Analysis of Firewall Performance Variation to Identify the Limits of Automated Network Reconfigurations

Lionel Saliou, William J. Buchanan, Jamie R. Graves, & Jose Munoz
Centre of Mobile Computing and Security, Napier University, Edinburgh, UK
l.saliou@napier.ac.uk
w.buchanan@napier.ac.uk
j.graves@napier.ac.uk
j.munoz@napier.ac.uk

Abstract: Security in computer networks is typically passive, static, and reactive. This is typically due to most networking devices being rule-based, and when updates are necessary, they are normally done manually. Ultimately, the social and hierarchical structure of an organisation should be visible within the configuration of networks. Hence, it is desirable for a distributed system to be capable of reconfiguring itself in a timely-manner to reflect changes in policy, in practices, and in the social hierarchy, such as the promotion of a member of staff, or in the face of a security threat, such as in malware propagation.

This paper builds on the concept of an automated mitigation and reconfiguration system for networked devices, and evaluates key firewall system performance tests. These could be important in defining the criteria for the success of this type of security implementation. It thus defines a range of experiments, which evaluate firewall parameters, such as number of rules, and their position in relation to performance metrics, such as CPU utilisation, bandwidth consumption, and network latency. The paper also includes tests with up to 65,000 rules, and presents results on the positions of the rules, such as on the incoming and outgoing ports, and the effect of different network throughputs.

It concludes that networks can be made more resilient, under heavy network loads and large rule sets, if rule sets are applied on the outgoing ports. It also shows evidence that configuration interfaces are the performance bottleneck for multi-agent systems that may use these to reconfigure network equipments dynamically.

Keywords: firewall management, computer network defence, dynamic reconfiguration, mitigation, firewall performance metrics.

1. Introduction

This paper defines, and presents results for, a range of experiments designed to identify the limitations, in terms of performance, of network firewalls. It also critically evaluates the performance of software agents that could be interfaced with the equipment to address the lack of dynamic enforcement of such equipment, and their limited abilities to evolve. In conclusion, it advocates a scientific, and repeatable, approach to network security evaluation.

Most networked systems are rule-based, static, and, therefore, are often difficult to evolve. In addition, such systems are often vulnerable from internal intruders, or anyone capable of exploring the network infrastructure, because flaws, or lapses, remain unfixed for long periods. Some researchers, such as Glenn (2003, p12), argue that modification tasks which are typically required in these situations, are much more costly in terms of human resources, planning and so on, than the fresh deployment of new systems. Alternatively, flexible and dynamically configurable equipment, such as Active Network (AN), exists. However, they do not offer the same level of performance as traditional equipment, and hence are seldom deployed in corporate networks (Campbell *et al.* 1999, p7). These issues are particularly relevant when organisations use an integrated security framework that aims to ensure the implementation of security requirements, as well as enabling networked system with the ability to thwart threats in real-time, without hindering the organisation’s objectives (Saliou *et al* 2005, p306).

2. Related research work

This paper defines that a hybrid architecture, based on multi-agents system (Santana Torrellas *et al.* 2003, p369), could be used to meet: security requirements; performance; and, in addition, provide dynamic reconfiguration capabilities. A critical factor in the success of such architecture is the ability of the multi-agent system (MAS) to assess the current situation, as well as to anticipate the outcome of possible changes. More importantly, to effectively combat threats, and thus take appropriate action, the MAS needs to be aware of the time required between a decision being made, and its actual effective deployment on a device.

One common argument against the adoption of security requirements is that it is costly in terms of performance. In practice, the possible effects on the network include:

- Increased delays to access services.
- Undesirable stress level on intermediate equipment, thus possible limited tolerance to traffic load.
- Reduced available throughput.

Although most firewall models describe the traffic filtering operations, they do not indicate the potential performance bottlenecks (Kamara *et al* 2003, p215). As for an evaluation carried out on live devices, this is often incomplete. In Al-Tawil *et al.* (1999), for instance, tests are designed with ease-of-distribution in mind and are based on a combination of software tools that search for known vulnerabilities, and human observations as opposed to performance issues. Furthermore, the emphasis of Al-Tawil *et al.* (1999) is not on the performance offered by the system, but rather the focus on how well firewall devices participate in the protection of individual network hosts. Consequently, the drawback of such an approach is that the results lack key parameters, such as load sensitivity, and other attributes that might be considered essential for the organisation to fulfil its objectives (Saliou *et al.* 2005).

