What Do Firewalls Protect?
An Empirical Study of Firewalls, Vulnerabilities, and Attacks

Sean Peisert and Matt Bishop
Department of Computer Science
University of California, Davis
{peisert,bishop}@cs.ucdavis.edu

Keith Marzullo
Dept. of Computer Science & Engineering
University of California, San Diego
marzullo@cs.ucsd.edu

March 30, 2010

Abstract

Firewalls are a cornerstone of how sites implement "defense in depth." Many security policies assume that outside attackers must first penetrate a firewall configured to block their access. This paper examines what firewalls protect against, and whether those protections are sufficient to warrant placing the current level of trust in firewalls.

1 Introduction

Individuals and corporations build multiple layers of security mechanisms to protect their computers. They use firewalls, intrusion detection systems, virus scanners, and other protective software, and these mechanisms provide some level of assurance that the security policies for the site are properly implemented. An active system administration staff, knowledgeable about security, adds to this assurance. But is this assurance misplaced? How effective are the security mechanisms upon which the individuals and the corporations rely?

Firewalls are the cornerstone of most computer and network security defenses. They are widely deployed. But they are hard to configure properly, and those who configure them may not have a good understanding of *current* threats and attacks. For example, an administrator may open a hole in a firewall to accomplish some task, but in doing so enable attackers to enter through the firewall. Or the firewall may have a vulnerability that enables attackers to defeat its prohibitions.

Additionally, by design, firewalls constrain use of systems and networks by blocking services. The number of blocked services can clearly be quantified, as can the time spent by administrators to attempt to circumvent the firewall by tunneling the service through the firewall, as well as the performance degradation by using tunneling. But, without knowing how effective these protection mechanisms are, it is very hard to define tradeoffs between cost, usability, and degree of protection. In some cases, firewalls may not provide protection for a particular resource no matter how the firewall is configured. The paradigm is sometimes simply inappropriate. For example, attacks may originate internally, either by insiders or by bots which subvert the purpose of firewalls. Hence, it is hard to quantify the amount of protection firewalls bring to a system.

Best practice suggests using both host-based and network-based firewalls, and keep them as restrictive as possible. This follows the principle of fail-safe defaults, which says to deny access by default [SS75]. Such a posture assumes that the rulesets controlling the firewalls are complete, that the firewalls have no vulnerabilities that external attackers can exploit, that the insider threat [Bis05] does not exist, and that the firewall runs only safe services.

In this paper we examine the these assumptions empirically and determine for what and when firewalls are appropriate and inappropriate. To do this, we look at vulnerability databases, lists of most-attacked network ports, and sets of firewall rules used in practice.

We seek to answer the questions:

1. Is the rule set complete? If not, which rules are missing?
2. Given the known vulnerabilities and most-attacked network ports, what is a minimal set of rules that will block the attacks?
3. Given the above information and a set of configuration rules and parameters, what services are unsafe for a firewall to run?
4. Can we use multiple sets of known vulnerabilities and lists of most-attacked ports to generate an ideal ruleset?

The rest of this paper continues as follows: Section 2 discusses related work, Section 3 discusses our theoretical method of firewall analysis, Section 4 describes our experimental method and results, Section 5 presents a discussion and analysis of the results, Section 6 presents our ideas on designing and using firewalls more effectively, and Section 7 presents our conclusions and ideas for future work.

2 Related Work

Firewall rules provide protection for sites. Recently, a number of studies have demonstrated how errors or conflicts in rule sets can diminish protections [ASH04, CCBGA05, UC07, YMS+06]. Others have compared rulesets to "best practices" [BMNW04, EZ01, MWZ06, Woo01, Woo04]. These studies identify ways in which firewall configurations can be improved.

However, "best practices" are rapidly moving targets. For example, a recent community effort to determine a set of recommended firewall rules for the host-based firewall `ipfw` on Mac OS X/FreeBSD systems produced a large and complicated set of rules [MwL+07]. Complicating these rules is the way Mac OS X handles the ipfw program; it is included in the Mac OS X version 10.5 (Leopard) and 10.6 (Snow Leopard) distributions, but is not the default firewall. Apple's own Application Layer Firewall [App07], the default firewall, has a simpler user interface than the BSD `ipfw` firewall, but offers coarser filtering control. Given all this, how does one determine what the best practices for such a system are, let alone whether the current configuration meets them?

We approach the problem from a different perspective. We want to determine empirically how well firewalls thwart current attacks, and how well they protect against the exploitation of known vulnerabilities. So we do not compare sets of firewall rules with one another for consistency. Instead, we compare the rules against attacks and vulnerabilities.

