VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Assessment of Packet Filter Technology

Christian Seifert, Ian Welch, Peter Komisarczuk
{cseifert, ian.welch, peter.komisarczuk}@mcs.vuw.ac.nz

Technical Report CS-TR-06/12
May 2006

Abstract

Packet filters are widely adopted security technologies that provide strong security defenses to a network. However, despite their strength they also pose a danger with a false sense of security. In this paper, we assess packet filter technology to provide an increased understanding of their limitations. We describe shortcomings around design, administration, and performance of packet filters and how these shortcomings decrease the effectiveness of packet filters. While operators of packet filters might be able to address these shortcomings by practicing defense in depth and breadth, we also present research opportunities to continue to improve packet filter technology.

1 Introduction

Firewalls and packet filters have been widely adopted by many companies and organizations as essential components to protect their computer networks. According to the 2005 CSI/FBI Computer Crime and Security Survey, 97 percent of respondents make use of firewall security technology [6]. The success of firewalls can be contributed to their ease of deployment and their effectiveness. According to the major firewall vendor Checkpoint, 90 percent of network attacks are commonly defeated by firewalls [2]. As such, firewalls are the success story of network security appliances.

The positive reputation of firewalls and packet filters can, however, also have some drawbacks. With such a power horse in place defending a network, a false sense of security might be adopted by the operators of the network. In addition, a common set-and-forget mentality on firewalls potentially decreases their effectiveness in the face of changing technology and newly emerging threats. False sense of security could also lead to neglecting practicing defense in breadth and depth. A strong firewall should not lead to lax security measures within the internal network, which would leave those systems unprotected in case of firewall failure.

We believe the potential problems of firewalls, in particular packet filters, are not well understood. However, deep understanding of this security technology is necessary in order protect assets effectively. We assess firewall packet filter technology in this paper, which should add to the understanding of the capabilities and shortcomings of firewalls for network administrators and security officers. In areas that pose difficulties to the operation of packet filters, we explore the work in the literature that is aimed at addressing these problems. Further, our assessment should help to spur ideas for researchers working in this area.

First, we provide some background information on firewalls and packet filters in section 2. We assess various aspects, such as design, administration and performance, of packet filter technology in section 3. We conclude in section 4.

2 Background

2.1 Firewalls

A firewall, in the traditional sense, is a passive fire protection system designed to keep fire from spreading through a building via insertion of fire-resistant walls. In the area of computer security, firewalls are also keeping something from spreading through a network. Instead of fire, they inhibit the spread of malicious activity, such as self-propagating worms entering or escaping from a network. Further, firewalls protect networks from unauthorized access, such as access to an internal fileserver, and they also defend against certain types of denial of service (DoS) attacks. Network firewalls first appeared in the late 1980s with the first implementations as part of existing router technology.

Firewalls are components that restrict access between a protected and unprotected network. They may consist of packet filters, network address translation (NAT) devices, and proxies, which are configured to enforce a network security policy on access between a protected and unprotected network [16]. Depending on its configuration, a firewall might only make use of some of the listed components as they enforce different aspects of a security policy:

- NAT device - NAT is the practice of rewriting source and destination addresses of IP packets as they traverse through a NAT device. While NAT's primary purpose is to enable multiple hosts on a private network to access the internet using a single IP address, NAT also relates to network protection. By performing NAT, identity of the

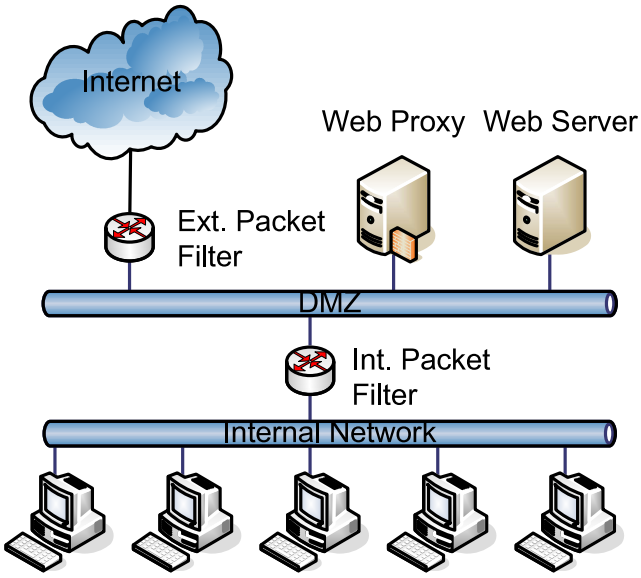

Figure 1: Firewall Architecture with DMZ

hosts of the protected network is concealed and therefore direct access to these hosts from the outside is restricted.