Lyu *et al.* (2000, p116) argue that the intuitive belief about the trade-off between security and performance does not hold, and propose to evaluate the usefulness of different security policy levels, and their impact on the overall network performance. Their proof, unfortunately, only includes latency and task-completion metrics at slow network speeds. In addition, their focus appears to be mostly the evaluation of software security tools.

An enhanced view is that fine-grained security requirements could require large firewall rule sets. Consequently, the hypothesis is that the more rules there are, the lower the performance, and in Lyu *et al.* (2000, p117) the number of rules used is low for modern day application. Furthermore, with respect to models, such as proposed by Kamara *et al.* (2003, p215), there are no indications of the effects of rule-set position. For instance, findings on these and their modelling, could be used to refine the security requirements of an organisation that uses an integrated security framework (Saliou *et al* 2005, p307).

3. Firewall basics and configuration

Network firewalls are one of the most widely deployed devices used to filter network traffic. The detail of the communication that is allowed, or not, is defined by rule sets which are collections of individual statements. Ideally, these rule sets should reflect the policies adopted by the organisation (Al-Saer *et al.* 2004, p3). Generally, each statement makes decisions on the communication's attributes, such as:

- **Source address**: the host, or the network, the packet is from.
- **Destination address**: the host, or the network, the packet is intended to reach.
- **Protocol in use**, such as TCP or UDP.
- **Service requested**, such as Web (HTTP), e-mail (SMTP), and so on.

In practice, firewalls have multiple ports, and Kamara *et al.* (2003, p215) summarise firewall's operations as: the reception of network packet on the incoming port; integrity verification;

evaluations against rule sets; and evaluation against the available network routes. After which the data packet is transferred to the outgoing port, where it could be once more subject to checks against the enabled rule sets, before being released onto the destination network.

On Cisco devices, such as the one used in these experiments, rule statements are typically implemented using Access Control Lists (ACL). Figure 1 is a simple example of a rule set, which set is designed to allow Web (HTTP) traffic on the local network, and prevent all other type of traffic. The ACL number defines the rule set to which each statement belongs.

```
Access-list 101 permit tcp 172.16.0.0 0.0.255.255 any eq 80
Access-list 101 deny ip any any
```

Figure 1: ACL rule set snippet

Cisco devices support multiple rule sets, which can be applied to packets which are either inbound, outbound, or both. Figure 2 shows the information that would be given when the rule set in Figure 1 is applied on an interface for inbound packets.

```
interface FastEthernet0/0
  172.16.0.1 255.255.0.0
  ip access-group 101 in
  no shutdown
```

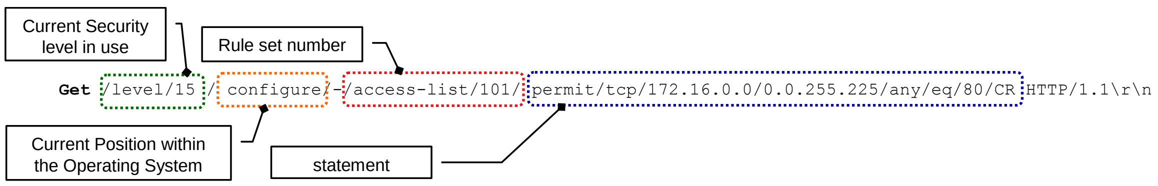
Figure 2: Rule set applied to inbound traffic

3.1 Typical configuration interfaces

Network devices can typically be configured using protocols, such as HTTP, Telnet, TFTP, Secure Shell (SSH), and others. As this paper presents deployment agents that use the HTTP and Telnet protocols to establish communication with Cisco devices, this section outlines details of these methods of interaction.

3.1.1 HTTP Service

When a device supports the HTTP reconfiguration service, its configuration can be altered using a HTTP client. Typically, commands are applied by constructing each command bit-by-bit before committing it to the device. It is not compulsory to apply command in this manner; indeed, any command is accepted as long as the proper HTTP request is sent. Figure 3 shows an example HTTP request that adds the first statement of rule set presented in Figure 2, to the device.



```
Get /level/15 / configure/-/access-list/101/permit/tcp/172.16.0.0/0.0.255.225/any/eq/80/CR HTTP/1.1\r\n
```

Current Security level in use
Rule set number
Current Position within the Operating System
statement

Figure 3: Snippet of adding statement with HTTP

3.1.2 Telnet Service

This service allows for the modification of the configuration in a computer terminal-type environment. With this service the correct position within the hierarchy of the operating system must be assumed, otherwise the command might be invalid within its scope. Thus, compared to the usage of HTTP, a Telnet-based agent must incorporate additional commands to access the correct level, before issuing directives. Unfortunately, the Telnet service can only handle one character at a time, thus, commands are normally separated by a series of individual characters, and sent in sequence, followed by a carriage return/line feed character sequence.

