Performance Analysis of Network Based Forensic Systems for In-line and Out-of-line Detection and Logging

J R Graves, Prof W J Buchanan, L Saliou, Dr J Old
Centre for Mobile Computing and Security, Napier University, Edinburgh, UK
j.graves@napier.ac.uk
w.buchanan@napier.ac.uk
l.saliou@napier.ac.uk
j.old@napier.ac.uk

Abstract: Network based forensic investigations often rely on data provided by properly configured network-based devices. The logs from interconnected devices such as routers, servers and Intrusion Detection Systems (IDSs) can yield important information, which can be used during an investigation to piece together the events of a security incident. A device, such as a router, which performs its intended duties as well as logging tasks, can be defined as having *in-line* logging capabilities. A system that only performs these duties, such as an IDS, can be described as an *out-of-line* logging system.

The usefulness of these logs, though, must be compared against the impact that they have on the systems that produce them. It is thus possible to introduce a detrimental burden on inline devices. This can thus reduce the capability of the device to provide core functionality, and, the extra evidence generated could place an increased burden on the forensic investigator. Therefore, when configuring network devices, the security practitioner is the key to producing a careful balance between security, performance and generated data volume.

This paper outlines an intensive experiment to compare and contrast different logging schemes. These tests are placed within the scenario of a forensic investigation, which involves extensive data logging and analysis. The metrics compare CPU utilisation, bandwidth usage, memory buffers, usefulness of these records to the investigation, and so on. The two logging systems examined are the Cisco 20x series based routers, for *in-line* logging capabilities with Syslog, and the IDS Snort for *out-of-line logging*. This work provides an empirical perspective by plotting the footprint that this logging scheme has on the core network infrastructure, thus providing a proposed optimal logging approach for a network, along with the comparative merits of *in-line* and *out-of-line* auditing systems.

Keywords: Logging, Digital Forensics, Network Management, Network Performance, Intrusion Detection

1. Introduction

Digital Forensics is an emerging and important area of research (Palmer, 2001). It has established itself as a major force in the design, implementation, and deployment of IT systems. This paper focuses on some of the issues surrounding Network Forensics, which is concerned with the investigation and analysis of the data gathered from a network. This information can then be used to discover how, and, why, a security incident took place. These investigations are used routinely to solve security incidents, such as for automated worm propagation, fraud, and hacking events. Thus, the techniques used to gather data for analysis in such investigations is ideal for exploration.

This paper concentrates on the tools used to gather information about the data traversing the network at any given point. With a networking infrastructure offering many different filtering options, such as the use of Access Control Lists (ACLs) on routing devices, there are a multitude of logging and auditing options. These capabilities can filter, and collect, any form of data destined to, or from, a host on the network. With the increase in demand for technologies to observe and collect network data, two such categories of system have emerged:

- Native, or *in-line,* networking equipment, such as routers and firewalls may block and filter traffic, and, in addition, report the number of data packets that have been successfully restricted.
- Dedicated, or *out-of-line*, devices, such as Intrusion Detection systems (IDSs), provide a mechanism for the system administrator to gather situational awareness regarding the network. This can then be used to provide real-time security incident response information, or may be used as a later time as evidence in some other investigation (Buchanan *et al.*, 2005; Saliou *et al.*, 2005).

This work proposes a framework for testing such devices, and thus establishing a set of benchmarks for both *in-line* and *out-of-line* auditing systems. This framework will provide quantitative information about the impact that these services have on the devices in question. The quality and quantity of this data will be assessed, and finally the effectiveness of this framework will be assessed.

2. Background and Methodology

Testing procedures and other quantitative analysis are recommended for the tools and techniques used by forensic investigators (Casey, 2002). Such testing regimes can improve the reliability and repeatability (Carney *et al.*, 2004) of these data gathering techniques and improve the likelihood that this data will be accepted as *bona fide* evidence (Carrier, 2003). Yet these testing recommendations only cover matters such as data integrity, assurance and authentication (Casey, 2002), and affect tools that are used in the post mortem analysis of IT systems. Yet, as Buchanan *et al.* (Buchanan *et al.*, 2005) highlight, security and forensics are an interlinked concept. The basis of this theory is that, the output of the most commonly deployed security systems, such as IDS, form the basis of any investigation. Thus, there is commonly some form of effect of logging and forensic constraints. Yet, the impact that these security, and thereby auditing, systems have, tend to focus primarily on the performance overheads of common filtering and mitigation systems (Al-Tawil *et al.*, 1999). Within this work, the link between security and performance is well known. So far there is very little analysis of these systems with relation to the impact that the logging and auditing capabilities have on a system, and how effective these systems are at what they do and provide.

