughput (both attack and background traffic combined). Figure 6 demonstrates that with an increase in the throughput of traffic, CPU utilisation also increases. One interesting result from Figure 6 shows that Snort appears to use the same amount of memory even at the highest playback speed. In producing such a result, two main strengths of Snort can be highlighted: (1) Snort is resourceful in its use of memory even in high volumes of network traffic and (2) Snort's effective utilisation of the CPU helps alleviate the need for memory usage during the detection of attacks.

In Figure 7, a comparison is drawn between packet loss and the speed of playback of the network traffic. Specifically, this figure shows whether Snort is able to analyse all traffic on a network even at very high throughput. As the figure shows, from 20 to 60 Mbps, Snort is able to cope with high throughput of traffic but at 80 Mbps playback and above, significant packet loss occurs. This figure highlights a limitation in Snort, since the occurrence of packet loss can result in attacks slipping past Snort and also cause difficulties during digital forensic investigations as described by Graves *et al*. (2006). Security, too, is an issue as even a single packet slipping past Snort may be an issue as the next paragraph describes.

Figure 8 shows a comparison between traffic throughput against both efficiency and effectiveness metric (see Section 3.2 for description on these two metrics). All but one test instance reported an effectiveness metric of 1, meaning all attacks were detected successfully. This shows that Snort is highly effective in the detection of attacks. However, it should be noted that there was one exception to this result during the experiment. At 80 Mbps playback, it can be seen that an effectiveness of 0.99 was reported. In investigating this result, it was discovered that Snort had dropped an attack packet along with a few arbitrary background traffic packets. Once again, this highlights Snort's limitation in its ability to analyse all packets, an attribute which is highly desirable for an IDS.

Perhaps the most unexpected data from this evaluation is the results produced by the efficiency metric. In each test run, approximately 30% of alerts raised were false-positives. Very little variation of these results were found, and upon running a test in which only background traffic was generated, it was found that the VRT rule set used would raise false-

positive alerts due to the background traffic used (which, in this case, was the DARPA 1998 data set).

The purpose of this experiment was to conduct a black-box evaluation using the framework implemented, and without a greater degree of analysis on both the detection mechanisms used by Snort, individual analysis of the VRT rule set and the behaviour of traffic in the data set, it cannot be said for certain whether it is Snort or the data set used which is at fault at this point in time. However, it does highlight some issues in regards to background traffic generation and Section 7 demonstrates the future work required in this area.

7. Future Work

When analysing the results from Figure 8, it can be seen limitations are still found in background traffic generation. In this work, static data sets were applied for the evaluation process. It is apparent that data sets are not very viable for the evaluation process since their very nature is static. Furthermore, the only widely available data set is the one produced by DARPA which is quite outdated at this point in time. As the results from this work has shown, a high number of false-positives are raised when the DARPA data set is applied in testing, therefore further investigation must be applied in this area of work in order to reach a conclusion as to why this is the case.

It is believe that using live traffic generation methods may allow for a more feasible evaluation to take place. However, there is always the possibility that the traffic being generated may not even resemble traffic seen in a physical network. This issue was demonstrated in the work of Ranum (2001), in which the author summarises that traffic generators can produce packets with pseudorandom TCP frames which may confuse an NIDS into assuming some form of denial-of-service attack is taking place due to its stream reassembly implementation. Furthermore, in the case of signature based IDSs specifying rules which look at traffic on specific ports, pseudorandom packets that are generated may be completely omitted therefore resulting in better than expected performance in comparison with real-life networks.

Therefore, it is proposed that further research must be undertaken in the area of background traffic generation. Although multiple live traffic generation tools exist already, one of the key problems faced is that not enough research has been carried out in attempting to compare each of the different traffic generation methods against each other. The following recommendations from this paper are:

- Provide a concise and in-depth review of existing traffic generation tools such as, for example, Harpoon and Swing (Vishwanath *et al.*, 2006) in order to understand how they work and the exact type of traffic they attempt to generate.
- Compare each existing traffic generation method in regards to how realistically it mirrors traffic seen in real life networks.
- Through research and experimentation, attempt to provide a non-biased conclusion on which method of generating background traffic, if any, is best suited for testing a IDSs which would produce the most realistic evaluation results.