4. Experimental Setup

A single network architecture is used to evaluate both the reconfiguration agents and the performance of the firewall devices. Similarly to Al-Tawil *et al.* (1999) and Lyu *et al.* (2000), the test environment uses the **D**evice **U**nder **T**est (DUT) as the entry point of all traffic crossing the network. The details of the elements, both in terms of software and hardware, used in these experiments is available in Appendix A.

4.1 Agent benchmarking

The objectives of the agent benchmark is to:

- Measure the time to deploy a new rule set, which is achieved by tasking the agents to deploy the same rule set of 1000 rules.
- Determine the best condition for deployment agents to operate, which is achieved by repeating the experiment at different network speeds, without any background traffic.
- Assess which application level protocol, such as HTTP or Telnet, is best suited for the task. This is achieved by monitoring the CPU usage of the DUT, and measuring the amount of data exchanged between the agents, and the DUT.

4.2 Firewall device evaluation

The test environment is composed of two distinct networks interconnected by the DUT, where one is setup to initiate, the main traffic flow, and is illustrated on the left-hand side of Figure 4, whereas the second network acts as a sink for the traffic. The environment has a traffic generator (Turner, 2005) and, which creates realistic network conditions. In order to ensure the repeatability of this series of experiments, the traces used are publicly available from the Lincoln laboratories, and were collected during the first week of 1998 DARPA evaluation project (Lippmann 2000, p583). Furthermore, the traffic generator allows for replaying the traffic at different rates. In these experiments, this rate is typically a percentage of the network bandwidth available to the host running the traffic generator. Henceforth, the **network-load level** refers to the bandwidth used by the traffic generator.