This paper aims at starting such an analysis of these issues, from a forensic perspective. With reference to existing work, and our hypothesis, we can assert that the main needs of a network can be summarised within Figure 1.

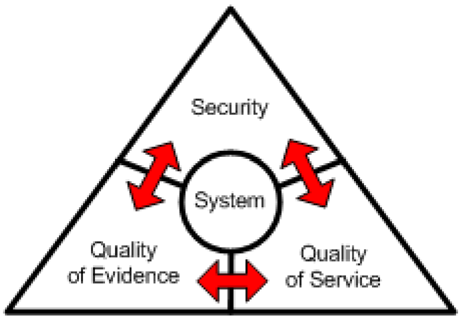

Figure 1 – The Balance of Needs

- **Security** – Systems should be able to restrict malicious activity and provide situational awareness for those in administrative charge. When improperly configured, security can introduce a detrimental effect on the system, and be perceived as a barrier to productivity (Viega, 2005).
- **Quality of Service (QoS)** – An IT system should be able to provide core services that match the objectives of an organisation.
- **Quality of Evidence** – The evidence gathered by logging and auditing systems should be complete, authenticated and reliable. Wherever possible, the proper configuration of these auditing systems should ensure superfluous data is not produced, and, thus, not have a negative impact on both the system producing it, and the investigation.

Thus, if an IT system implements all of these features, there must be a balance. If this balance is lost, there may be a detrimental effect on that system. Yet lack of evidence can have a detrimental effect on an investigation. If evidence is dropped or missed by auditing devices then the evidence collected by this device cannot said to be complete. This missing evidence could contain the crucial information that would prove the innocence or guilt of an individual. Markatos et al. (Markatos *et al.,* 2002) highlight the amount of data that can be lost by the IDS Snort. Like most of the work in this field, it approaches this topic from a security perspective. Yet, the implications for a forensic investigation are grave. We therefore present an analysis of this particular problem from an investigative perspective.

Additional motivation for this work is highlighted by Mocas (Mocas, 2003), who states, that any work conducted in the field of digital investigation is unique, in that the researcher must be conversant with theoretical research and, must obtain a knowledge of the realities of investigation, whether it be procedural, or technical. Thus, in the development of this framework, we hope to bridge the gap between both of the essential fields of theoretical research, and practical application. This work will form the basis of a set of metrics and recommendations that may be used by practitioners in order to create a careful balance between these factors.

The framework presented in Section 3, is an adapted and improved version of the testing framework presented in (Saliou *et al.*, 2005). This framework was deployed in order to test a network router, while agent systems attempted to reconfigure the device in the face of an attack. The framework presented in this paper improves upon the previous design by implementing a testing strategy that can push a device to its breaking point. In addition, the traffic load placed upon the device more accurately represent the types of traffic found on a network. This allows for more realistic testing scenarios. In line with the practical and open philosophy outlined by Mocas, all of the software components, with exception of the Cisco Internetwork Operating System (IOS, the software that runs as standard on Cisco hardware) software, are open source, and can be freely used.

3. Design and Implementation

3.1 Configuration

Figure 2 shows the logical configuration of the framework. It consists of four Pentium III machines with 100Mbps network cards, and a single Pentium 4 Windows XP machine acting as a Syslog machine. A single switch provided separate VLAN access that acted as an inner and outer network. For this particular framework, a single Cisco 2611XM router and Cisco 3550 switch were used.