- proxy/ application level gateway - In order to restrict direct connections from one network to the other, a proxy acts on behalf of a client or server as an intermediary. It relays requests on behalf of the client to the server and relays answers on behalf of the server back to the client.

- packet filter - In order to restrict network traffic flowing between the networks, packet filters can be used to control the traffic based on certain packet header characteristics and state. It can reject or accept a packet flowing between the networks. With Screend, Mogul introduced the first packet filter in 1989 [14].

Figure 1 shows an example firewall architecture with the components described above. An exterior packet filter protects the entire network from unauthorized access from the Internet, but does provide access to the servers located in the demilitarized zone (DMZ). The interior packet filter provides an additional layer of protection to the internal network. No traffic is allowed to pass into the internal network from the Internet. Hosts of that network communicate to the Internet via the web proxy of the DMZ. This configuration is very effective to protect internal assets while at the same time providing services to the Internet. It is an example and many additional configurations exist to meet specific needs. Chapter 6 of Building Internet Firewalls describes various firewall configurations [16].

2.2 Packet Filters Definition

A packet filter is a firewall component that is able to control flow of network traffic between networks based on characteristics of the packets making up the network. They work with the data that can be found on OSI layer 3 and 4. Header information from the Internet layer (IP) and transport layer (TCP, UDP, ICMP, etc.) of the packet are used to determine what filtering action the packet filter is to take on the packet. The most common actions are accepting and therefore routing the packet, and rejecting it. The characteristics and action are commonly referred to as a *filtering rule* that are combined into a set of filtering rules or *policy*.

Rule Name	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
Allow Incoming DNS	UDP	*	53	192.168.0.0/16	*	accept
Allow Outgoing New DNS	UDP	192.168.0.0/16	*	*	53	accept
Drop everything else	*	*	*	*	*	reject

Table 1: Packet Filter Rule Set Example

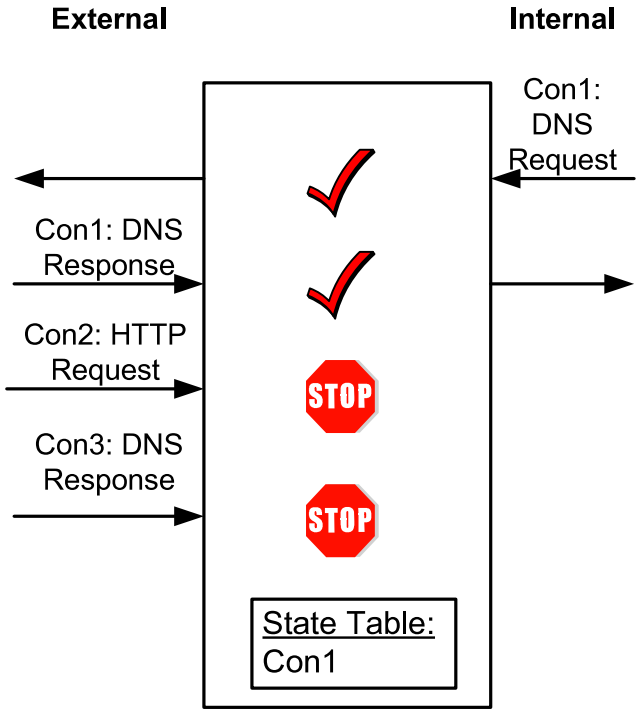

Figure 2: Packet Filter Example

A rich variety of different implementations of packet filters exists today. Open source implementations to commercial software and hardware implementations cater to every need that operators might have towards packet filters. Iptables and PF are two examples of open source software that are supplied with the Linux and OpenBSD operating system. Please refer to current search engines and firewall surveys for an up-to-date list and description of packet filters that exist today.