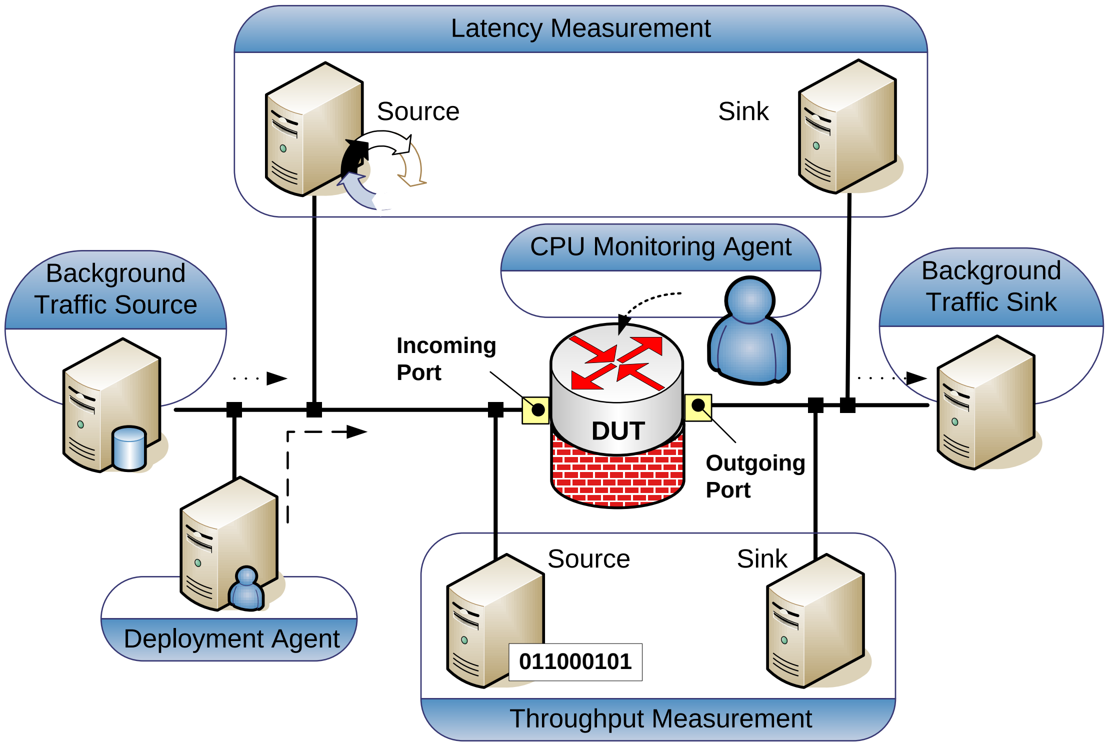


Figure 4: Test environment logical layout

A node using HPing (Sanfilippo, 2005), then measures the network latency by contacting the remote host across the firewall device. This utility measures the network latency more accurately than a standard Ping utility. Furthermore, it can also measure the time required to reach services, as opposed to the node, only. In this experiment, HPing is configured to measure the Round-Trip-Time (RTT) required to reach a service on the remote host, and throughput measurements are achieved with the Netperf utility (Hewlett-Packard, 2005) between two nodes placed on either side of the DUT.

It is likely that firewall manufacturers enable their devices with algorithms that are able to summarise, or optimise, rule sets once loaded in memory. In order to promote a scientific approach in these experiments, it is essential to make the method repeatable, and mitigate the effects of such algorithms, as much as possible. Otherwise, comparisons with other equipment, or devices from other manufacturers, and so on, would be difficult. Consequently, the DUT has to evaluate each packet against every single statement of the rule set. This is achieved by having rules which test IP addresses that will never appear in the network traffic, and permits the measurement of the full impact that the rule set has on the device's performance. Appendix B provides a sample of the rules used in this experiment implemented using Cisco ACLs.

The tests are repeated for:

- Network speeds and modes supported by the device: 10Mbps; 100Mbps (for half and full duplex modes).
- Network loads: traces played at recorded rate: 10%; 25%; 50%; and 75%; of the available network bandwidth.
- No rules present on the device, which is referred to as the **baseline**.
- Rule set of 65,000 statements enabled on the incoming port.
- Rule set of 65,000 statements enabled on the outgoing port.

5. Experiments Results

This section presents a selection of the results obtained during the experiments.

5.1 Deployment results

Table 1 shows the results rule set of 1000 statements, which has 40,092 characters. It shows that higher network speeds do not permit the agents time to deploy firewall rules rapidly. In terms of bandwidth efficiency, the Telnet-based deployment agent performs poorly, as it typically sends only one character per packet, and thus this communication places a heavy burden on the CPU. On the other hand, the HTTP-based deployment agent sends one complete statement per request, but it would seem that this request still has to be interpreted by the DUT's operating system. This, however, does not seem to be the case for the Telnet agent. Hence, despite a lower footprint in terms of CPU usage and bandwidth utilisation, the HTTP-based deployment agent is much slower than its Telnet version for deploying rules.

Table 1: Benchmark results

Network Speed (Mbps)	**Agent Type (protocol)**	**Bandwidth Usage – Agent only (byte)**	**Total Bandwidth Consumption (byte)**	**Average Packet size (byte)**	**Elapsed Time (sec.)**	**Firewall CPU usage (%)**
10	HTTP	213,245	1,792,591	213	212.51	53
10	Telnet	1,247,644	3,833,479	62	56	89.4
100	HTTP	213,245	1,792,903	213	212.5	53
100	Telnet	1,225,991	3,833,329	62	56	89.4

5.2 Firewall benchmark analysis

5.2.1 Low network speed results

Figure 5 shows how the latency across the DUT is affected by the varying amount traffic, that the device has to handle. Without any rules on it, the latency only presents a minor increase, but placing the rules on the incoming port of the DUT makes the latency increase after 25% of the maximum throughput. Interestingly, placing the rules on the outgoing port provokes an increase after 50% of the maximum throughput. Overall, the latency does not reach a point where the network is made unusable, thus the experiment shows that the device copes well at low network speeds.