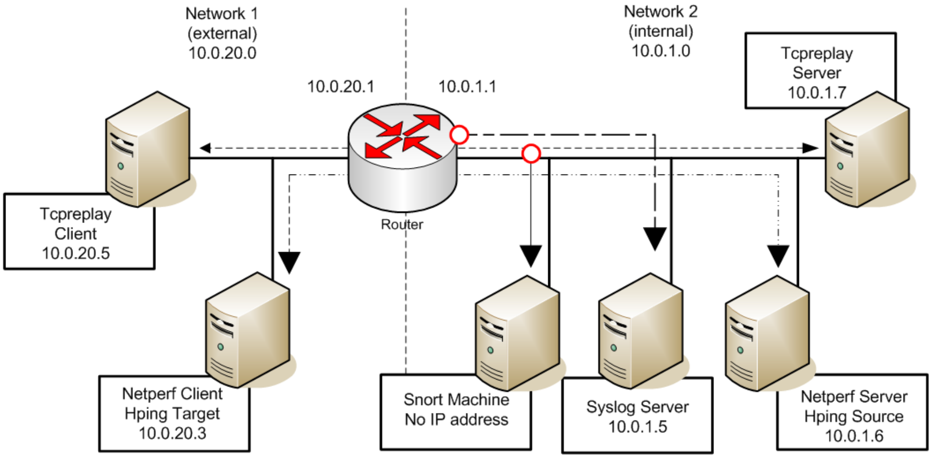

Figure 2 – Logical Configuration of Test System

Router-firewalls are active devices that are capable of filtering and dropping network packets. The logic instructing them to do so, in the case of Cisco devices, is contained in rule definitions, or ACLs. The decision to use an ACL that blocks spoofed traffic was made. The choice of these ACLs allow for the testing of a rule set that is likely to be deployed on an actual network device, and are standard ACLs that are recommended by Cisco (Cisco Systems, 2004, p.122). In addition, these rules supply a large enough range of addresses to fully test the Syslog logging feature without introducing further stress on the router. Overall, the more rules that are added to the ACL, the higher an impact this has on the device (Lyu *et al.*, 2000). A sample set of the ten ACLs deployed on the device are outlined in Figure 3.

The command *log*, which is highlighted, is an optional command that tells the IOS to log the instances of traffic matching that rule to the desired source. The router can log to a local source in memory, or, as in this case, to a dedicated Syslog machine. Conversely, if the option log is absent, the IOS will not log instances of that particular rule. Syslog (Lonvick, 2001) is a common standard for forwarding and logging messages in networks. Since it is a standard protocol, it can be implemented in a heterogeneous environment.

```
access-list 150 deny   ip host 255.255.255.255 any            log
access-list 150 permit ip any  10.0.0.0        0.0.255.255    log
```

Figure 3 – A sample of the IOS ACLs used

Snort (Roesch, 2006) (ref. Figure 2), as the *out-of-line* device, is tasked with analysing, and processing all network traffic, in the same manner as the router, and, like the router, will have to filter and log all malicious traffic based on a set of rules. The Snort rule set contains precisely the same ten filtering values as the ACL used on the router. Although there are database logging options available for Snort, in this instance we use the standard, file logging mechanism. Figure 4 is an example of the Snort rules used.

```
alert ip 16.2.1.0/24 any -> any any (msg:”16.2.1.0  net Trigger”;)
alert ip 0.0.0.0/8   any -> any any (msg:”0.0.0.0/8 net Trigger”;)
```

Figure 4 – A sample of the Snort rules used.

The software used to test the hardware components has to provide realistic network traffic. Thus in order to satisfy this requirement, Tcpreplay (Turner, 2006) was used. This tool allows data to be replayed across devices and networks from data that has been previously captured from a network using a tool such as the UNIX tcpdump utility. By using Tcpdump, in conjunction with existing network captures, highly realistic network conditions can be simulated as the information being sent across the network can be rewritten in order to match the specific logical requirements. In addition, network files being played from this utility can be replayed at differing speeds. Thus, it is possible to simulate different network conditions without the need to recapture data at different speeds. The data sets used are those collected by DARPA for the 1998 offline evaluation challenge (Lippmann *et al.*, 2000). This data contains a mix of different network types, including HTTP, FTP and Telnet traffic, along with other, more exotic packets. Unlike the main attack data that contains data with malicious items, the traffic used for this work was the training data from Monday, Week 1.

In accordance with the testing methodology outlined in Section 2, the metrics used to evaluate the devices under test must provide information about the stress experienced, and also the functionality provided under the different network loads. In addition, the amount of evidence collected and quality of this evidence will be assessed. The metrics are defined as:

- **Router CPU Usage** - This metric was gathered by using the native CPU monitoring function present within the Cisco IOS and using a console connection between the device and the monitoring computer. By querying the IOS, the CPU usage for the past 60 seconds can be displayed, and, an average of the percentage over one minute is used to calculate this metric.
- **Network Latency -** This performance metric is gathered using the average response time from the Hping tool (Sanfilippo, 2004), which acts very much like the generic Ping utility, in that it sends an ICMP packet from one device to another. The round trip of this packet gives an indication of network latency as this round trip time tends to increase as network throughput decreases.
- **Network Bandwidth –** Is calculated using the Netperf utility (Jones, 2006), which uses two software agents. One acts as a server, the other as a client. The packets sent between these two are used to calculate the bandwidth available in Mbps.
- **Snort Metrics -** The number of packets analysed and dropped by the system at each network load was collected from the output of Snort’s own auditing system.
- **Additional Metrics –** The quantity and quality of evidence collected by each device will be assessed.