3 Firewalls

The widespread deployment of firewalls have often given the impression that vulnerabilities in network-facing programs are difficult to determine in order to exploit remotely, and therefore the firewall ameliorates their severity. On the other hand, the widespread use of mobile computing means systems behind a firewall may not remain behind that same firewall. Instead, they may move to another network where there is no firewall or where the firewall rules are different. This means that *there is simply no longer a perimeter for computer systems and networks.* There are only "guards" over information flow [Smi94]. For example, consider a mobile host within a company, where the network is administered by a company-wide IT department. The owner is a vendor

from another company who takes that host into a conference room and connects to the network in a specific company subdivision. That subdivision has more stringent security policies than the company-wide policies (perhaps because it is working on a highly sensitive project). If the host is compromised on the company-wide network, it functions as a classic insider on the subdivision's network: something that is trusted, yet betrays that trust [BEP+08, BEP+09, BEF+10]. This notion of a compromised, mobile machine, moving freely behind firewalls, is a version of the insider problem. As such, is more difficult to defend against than external threats because an insider attack is hard to distinguish from legitimate use, and the security process should not prevent users from doing their jobs.

Security policies, including the use of firewalls can be complicated to configure. In an ideal situation, a formal analysis such as the one we describe now would be done. However, there is little to no coordination of layers of defense in practice, including in the rulesets that we analyze in this study. The set of firewalls that protect an organization can be represented as a graph G. In some organizations, particularly large ones, this graph starts at the network entrance points, has a second level to protect individual sub-domains, and a termination at host-based firewalls. We denote the border and internal firewalls as *network-based firewalls* and represent them as internal nodes to distinguish them from the leaf nodes representing *host-based firewalls*. One way to measure the efficacy the defenses represented by graph G is to take a set of attacks and see at what points, and at how many points, in G an attack a would be blocked. The number of points is critical because it identifies the number of points at which the attack would be blocked. As most firewalls are configured to "fail open" in order to remain usable, if only a single point blocks an attack, that one point must not fail. Further, given the possibility of distributed firewalls, there could be multiple paths that the attack could exploit, and each of these must be checked.

Define a node e to be an entry point at a gateway, which may also represent a firewall. Let m be a machine, a be an attack, and let a_m contain sets of firewalls that lie on the path between the exterior network(s) and m (inclusive). Initially, $a_m = \emptyset$. Then the following algorithm determines the set of points at which the attack a on machine m could be blocked:

For each machine m:
  For each entry node e:
    For each path (m, e):
      For each attack a:
        if a can traverse (m, e) then:
          Let $n_1, ..., n_i$ where $m = n_1$ and $e = n_i$ (if m and e are firewalls) be the set of firewalls along (m, e) (inclusive). Then $a_m = a_m \cup \{\{n_1, ..., n_i\}\}$

At the end of this algorithm, each set a_m contains sets of firewalls that lie on the path between the exterior network(s) and m. If a_m is empty, then there is no path on which the attack will be successful (without a firewall failing). If a_m is not empty, but there exists a set within a_m that is empty, then an a can successfully traverse (m, e) without being blocked by any firewall. If a_m is not empty and no set within a_m is empty then to determine the minimal set of points at which the attack a on machine m can be blocked, choose the set of nodes with minimum cardinality and having at least one node from each set in a_m.

Note that for mobile hosts, this calculation potentially changes each time the host moves. For our purposes, we do not perform vulnerability analyses of the hosts themselves, since our purpose is to study the usefulness of firewalls, not to quantify the overall security of the network. In our study, because of the difficulty of obtaining large sets of firewall rules with multiple layers, and the need to know when they hold with respect to the period during which a vulnerability is active, our

graph G has a depth of one. Also, we ignore the question of whether vulnerabilities actually exist on the hosts behind the firewalls. Even if the specific vulnerabilities do not exist on the hosts, we assume that the ports that *unknown* or *zero-day* vulnerabilities will appear in roughly the same distribution as the previous, *known* vulnerabilities.

4 Method and Results

As we discuss in this section, our findings on attacks and vulnerabilities show that firewall placement and configuration can be challenging, and applying the algorithm that we described above is a necessary component of a methodical approach for defending a set of assets from a set of threats.

Sites that do not protect against attacks and vulnerabilities are (knowingly or unknowingly) more at risk than sites that are protected, even if the firewall rules protect against other vulnerabilities specific to the site. This suggests our methodology: we analyze the firewall rules using data about known vulnerabilities and known attacks. Specifically, we compare the most vulnerable and most attacked port numbers with port numbers protected by firewall rules.