Table 1 and figure 2 show an example of a rule set and the corresponding packet filter behavior for some sample packets. This rule set accepts outgoing DNS requests (UDP, port 53), incoming DNS responses (UDP, port 53) and rejects all other traffic. (In this example, we assume that the packet is evaluated against each rule from top to bottom and on the first match, its action is applied. However, how a packet filter evaluates the rules is determined by the specific implementation.)

Packet filters focus on network packets, but do have the ability to control the packet based on its associated connection state. The packet filter can take action depending on whether the packet is part of an already established connection or whether it is part of a new connection. This allows for more granular rule sets that, for example, reject new connections from being established into the protected network while at the same time allowing new connections being

established from within the protected network. They fall into two categories in respect to tracking connection state: stateless and stateful.

2.2.1 Stateless Packet Filters

Stateless packet filters evaluate characteristics of the packet itself in order to determine whether the packet belongs to a new connection or an established connection. For packets of the transport layer TCP, this can be achieved by inspecting the TCP SYN and ACK flags that are changing as part of the three-way handshake that is performed to establish TCP connections. If the SYN flag is set and the ACK flag is not set, the packet belongs to a new connection, whereas other flag combinations would indicate that the packet belongs to an established connection.

Stateless packet filters do not need to maintain state in order to evaluate a packet. While this might lead to a simplification and potentially performance advantages over a stateful packet filter, it limits the packet filter in evaluating a packet against the true connection state. On the transport layer TCP, a stateless packet filter will classify a lonely packet with SYN flag not set and ACK flag set to be part of an established connection, even if its three-way handshake has not occurred to establish the connection. As a result, it would incorrectly let this packet pass through the filter. Furthermore, stateless packet filters fail on evaluating packets that are connectionless in nature, such as UDP, as the packet does not contain information about its connection state. This was first pointed out by Chapman [1].

2.2.2 Stateful Packet Filters

Stateful packet filters remedy the shortcomings of stateless packet filters by keeping track of established connections in a state table [12]. A unique key is generated and stored based on protocol, source IP address, source port, destination IP address, and destination port, which each packet is evaluated against. The existence of an entry indicates that the packet is part of an established connection whereas the lack of an entry indicates that the packet is part of a new connection. Walking through the example above would yield the correct classification of the packet. Furthermore, UDP packets can be evaluated against their connection state with a stateful packet filter.

Figure 2 demonstrates how this works in practice. The outgoing DNS request of connection one (con1) leads to the addition of con1 to the state table. As a result, the DNS response of con1 is permitted back into the internal network. However, an unsolicited DNS response targeted towards the internal network, as represented by connection three (con3), is correctly rejected.

3 Packet Filter Technology Assessment

3.1 Design

Packet filters are components that work well in protecting a network from malicious activity and unauthorized access. Their simple design make packet filter easy to understand and effective. They focus on evaluating packets against a set of rules to decide whether a packet should be accepted or rejected. However, despite these positive aspects of the design, there are several design flaws that one needs to be aware of when operating a packet filter to protect a network. Some of these shortcomings can be mitigated, but all have the potential to reduce the packet filter's ability to enforce a specified security policy.

3.1.1 Tunneling

Packet filters inspect network traffic on the packet level. As such, their view of the underlying content is limited to a packet at a time. With such a intermittent view of the network traffic, packet filters do not have the ability to make filtering decisions based on the underlying content. As a result, a packet filter that is intended to block content might easily be circumvented by routing the traffic with different characteristics that are not blocked. For example, a packet filter might intend to allow only HTTP traffic (TCP, port 80), but as other applications use the same packet header characteristics, the packet filter unintentionally will allow for non HTTP traffic to pass. Applications like Skype and MS Media Player make use of that specific loophole.

Application-level gateways are often cited as the technology that complements packet filters to mitigate this design flaw. They relay requests on behalf of the client to the server and relay answers on behalf of the server back to the client. In order to do so, they need to be aware of the underlying protocol and therefore be able to make additional filtering decision that the packet filter is not able to make. For example, a web application-level gateway could restrict access to specific content, such as external web-based mail servers, through blacklisting or inspection of page content.