3.2 Experiment Design

Metric collection occurs while each device is subjected to differing network traffic speeds. Since the ACL and Snort rules used have 10 rules, 10 streams of data are played by Tcpreplay. Each stream of data is designed to activate each Snort rule or ACL. The network speed was set to 10 Mbps. Tcpreplay is capable of replaying traffic at the original recording speed. Therefore, the speed 'nominal' relates to the original speed at which the traffic was recorded. The speeds were then increased incrementally to 1.0, 2.5, 5.0 and 7.5 Mbps. A test stream of 10 Mbps was omitted as it was considered that a network at its full capacity is unable to provide a service worth testing. An additional test is included where the network speed is increased to 100 Mbps. This experiment is designed to explore whether there is a similar trend with higher baseline network speeds.

Each experiment is conducted over a 15 minute period, to ensure the same amount of data was sent in each test run. The experiment scenarios, with their designation, are:

- **Baseline (BL)** – Traffic which is played across the device at the five speeds, with no ACLs and no Syslog logging capabilities.
- **Router with ACL, No Logging (ACLONLY)** - The router is loaded with the ACLs, yet the log command is absent, and logging capabilities are disabled on the router.
- **ACL and Logging All Traffic (LOGALL)** – The router is loaded with the ACLs, and the keyword log is present after every rule declaration, and, with the full Syslog logging capabilities. Thus, the device logs instances of both friendly, and malicious, traffic.
- **ACL and Logging Attack Traffic Only (LOGAO) -** The router is loaded with the ACLs and the keyword *log* is present only after the rules to filter undesirable traffic, and full Syslog logging capabilities are configured. Thus, the device only logs instances of malicious traffic.
- **100 Mbps ACL Syslog Test (100MbLOGAO)** – The router and switch are configured to operate at 100Mbps. The ACL is loaded with the *log* command for malicious traffic. Full Syslog logging capabilities are configured. All traffic speeds are increased accordingly with the new baseline network speed. A new baseline for this test is taken with the higher speeds for comparison.
- **Snort (Snort)** – Snort was tasked with filtering the traffic in the same manner as the ACL. The traffic to be monitored was the attack traffic, thus it follows the **LOGAO** model.

4. Framework Evaluation

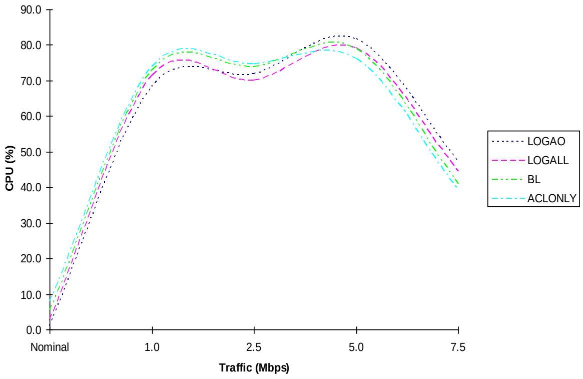

Figure 5 – CPU data (10 Mbps)

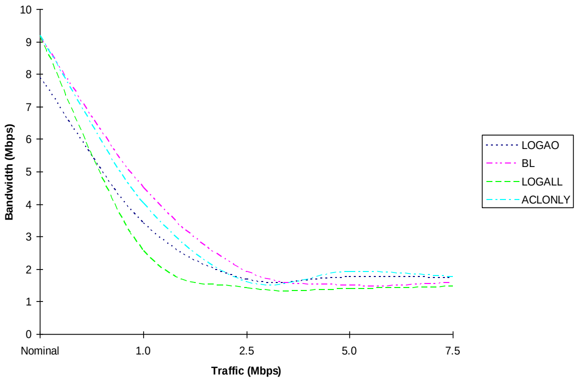

Figure 6 – Netperf Data (10 Mbps)

It can be observed that there is a weakness in the testing framework, when conducting the tests at 10Mbps. This can be observed particularly in Figure 6, yet is apparent in all of the graphs (ref. Figure 5, 6 & 7). After the traffic speed exceeds 2.5 Mbps, the results merge together. This is likely to be due to the fact that the traffic throughput is too high. At the higher speeds, packet loss occurs and both the traffic generator and the router have problems handling data at these speeds.