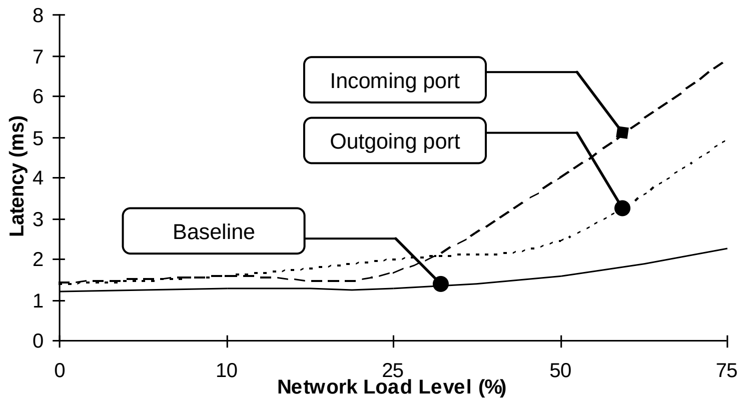


Figure 5: Latency benchmark (10Mbps)

Figure 6 shows the DUT CPU usage against the network-load level, and suggests that, in all three cases, the trends are almost similar and nearly linear. The effect of deploying rules is obvious, especially on the incoming port. It also suggests that the CPU usage is approximately related to the amount traffic that flows across the DUT.

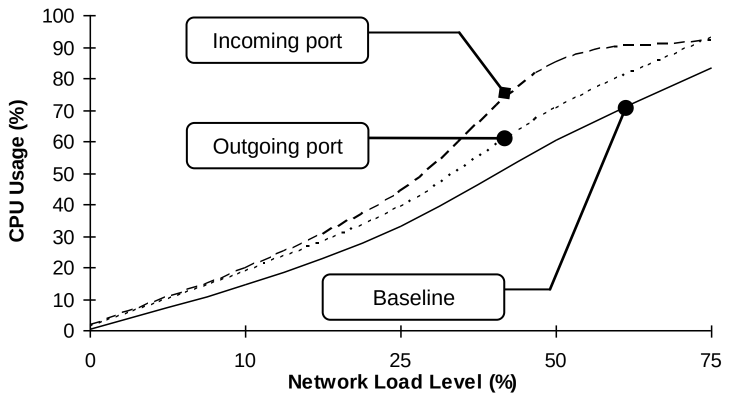


Figure 6: CPU benchmark (10Mbps)

Figure 7 shows the throughput performance, and its variation during the tests. In this case, the placement of the rules on the DUT has no noticeable effects, but the results allow the measurement of the footprint of the rule set as being the difference between the baseline reading, and those obtained with rules being deployed. These are almost always consistent throughout the benchmarking process.

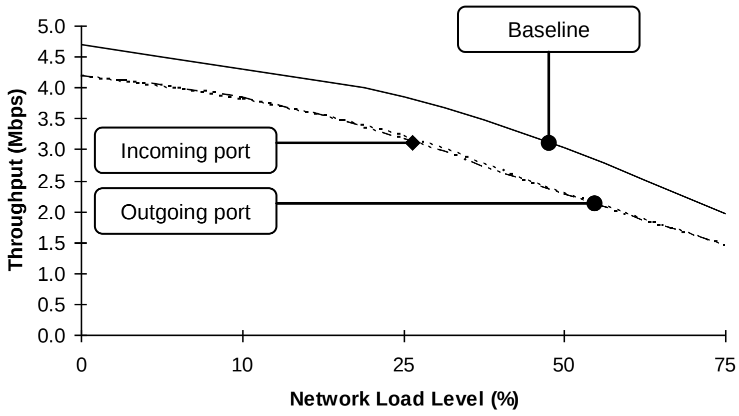


Figure 7: Throughput benchmark (10Mbps)

5.2.2 High network speed results

For rules deployed on the incoming port, the latency increases dramatically after the 10% of the maximum network throughput. The difference between the results obtained with rules on the outgoing port, for instance, are so significant that the scale of the difference between the readings is not suitable for interpretation based on a graphic. Thus, the results on the latency are not presented in this section. It is, however, noteworthy to mention that latency problems caused many throughput tests to fail.

Figure 8 shows the CPU usage of the DUT during the high network speed tests. In contrast to the same tests being conducted at slower speeds, this benchmark does not allow a measurement of the overhead incurred by the rules. In addition, in all situations, it would seem that the DUT has a limited tolerance to high network load.