Application-level gateways are not the best solution in mitigating this design flaw as they complicate deployment of the firewall and add administrative overhead. Recently more intelligent packet filters started appearing that reassemble traffic streams, are protocol aware, and as such are able to make filtering decisions based on the underlying content. Examples are Snort-inline and Netfilter with Data Replacement Patch. These packet filters are able to understand content of the more widely used protocols, such as HTTP and ftp, but fail to restrict content of other protocols or encrypted traffic. As such, the effects of the design flaw have been decreased, but have not disappeared entirely.

3.1.2 Information Leak Due to Fragmentation

Fragmentation reveals a design flaw of packet filters on the other side of the spectrum in making filtering decisions. In contrast to their failure to deal with aggregated packets in the form of content, packet filters also have problems dealing with fragments of packets. Fragmentation is a mechanism of the network layer to allow traversal of packets on the link layer with specific packet size limitations by dividing a large packet into smaller fragments, which are reassembled at the destination host. Only the first fragmented packet will contain the full header information of the higher-level protocol (TCP or UDP), whereas the subsequent fragmented packets will just contain information on how to reassemble the fragmented packets into one packet.

Chapman noted in 1992 that fragmentation will lead to complications on packet filters as they will not be able to filter non-first fragments due to missing header information of the higher level protocol [1]. Packet filters commonly accepted any non-first fragment, assuming those fragments could not be reassembled without the first packet fragment in case the first packet was rejected by the filter. However, allowing non-first fragments to freely pass through a packet filter opens the door for information leaks (the actual data in the fragment, ICMP responses to incomplete fragments) as well as DoS attacks (e.g. crashes of IP stacks due to overlapping fragments) [16].

Packet filters that perform fragment reassembly solve this issue. They reassemble packets (or fragment packets) locally before filtering and routing the packets. The cost is increased load on the packet filter, which needs to be weighted against the security requirements of the network the packet filter is protecting. Scrub, Iptables and many commercial products allow for fragment reassembly today.

3.1.3 Lack of Control due to Ephemeral Port Assignment

Ephemeral port assignment poses another problem on packet filters effectively enforcing security policies. Ephemeral ports (in addition to the client's IP address) are ports on the client, which are opened to specify where the client request came from and where the server should send its response as TCP/IP traffic is bidirectional. In order for TCP/IP connections to function over a packet filter, the packet filter has to accept traffic to the client's ephemeral port for that connection and is essentially temporarily opening a port within its filtering rules. Assume a user is requesting a web page of server A. The user's web client generates packets to destination port 80 of server A, but also opens up an ephemeral port, for example 61000, to receive the response of server A. Server A generates a response and sends those via packets with destination IP of the client and destination port 61000. The packet filter temporarily opens port 61000 to allow for the reception of the web server response.

On stateful packet filters that only accept outgoing connections from the protected network, this should not pose a problem as the ephemeral port is only accepted for the duration of the connection as requested by the client. However, since the assignment of the ephemeral port is not random as pointed out by Chapman [1], a problem might occur if the packet filter is tasked with rejecting all incoming connections into the protected network. It is possible to circumvent that security policy if two protected hosts are able to predict the next ephemeral port of each other and start sending UDP packets to those ports at the same time. Since entries in the session table are created, the packet filter assumes that the incoming packets are responses to an established connection although they are not. It would incorrectly accept these packets.

As outgoing traffic is permitted, implicitly certain incoming traffic into the network is also permitted, which might not be acceptable according to one's security policy. True random assignment of ephemeral ports would be a solution to this problem. However, it seems like this is currently not implemented on some of the newer operating systems, such as Red Hat Fedora Core 4 and Windows XP SP2.

3.1.4 Denial of Service on Connection State Tracking

Connection state tracking is a desirable functionality of packet filters to allow for more fine-grained rule definition. However, with increased functionality also come some problems that an operator of a firewall needs to be aware of.