Additionally, we can observe that the CPU metric is not conclusive piece of evidence when observing the effect of this type of test. The trend shown in Figure 5 highlights that CPU usage on this particular type of router does not give an accurate representation of the load a device is experiencing, as all of the metrics are close together.

5. Quantitative Analysis

5.1 In-line device Evaluation

Figure 7 shows the latency data from each experiment. A low latency means higher bandwidth and thus better network availability. The test **LOGALL** is significantly different from all of the other tests, with an initially similar latency to the rest, but diverging as much as 55% from the **BL** test at 2.5 Mbps. This shows the effect of a log everything policy on the device. The other traffic profiles are similar. When looking at their profiles at the 2.5 Mbps mark, it can be seen that that they offer a fairly good insight into the latency of each policy. Yet, after this point, the issues discussed in Section 4 skew the results.

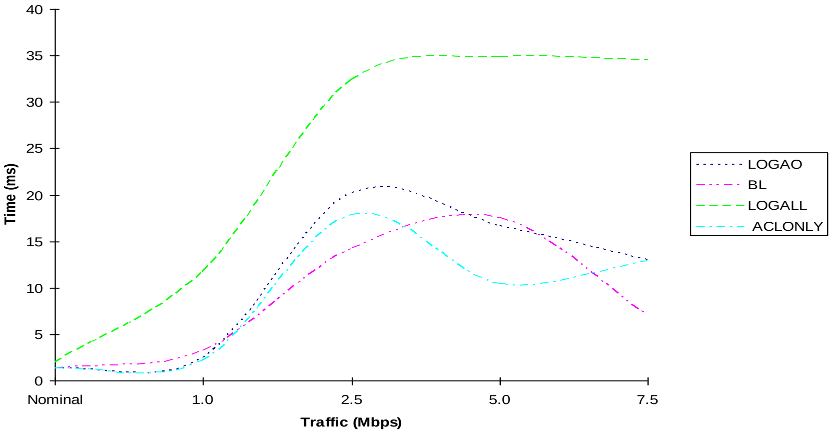

Figure 7 – Latency Data (10 Mbps)

We observed that the Netperf throughput measurements give the most detailed perspective, and therefore is the metric that will be the focus of the rest of this analysis. Figure 8 is a more detailed view of the graph produced by netperf. The merged data produced by device stress has been omitted.

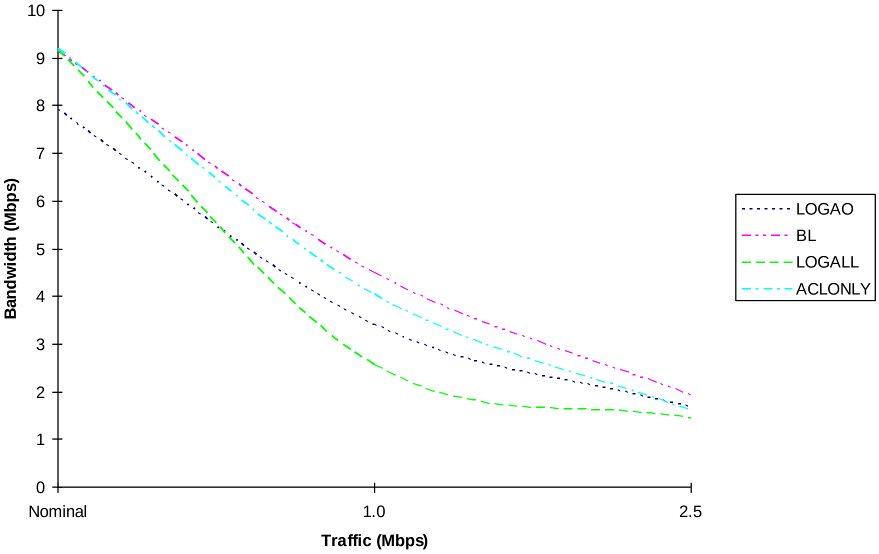

Figure 8 – Graph charting 10Mbps Netperf Experiment before Equipment Failure

The first thing to note about the results is the immediate difference between the **BL** and **LOGAO** test. We can see an immediate 13.4% drop in bandwidth across the router. Yet, the drop in bandwidth continues by following the same trend as the baseline, and, up until the device stress, the trend remains almost the same.