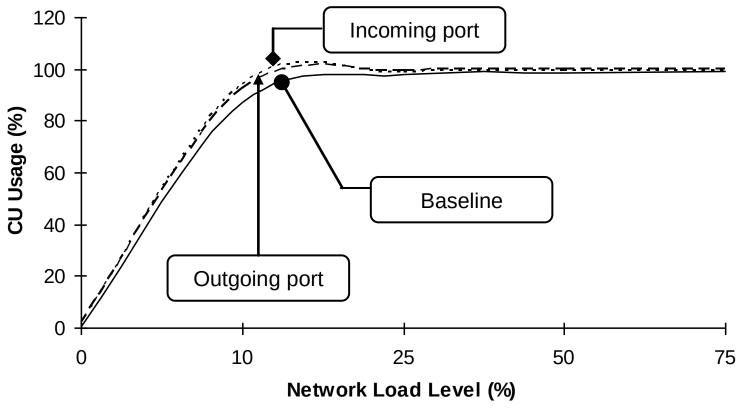


Figure 8: CPU benchmark (100Mbps)

The throughput measurements, shown in Figure 9, demonstrate that some level of performance can be ensured in high-speed networks with large firewall rule sets, provided that the correct port is chosen. Indeed, the degradation of performance when the rules are deployed on the incoming port is so significant that after the 25% of the maximum throughput, the network is completely unusable. In contrast, a rule set applied on the outgoing port makes the network more resilient. Arguably, the flat region of the graphic, 50% of the maximum network throughput, and beyond, is due to the software and hardware limits being reached.

All network devices need to buffer packets before processing them. It is expensive, in computation terms, to move packets from the incoming buffer to the outgoing one. Traditional models, such as Kamara *et al.* (2003, p125), propose that it is preferable to apply rule sets on inbound communications. This is acceptable for low throughput, as shown in Figure 7, as the buffer is less likely to over flow, and cause rejected packets, as the firewalling process has a long enough time to service the buffer. At high throughput, firewall checking struggles to cope with incoming data throughput, and, thus, result in low throughput (high packets loss), as shown in Figure 9. An improved procedure is, thus, at high throughput, to allow packets to be buffered in the main memory, which is typically larger than either the in or out port buffers, and to apply the firewalling process on outbound communications.

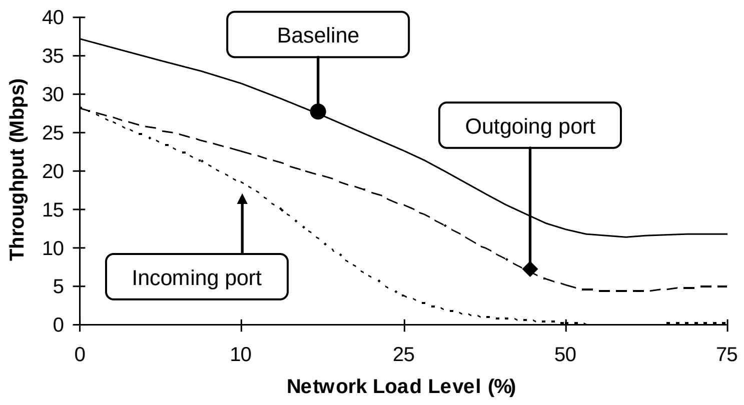


Figure 9: Throughput Benchmark (100Mbps)

6. Conclusions

This paper has established that the application of rule sets and throughput has a direct impact on firewall performance, and that the intuitive belief that rule sets are best applied on the incoming port, does not always hold. This practice, unfortunately, could incur dramatic performance degradation leading to the network being unusable for high throughput. It has also been shown that this could be the case for as little as 25% of the maximum throughput. Hence, network resilience can be enhanced by placing a firewall rule set on the outgoing port of a firewall. In the situations where rule sets must be present on the inbound port, this paper recommends that the rules are optimised, by using methods such as presented in Al-Saer *et al* (2004). Furthermore, results on CPU usage and latency demonstrate that readings should always be compared against baseline records. The presence of rule sets seldom creates significant differences with the baseline, thus trends and values could be misleading.

The results in the experiments correlate with some of the findings presented by Lyu *et al.* (2000, p121) and Briesemeister *et al.* (2003, p74), where it would appear that networks are more resilient even under heavy burden, both in terms of security and traffic, at low network throughput. Briesemeister *et al.* (2003, p67) argue that networks are built to fulfil objectives, which could include high network throughput and fine-grained security, and thus adequate network architecture must be selected ensure survivability.

The paper also presented findings on the effects of using agents to deploy the firewall rule set in a dynamic manner. Experiments have shown that the HTTP and Telnet interfaces used to configure firewalls represent a significant performance bottleneck for an agent-based system that would be used to thwart security threats in real-time, or deploy new security requirements. Results also suggest that if a network is equipped with a multi-agent system to provide flexibility and dynamic enforcement, lower network throughputs constitute a more suitable environment for the agents to operate.