Gill, for example, describes several attacks that would lead to a DoS on commercial stateful packet filters that store connection state in a session table [5]. By sending SYN flood, UDP flood, or Crikey flood traffic, the packet filter would add an entry to the state session table for each packet that it inspects. The state session table would quickly fill to its maximum capacity. With the exhausted resources, a packet filter would not allow further connections to be established. Surprisingly, these attacks require only a small fraction of traffic that the packet filter is capable of handling to make the attack successful (e.g. the Netscreen 100ES, a Gigabit rated packet filter that is able to handle 300,000 entries in its state table, would be overwhelmed by only 2.6MBit of traffic).

Gill also describes measures that can be taken in order to counter these sorts of attacks through several performance optimizations. The author lists checksum validation, init flow timer optimization, session table expansion, session table limiting, rate limiting, aggressive aging, TCP proxy, TCP SYN cookies, TCP SYN required and TCP fastpath as possible solutions. While these measures do not prevent the DoS, they put more burden on the attacker (e.g. more bandwidth is needed) and reduce the likelihood of a DoS occurring. Gill falls short in determining what needs to take place in order to successfully deflect DoS attacks. As result, operation of a packet filter makes a network vulnerable to DoS of external

network connectivity.

3.2 Administration

Administration is another aspect of packet filters that we would like to assess. Packet filters are simple and therefore easy to understand. They are easily deployable to a network as they can be placed as a black-box at distinct, well-defined choke points within a network. Deployment is a mere plug-and-play and networks do not necessarily need to reconfigured as a result of inserting a packet filter. However, there are several aspects that are causing problems in the realm of administration, which we would like to discuss in more detail. Those aspects are rule configuration and monitoring.

3.2.1 Rule Configuration

Chapman was the first to describe that filter rules are difficult to configure even though a packet filter rule makes a simple binary decision on whether a packet should be accepted or rejected based on a few characteristics. With an increased number of rules, complexity increases tremendously as filter rules start to overlap, and the order of rules becomes important for making a filtering decision. Functional testing is essential for every operator before bringing a packet filter online.

Before we review aspects of functional testing, we review a prerequisite to functional testing: the specification. In the world of packet filters, the specification is the security policy that the packet filter enforces. Jalili et al. presents the Security Policy Specification Language [11]. This work allows security experts to define the security policy the firewall needs to enforce in terms of roles within the network. The language supports analysis and verification of the security policy allowing for detection of inconsistencies, detection of lack of coverage, and querying against the security policy, thus allowing for functional testing based on the formal specification to occur.

The following works are ignoring such testing formalism and are rather concerned with the practical difficulties of functionally testing firewalls. Those works assume tests are derived from the actual packet filter configuration file. While this approach tests against the behavior of the packet filter, it does not allow for testing against the security policy the packet filter is tasked with enforcing. Such tests miss errors that result out of the incorrect translation of the security policy to the packet filter configuration.

Hoffman et al. introduces a framework for test automation, which addresses such practical concerns of functional test execution [10]. They describe the test setup that isolates the packet filter under test for focused testing. The authors describe and demonstrate a testing API for execution of complex test scenarios, such as SYN flood traffic and bad TCP flags. In addition, their work provides us with tools for regression testing of firewalls.

El-Atawy et al. is concerned with the difficulty of test case creation [4]. The authors represent the rule set as a binary decision diagram (BDD), which allows them to derive test cases that traverse down each path of the BDD. Policy segmentation, as this process is called, generates test cases that accommodate comprehensive coverage for white box testing.

3.2.2 Monitoring

Monitoring of packet filters, an essential administrative task, is difficult. Log files that capture rejected packets seem to be the primary means of monitoring packet filters even today. With the different IP stack implementations and vast amount of data that passes through a packet filter, log files will make note of too many rejected packets for an administrator to be able to absorb and analyze. Some products, such as Snortalog and Insideout, filter logs and provide

statistics that assist security experts in dealing with the data. Standards around log files also do not exist, leading to custom transformations that need to take place before utilizing firewall logs in such tools and, as a result, hinder their development and acceptance.