When comparing the **BL** and **LOGALL** experiments, although there is no immediate decrease in the available bandwidth, as the traffic throughput increases, there is a significant decrease in the amount of bandwidth available. This trend does not follow the same trend of the **BL** data. The difference is most noticeable with the throughput at 1 Mbps. At this point, there is a 43.1% drop in throughput.

When comparing the **LOGAO** and **LOGALL** experiments, the effect of a log all policy and a log attack data only policy can be seen. Although the policy to log only attack data has a significant effect on the device immediately, overall, the policy to log all data has the most detrimental effect on the device. The difference between these two, at its most noticeable is 24%.

The test **ACLONLY**, was performed to ensure that the loss of bandwidth was not primarily due to the ACLs deployed on the device. Lyu et al. (Lyu et al., 2000) have already established this as a factor to be considered when deploying these configurations. Thus when comparing the **ACLONLY** test and the **BL**, it can be observed that there is a 16.4% effect on bandwidth at the 2.5 Mbps throughput, but it is not as great as the effect caused by the tests **LOGAO** and **LOGALL**, which have logging enabled.

Finally, Figure 9 shows a comparison of the netperf data from the 100Mbps baseline test and the **100MbLOGAO** test. This graph shows that the performance of the device does suffer at higher speeds, yet the trend is different from that experienced during the 10 Mbps tests. When the traffic reaches 50Mbps, the device stops responding, and in effect, all network provision is lost. This test data shows us the failure point of the device under a high load. This is something that the 10Mbps test does not show.

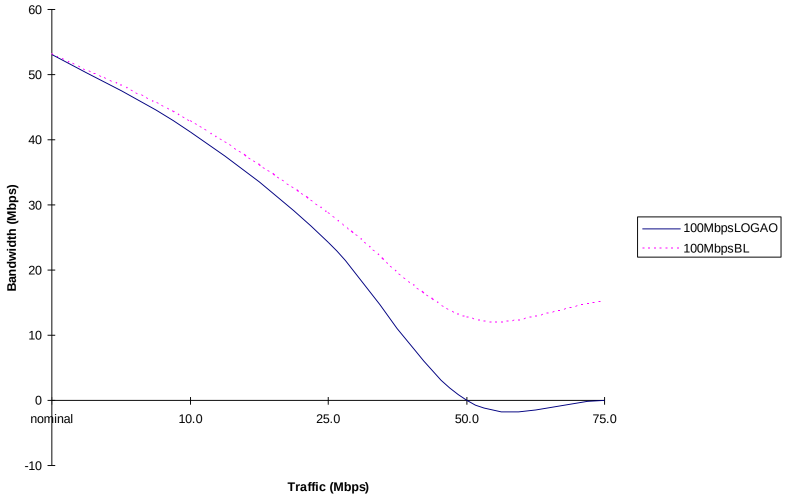

Figure 9 – Netperf data (100 Mbps)

5.1 Out-of-line device Evaluation

When analysing the data from the **Snort** test results (ref. Figure 10), it is difficult to directly compare them to those provided by the *in-line* tests. Thus, Snort needs to be judged by its ability to log all relevant data, and what that data can be used for. A qualitative analysis of this data is provided in the next Section. With the nominal traffic, Snort drops, or does not analyse 33.5% of the data it receives. This increases significantly to 96.2% when the traffic is at its highest. The loss of this data could have a significant impact on a given investigation, and in this instance, it can be said that Snort does not provide a complete record of network evidence. Table 1, which shows the size of log directory against packets dropped and analysed, highlights this trend. Although these poor results may be attributed to the speed of the machine, it is surprising to see such a poor performance at even nominal speeds.