Most models of firewall are focused on the logical attributes, hence methods to automate test procedures must be researched. This will allow a common framework for devices under test, and thus obtain data for the creation of models oriented on quality metrics of firewalls.

7. Appendix A: List of equipment used during the experiments

Firewall device:
- Cisco c2600MX series
- Cisco IOS 12.3 (27), release software fc3.

Network Switch:
- Cisco Catalyst 3550 series
- Cisco IOS 12.1(13) EA1a, release software fc1.

Traffic Generation Nodes:
- Pentium III 700MHz, 128MB of RAM, 3 Com 10/100Mbps compatible Ethernet card
- FreeBSD 5.4 Operating System
- TCPReplay version 2.2.0, 1
- DARPA trace from the 1st Monday of the 1st week of the year 1998 without ARP preamble

Latency Measurement Nodes:
- Pentium III 700MHz, 128MB of RAM, 3Com 10/100Mbps compatible Ethernet card
- FreeBSD 5.4 Operating System
- Hping version 2.0.0r3, 1(Emitter only)

Throughput Measurement Nodes:
- Pentium III 700MHz, 128 of RAM, 3 Com 10/100Mbps compatible Ethernet card
- FreedBSD 5.4 Operating System
- Netperf version 4a

Agent Node:
- Pentium IV 2.8GHz, 512MB of RAM, 3 Com 10/100Mbps compatible Ethernet card
- Microsoft Windows XP SP1 Operating System
- Microsoft .NET Framework 1.1
- Agents are written in C#

8. Appendix B: Experiments' rules sample

Rule Number	
1	access-list 65 deny host 177.129.1.56
2	access-list 65 deny host 118.226.112.42
3	access-list 65 deny host 124.84.62.221
4	access-list 65 deny host 189.119.171.195
5	access-list 65 deny host 33.83.183.49
.	...
.	...
.	...
64999	access-list 65 deny host 65.84.52.14
65000	access-list 65 permit any

Figure 10: Sample of rules used during the experiments

9. 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-Saer, E. S. and Hamed, H. H. (2004) "Modelling and Management of Firewall Policies", *IEEE Transactions on Network and Service Management*, Vol. 1, No. 1, pp 3 - 13

Al-Tawil, K. and Al-Kaltham, I. A. (1999) "Evaluation and testing of internet firewalls", *International Journal of Network Management*, Vol. 9, No. 3, pp 135 - 149

Briesemeister, L., Lincoln, P. and Porras, P. (2003) *Epidemic profiles and defense of scale-free networks*, ACM workshop on Rapid malcode, Washington, DC, USA, pp 67-75

Campbell, A. T., De Meer, H. G., Kounavis, M. E., Miki, K., Vicente, J. B. and Villela, D. (1999) "A survey of programmable networks", *ACM SIGCOMM Computer Communication Review*, Vol. 29, No. 2, pp 7 - 23

Glenn, M. (2003). A *summary of DoS/DDoS Prevention, monitoring and Mitigation Techniques in a Service Provider Environment* (White Paper): SysAdmin, Audit, Network, Security Institute, pp 1-30

Hewlett-Packard (2005) "Netperf", [online], http://www.netperf.org/netperf/NetperfPage.html

Kamara, S., Fahmy, S., Schultz, E., Kerschbaum, F. and Frantzen, M. (2003) "Analysis of Vulnerabilities in Internet Firewalls", *Computers & Security*, Vol. 22, No. 3, pp 214 - 232

Lippmann , R., Haines, J. W., Fried, D. J., Korba, J. and Das, K. (2000) "The 1999 DARPA Off-Line Intrusion Detection Evaluation", *Computer Networks*, Vol. 34, No. 4, pp 579-595

Lyu, M. R. and Lau, L. K. Y. (2000) *Firewall Security: Policies, Testing and Performance Evaluation*, 24th International Computer Software and Applications Conference, pp 116-121

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, Glamorgan, United Kingdom, pp 303-311

Sanfilippo, S. (2005) "Hping Security Tool", [online], http://www.hping.org/

Santana Torrellas, G. A. and Villa Vargas, L. A. (2003) *Modelling a flexible network security systems using multi-agents systems: security assessment considerations*, 1st international symposium on Information and communication technologies, Dublin, Ireland, pp 365 - 371

Turner, A. (2005) "Tcpreplay: Pcap editing and replay tool for *Nix", [online], http://tcpreplay.sourceforge.net/