An attempt to improve the situation is made by Lee at al. [13]. They represent data from firewalls in rich graphical views that contain different levels of detail, different time scales, as well as reaction of the firewall to the traffic. On the graphs that are relevant to packet filters, displays of real time traffic, statistical views, and connection views are described. Examples on how attacks, like port scans, worm attacks and DoS attacks are displayed and it is shown how these attacks can be visually detected. While the data visualization still requires security experts to deal with large quantities of data that a packet filter creates, it is a step in the right direction to tackle the problem around monitoring.

3.3 Performance

Performance is a concern for packet filters that have to filter through all traffic that flows to and from the network. For low volume choke points, performance is not a concern. However, with traffic volume and speed increasing, packet filters do have a difficult time handling this additional workload gracefully. In case packet filters become overburdened with traffic, communication from and to the network under the packet filter's control will be impaired. In this section, we review the work on determining the performance characteristics of packet filters.

The Internet Society published two RFCs that laid the groundwork on firewall and packet filter benchmarking and performance testing [15, 8]. RFC2647 defines terminology to be used when dealing with benchmarking and performance testing of firewalls, which has been derived and extended from the corresponding RFCs on routers and switches. RFC3511 describes the test setup and the actual tests that need to be executed for benchmarking firewall performance. The RFCs stay general in the description allowing the framework to be used to evaluate specific performance characteristics of a firewall, but also to determine scalability characteristics through load tests.

Hoffman et al. is an example on investigating scalability of the packet filter Iptables through load tests [9]. In several iterations, the authors execute performance tests on Ipfilter with an increasing rule set and growing traffic load. They measured IP throughput and latency while varying rule set size from 0 to 500 rules and traffic from 10Mbps to 1000Mbps, concluding that Iptables under their configuration were scalable up to 10Mbps/500 rules and 100Mbps/100 rules. Degradation resulted from the more demanding configurations showing that even for current implementations and configurations of packet filters, performance and scalability are concerns.

In response to performance and scalability issues, researchers have reviewed and improved upon the filter algorithms of packet filters. Different representations of filtering rules for performance optimizations are explored by Hazelhurst et al. and Christansen et al. [7, 3]. Their work led to performance optimization around evaluation of packets against rule sets. Hazelhurst et al. presents rule sets as BDDs, in which each node represents a test condition of a rule and each terminal a filtering action. As test conditions of a rule are usually numerical, the numerical value is converted into a bit representation in which each bit represents a node in the BDD. This will lead to a large BDD, but with the assistance of Boolean logic that can be applied to these representations, the BDD size is reduced considerably. Using a sample rule set of 430 rules, the maximum operations for classification of a packet against the resulting reduced, ordered BDD would be 72, as opposed to 2500 against the rule set table. No actual performance tests were executed to quantify the performance gain with the BDD representation.

Christiansen et al. further optimizes the representation of rule sets and propose to represent the rule set in an Interval Decision Diagram (IDD) that is able to handle numerical values, but is also subject to the advantages of Boolean logic to reduce the size of the diagram. As a result, the overall size of the IDDs is reduced compared to BDDs. Since Boolean evaluation and integer evaluation on generic CPUs is comparable, IDDs are more performant than BDDs. The authors provide empirical data to support the performance claims of the IDD-based packet filter by comparing it to traditional implementations of packet filters, but fail to compare it to a BDD-based packet filter. Hazelhurst et al. and Christiansen et al.'s work illustrate that performance characteristics are heavily dependent on the internal representation of rules of a packet filter implementation.

4 Conclusion

In this paper, we have provided an assessment on packet filter technology in respect to design, administration, and performance. Our assessment has revealed that a number of problems do exist despite the many positive aspects of packet filters. Operators of packet filters need to be aware of these shortcomings in order to implement a security policy effectively.

In the area of design, we have identified several problems around functional aspects of packet filters. Tunneling, information leaks due to fragmentation, lack of control due to ephemeral port assignments, and DoS susceptibility are all aspects of packet filter that reduce the functional effectiveness in enforcing a defined security policy. Often mitigation strategies exist, but they are made with the cost of decreased performance. Around these areas lie great opportunities for researchers on expanding data analysis capabilities and handling of state information in order to overcome those issues.