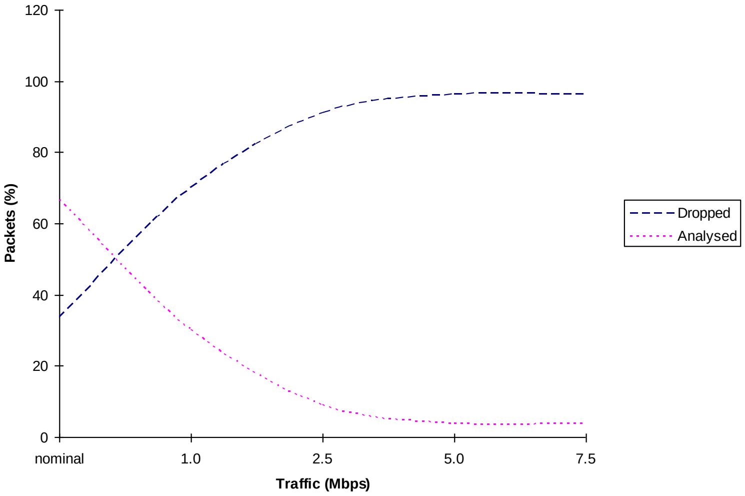

Figure 10 – Packets Analysed and Dropped by Snort

Table 1 – Comparison of Directory Size and Dropped vs. Analysed Packets

Test (Mbps)	Dir Size (MB)	Dropped (%)	Analysed (%)
nominal	47	33.5	66.5
1	435	69.9	30.1
2.5	344	91	8.9
5	247	96.3	3.7
7.5	253	96.2	3.8

6 Qualitative Analysis

This Section establishes how useful the data logged by either system would be for an investigation. The kind of investigation that would be conducted when spoofed traffic is detected originating from an internal, trusted network, would involve tracing the spoofed packet back to the originating host. The information needed to do so would be the physical MAC address of the machine sending the packet. Snort can be configured to log this information, along with any other details of the packet, yet the overhead that this incurs is highlighted in Section 5.1. Although Snort did drop a substantial number of malicious packets, for those that were logged, some form of follow up investigation would have been possible (Figure 11). For investigations that require more data to be logged for analysis (Casey, 2004), the performance of Snort may decrease even further, as data relating to the application layer would need to be analysed and logged, as well as the network and transport data.

```
[**] 192.0.2.0 net trigger [**]
02/17-14:08:49.291973 0:C0:4F:A3:58:23 -> 0:D:29:62:BB:81 type:0x800
len:0x436
192.0.60.182:2129 -> 192.0.112.149:25 TCP TTL:64 TOS:0x0 ID:5488 IpLen:20
DgmLen:1064 DF
***AP*** Seq: 0xB149828E  Ack: 0x8702B20  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 11 – Snort log data.

Figure 12 show the messages that are logged to Syslog. This is one example of a number of messages logged. When compared against the data logged by Snort (ref. Figure 11), the information provided does not form any basis for back-tracking, as all that is provided is the originating and destination IP address, the ACL that the packet contravenes, and a time stamp. There are no additional logging options available, as what is provided is a synopsis of the number of packets that have contravened the ACLs on the device. Yet the information provided, which is highlighted in Figure 12, does not correlate with the counter information provided by the IOS when queried via the console connection. Thus the aggregate information sent to the Syslog server does not give an accurate picture of the number of packets being processed by the device.