8. Conclusion

A concise methodology for the evaluation of IDSs is still lacking. Being a last line of defence for computer security threats, it is highly critical that we are able to evaluate these devices to a certain standard. This paper has stated that there are three main requirements in evaluating an IDS: attack traffic, background traffic and meaningful metrics of evaluation. A framework was developed in order to cater for this need.

From the experiment carried out, this paper has shown that the framework is capable of generating both attack and background traffic. Variable playback speeds were achieved, meaning that a comparison between the resource utilisation metrics and detection metrics could be achieved. Furthermore, as the results demonstrated, the metrics applied allow for meaningful evaluation of an IDS to occur. Thus, it can be stated that both attack traffic

generation and defining meaningful metrics have now been met in the IDS evaluation process. The limitation we are faced with is in the area of background traffic generation.

Future work must be carried out in the area of background traffic generation. This includes an in-depth review of existing traffic generation tools along with providing a non-bias comparison between each tool. In doing so, we may reach a conclusion as to what method is best suited for evaluating an IDS and, more importantly, to reach a point whereby a standard widely accepted IDS evaluation methodology may be fulfilled.

9. References

Athanasiades, N., Abler, R., Levine, J., Owen, H. & Riley, G. (2003), Intrusion detection testing and benchmarking methodologies', *Proceedings of First IEEE International Workshop on Information Assurance*, 2003. IWIAS 2003. pp. 63-72.

Brugger, S. & Chow, J. (2007), An assessment of the DARPA IDS Evaluation Data set using Snort, Technical report, *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.*

Corsini, J. (2009), Analysis and Evaluation of Network Intrusion Detection Methods to Uncover Data Theft, Master's thesis, *Edinburgh Napier University, Edinburgh, UK*.

Graves, J., Buchanan, W., Saliou, L. & Old, J. (2006), Performance Analysis of Network Based Forensic Systems for In-line and Out-of-line Detection and Logging. *Proceedings of the 5th European Conference on i-Warfare and Security (ECIW), Academic Conferences Limited*.

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J. & Das, K. (2000), The 1999 DARPA off-line intrusion detection evaluation, *Computer Networks* **34**(4), 579 - 595.
Mahoney, M. V. & Chan, P. K. (2003), An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection, *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, Springer-Verlag, pp. 220-237.

McHugh, J. (2000), 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* **3**(4), pp. 262-294.

Mell, P., Hu, V., Lippmann, R., Haines, J. & Zissman, M. (2003), “An overview of issues in testing intrusion detection systems”, *National Institute of Standards and Technology ITL* , Technical report, NIST IR 7007, Gaithersberg, MD.

M.L Laboratory. (1998), *MIT Lincoln Laboratory: Information Systems Technology,* [online], http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998/training/week1/monday.tar

Ranum, M. (2001), Experiences benchmarking intrusion detection systems, NFR Security White Paper.

Rossey, L. M., Cunningham, R. K., Fried, D. J., Rabek, J. C., Lippmann, R. P., Haines, J. W. & Zissman, M. A. (2001), LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed, Submitted for publication, *IEEE Proc. Aerospace Conference*, pp. 2671-2682.

Sommers, J. & Barford, P. (2004), Self-configuring network traffic generation, *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ACM, New York, NY, USA, pp. 68-81.

Sommers, J., Yegneswaran, V. & Barford, P. (2004), A framework for malicious workload generation, *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ACM New York, NY, USA, pp. 82-87.

Sommers, J., Yegneswaran, V. & Barford, P. (2005), Toward Comprehensive Traffic Generation for Online IDS Evaluation, Technical report, Department of Computer Science, University of Wisconsin, Madison.

Sommers, J., Yegneswaran, V. & Barford, P. (2006), Recent Advances in Network Intrusion Detection Systems Tuning, *CISS '06: Proceedings of the 40th IEEE Conference on Information Sciences and System*, Princeton, NJ, USA, pp. 1490-1495.

Sourcefire. (2009), Snort [computer software], http://www.snort.org/

Turner, A. & Bing, M. (2009), Tcpreplay Tool, [computer software], http://tcpreplay.sourceforge.net/

Vishwanath, K. V. & Vahdat, A. (2006), Realistic and responsive network traffic generation, *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, New York, NY, USA, pp. 111-122.

VMWare. (2009), VMWare, [computer software], http://www.vmware.com/