In the administrative realm, we conclude that packet filters are very easy to deploy, which is likely to have contributed to their wide success. However, despite the ease of deployment, we have identified problems around the configurability of packet filter rules and monitoring of firewall logs. While those two aspects are not essential in operation of a packet filter, it is highly recommended to assess correctness of packet filter configuration through functional testing and continued monitoring of firewall logs to ensure the firewall is performing as expected. We appeal to the research community to address the complexity of firewall configuration difficulty as well as providing means for easier monitoring beyond on what has already been presented. Standards, models, and taxonomies on packet filters are sparse and need to be created in order to address these problems in a fundamental approach.

Finally, we have assessed performance aspects of packet filters. While performance testing and performance optimizations have taken place, it is expected that performance problems will continue to exist as workload, traffic speed and traffic volume increases. Continued evaluation and optimizations need to take place. The research community should study truly scalable configurations that perform independent of volume, increased workload, and traffic speeds and provide test data and test cases that would allow for an objective evaluation of packet filters.

In this paper, we have assessed several aspects of packet filters that hinder their functional effectiveness, administration, and performance, which are important in an operative environment. However, by defining these shortcomings, we have raised awareness and provided operators with the knowledge that allows them to mitigate some of these shortcomings in other ways, such as practicing security in breadth and depth. At the same time, we have highlighted several areas that are worthy of further research.

References

[1] Chapman, B. D. Network (in)security through ip packet filtering. In *Third USENIX UNIX Security Symposium* (Baltimore, 1992), USENIX, pp. 63–76.

[2] CheckPoint. Check point application intelligence, 2006.

[3] Christiansen, M., and Fleury, E. An mtidd based firewall. *Telecommunication Systems 27*, 2 - 4 (2004), 297–319.

[4] El-Atawy, A., Khaled, I., Hazem, H., and Al-Shaer, E. Policy segmentation for intelligent firewall testing. In *13th IEEE International Conference on Network Protocols* (Boston, 2005), IEEE Computer Society, pp. 67–72.

[5] Gill, S. Maximizing firewall availibility: Techniques on improving resilience to session table dos attacks, 2002.

[6] Gordon, L. A., Leob, M. P., Lucyshyn, W., and Richardson, R. Csi/fbi computer crime and security survey, 2005.

[7] Hazelhurst, S., Fatti, A., and Henwood, A. Binary decision diagram representation of firewall and router access lists. In *South African Institute of Computer Scientists and Information Technologists* (Gordon's Bay, 1998).

[8] Hickman, B., Newman, D., Tadjudin, S., and Martin, T. Rfc3511 benchmarking methodology for firewall performance, 2003.

[9] Hoffman, D., Prabhakar, D., and Strooper, P. Testing iptables. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research* (Toronto, 2003), IBM Press, pp. 80–91.

[10] Hoffman, D., and Yoo, K. Blowtorch: a framework for firewall test automation. In *20th IEEE/ACM International Conference on Automated Software Engineering* (Long Beach, 2005), ACM Press, pp. 96–103.

[11] Jalili, R., and Rezvani, M. Specification and verification of security policies in firewalls. In *1st EurAsian Conference on Advances in Information and Communication Technology* (Shiraz, 2002), Springer, pp. 154–163.

[12] Julkunen, H., and Chow, C. E. Enhance network security with dynamic packet filter. In *7th International Conference on Computer Communications and Networks* (Lafayette, 1998), IEEE, pp. 268–275.

[13] Lee, C. P., Trost, J., Gibbs, N., Beyah, R., and Copeland, J. A. Visual firewall: real-time network security monitor. In *IEEE Workshop on Visualization for Computer Security* (Minneapolis, 2005), IEEE Computer Society, pp. 129–136.

[14] Mogul, J. C. Simple and flexible datagram access controls for unix-based gateways. In *USENIX* (Baltimore, 1989), Digital Equipment Corporation, pp. 203–221.

[15] Newman, D. Rfc2647 - benchmarking terminology for firewall performance, 1999.

[16] Zwicky, E. D., Cooper, S., and Chapman, B. D. *Building Internet Firewalls*, 2nd ed. O'Reilly, Sebastopol, 2000.