```
 2006-01-26 12:14:49    Local7.Info 100.0.1.1   1563: 1d01h: %SEC-6-
IPACCESSLOGP: list 150 denied tcp 169.254.69.104(0) ->
196.254.114.207(0), 10 packets
```

Figure 12 – Syslog log data

7 Conclusions and Future Work

This paper shows that *in-line* logging techniques do have a significant effect on the devices implementing them. It has been shown by using a minimal, recommended rule set, deployed on routing devices, reporting to a Syslog server. This effect can be seen when logging, not only all instances of policy circumvention, but also malicious data. It highlights that the policy to log only the malicious data places an instant 13.4% throughput performance loss on the device, and that the policy to log all malicious data has an even more damaging effect on the device, where the maximum throughput loss observed is 43.1%.

From a forensic perspective, the Syslog data shows that the information supplied by this *in-line* device is ineffective, not only for a back tracking investigation, but, also in its precision at reporting the number of ACLs circumvented. Yet, not all Syslog data needs to be discounted. In many circumstances, situational data relating to the function of the router, such as system and administrative information can provide useful data in determining whether these core devices have been compromised.

The results from the *out-of-line* Snort analysis shows that, although the data provided is useful, and provides detailed information about the packets that are logged, the packet analysis has a detrimental effect on Snorts ability to analyse and collect data. A packet loss of 33% at nominal throughput, increasing to a 96.2% loss at the highest throughput, represents a significant amount of data loss. Thus, Snort cannot be relied upon to provide a complete record of evidence.

The framework presented in this paper does show the effect that different traffic throughput has on devices, and could be used for benchmarking production networks. This paper shows that in order to fully evaluate the balance of needs for an organisation (ref. Figure 1), a strong testing methodology of Design, Testing, Evaluation and Refinement is needed.

Therefore the recommendations from this paper are:

- *In-line* logging techniques need to be restricted to provide minimal information about the device itself, and not the traffic it is processing.
- That further work on o*ut-of-line* devices needs to be conducted to establish how much data is lost by these auditing systems, and the effect that this data loss can have on an investigation.
- Further work needs to be conducted in order to establish those of *out-of-line* devices that can provide sufficient, complete evidence for an investigation. Improvements need to be made so that detailed analysis can take place.
- An improved framework would include fine grained control of traffic speeds and a variable malicious-to-friendly traffic ratio.

8 Acknowledgements

The authors would like to thank Professor Rao Bhamidimarri, Dean of Faculty of Engineering and Computing, for making funding available for research, as well as Professor Jon Kerridge and Bob Rankin, former and current Head of School respectively, for assigning grants to this research program.

References

Al-Tawil, K. and Al-Kaltham, I. A. (1999) "Evaluation and testing of internet firewalls", *Int. J. Netw. Manag.*, Vol. 9, No. 3, pp 135-149

Buchanan, W., Graves, J., Saliou, L., Sebea, H. A. and Migas, N. (2005) *Agent-based Forensic Investigations with an Integrated Framework*, 4th European Conference on Information Warfare and Security, University of Glamorgan, UK, pp 47-52

Carney, M. and Rogers, M. (2004) "The Trojan Made Me Do It: A First Step in Statistical Based Computer Forensics Event Reconstruction", *International Journal of Digital Evidence*, Vol. 2, No. 4

Carrier, B. (2003) "Defining Forensic Examination and Analysis Tools Using Abstraction Layers", *International Journal of Digital Evidence*, Vol. 1, No. 4

Casey, E. (2002) "Error, Uncertainty, and Loss in Digital Evidence", *International Journal of Digital Evidence*, Vol. 1, No. 2.

Casey, E. (2004) "Network Traffic as a Source of Evidence: Tool Strengths, Weaknesses, and Future Needs", *Digital Investigation*, Vol. 1, No. 1, pp 28 – 43.

Jones, R. (2006) "Netperf Homepage", [online], Rick Jones, http://www.netperf.org/netperf/NetperfPage.html

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J. and Das, K. (2000) *Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation*

Lonvick, C. (2001) "RFC 3164 - The BSD Syslog Protocol", [online], Network Working Group, http://www.faqs.org/rfcs/rfc3164.html

Lyu, M. R. and Lorrien K.Y. Lau (2000) *Firewall Security: Policies, Testing and Performance Evaluation*, The Twenty-Fourth Annual International Computer Software and Applications Conference, pp 116-121

Mocas, S. (2003) *Building Theoretical Underpinnings for Digital Forensics Research*, Digital Forensics Research Workshop.

Palmer, G. (2001). *A Road Map for Digital Forensic Research - Report from the First Digital Forensics Research Workshop (DFRWS)* (No. Technical Report DTR-T001-01 Final). Utica, New York: Air Force Research Laboratory, pp 1-48.

Roesch, M. (2006) "Snort - the de facto standard for intrusion detection/prevention", [online], Sourcefire, http://www.snort.org/.

Saliou, L., Buchanan, W. J., Graves, J. and Munoz, J. (2005) *Novel Framework for Automated Security Abstraction, Modelling, Implementation and Verification*, 4th European Conference on Information Warfare and Security, University of Glamorgan, UK, pp 303-311.

Sanfilippo, S. (2004) "Hping Wiki", [online], Salvatore Sanfilippo, http://wiki.hping.org/.

Systems, C. (2004) *Fundamentals of Network Security*, Cisco Press, Indianapolis, pp122.

Turner, A. (2006) "Tcpreplay: Pcap editing and replay tools for *NIX", [online], http://tcpreplay.sourceforge.net/.

Viega, J. (2005) "Security---problem solved?" [online], ACM Press, http://doi.acm.org/10.1145/1071713.1071728.