Table 1: An analysis of which ports and protocols are most commonly blocked in the 794 firewall rulesets.

Rank	Protocol	Port number	# of FW rulesets Blocking	% of FW rulesets Blocking
1	TCP	443	763	96.1%
2	TCP	80	734	92.4%
3	TCP	25	710	89.4%
4	TCP	23	700	88.2%
5	TCP	389	697	87.8%
6	UDP	53	695	87.5%
7	TCP	22	694	87.4%
8	TCP	110	694	87.4%
9	TCP	1494	693	87.3%
10	TCP	993	691	87.0%
11	TCP	995	690	86.9%
12	TCP	119	689	86.8%
13	TCP	563	688	86.6%
14	TCP	1503	688	86.6%
15	TCP	636	687	86.5%
16	TCP	143	686	86.4%
17	TCP	513	685	86.3%
18	TCP	514	685	86.3%
19	TCP	1352	677	85.3%
20	TCP	5190	663	83.5%

The 794 (anonymized) firewall rulesets that we used in this study, shown in Figure 1, comes are real rulesets currently used by small to medium sized businesses (including non-profits, law firms, car dealerships, medical offices, and more), and also local governments and schools. Networks have between 10 and 400 users, with the median being apprixmately 30 users. In these instances, internal technical resources are generally either small or entirely outsourced. The information about

most vulnerable port numbers comes from the vulnerability database of the popular vulnerability scanner, Nessus.[1] We obtained the information about the most attacked port numbers from data that the SANS Internet Storm Center collected over a 21-month period.

We considered several other sources of data, such as Foundstone's database,[2] but the data was proprietary. Nessus contains a database of vulnerabilities and tests to determine whether the target has that vulnerability. The tests may try to exploit the vulnerability (and hence Nessus is often considered a "penetration testing" tool, even though this is not correct). New vulnerabilities receive a new test in Nessus, but new exploits do not necessarily receive a new test, unless the exploit tests a vulnerability not already in the database. Fortunately, Nessus uses an open, and relatively easy-to-parse scripting and plugin language to create the tests for target systems. In addition to considering the network ports that run software containing the most prominent vulnerabilities, we also included the ports most commonly *attacked*, because firewall rulesets should handle these as well.

Note that it is not sufficient to run the Nessus scanner and collect information using a firewall logger or packet sniffer to obtain the vulnerabilities. Most vulnerability scanners fingerprint a machine first to determine the operating system in use. Then they run the scan to collect information. So, the picture presented would be more limited because the results would include only the ports scanned for the particular version of the operating system being tested.

We circumvent this problem by analyzing the scripts that the scanners use to probe for known vulnerabilities. Unfortunately, analyzing thousands of scripts manually is impractical. So we use two methods: first, we read the rules files directly and parse the service/port/protocol information; second, we read the trace output of the `nasl` tool (a component of Nessus) on all of Nessus's vulnerability scripts. Because both methods rely on ad hoc parsers and use a collection of regular expressions, there is a margin of error. Results of the two methods are similar, but we take the union of port numbers. provided for a particular vulnerability by each method. We do this because the parser is much more likely to miss a port indicated in a vulnerability—such as when the port is assigned dynamically in the Nessus test script rather than statically—than it is to add one by mistake. In a manual sampling of the data, we found no evidence of test scripts being incorrectly parsed and having additional ports added as a result.

In principle, it would also be ideal to conduct the vulnerability part study using the largest and most complete set of vulnerabilities as possible, such as those collected by US-CERT,[3] the SANS Internet Storm Center,[4] or the Open Source Vulnerability Database.[5] However, those databases do not directly refer to network ports, and so the data in them is not directly comparable to firewall rulesets. Network vulnerability scanners do this translation with some loss of the size of the database as well as assumptions (e.g., that a particular service always runs on a particular port). This appears to be a reasonable approximation. Indeed, using this data identifies discrepancies between the apparent intent of the test firewall rulesets and the vulnerability and attack data, and leading the firewall administrators to correct the test rulesets by adding several missing firewall rules.

According to the Nessus database, approximately 5800 out of 21,725 total vulnerabilities are remotely exploitable.[6] In Table 2, we show the network ports that have the most vulnerabilities associated with them, and the most commonly used program on that port. We also show the status

[1] `http://www.nessus.org/`
[2] `http://www.foundstone.com/`
[3] `http://www.us-cert.gov/`
[4] `http://isc.sans.org/`
[5] `http://www.osvdb.org/`
[6] `http://nessus.org/plugins/index.php?view=all`

Table 2: Number of known vulnerabilities exploitable by accessing a given network port, in descending order, as well as the percentage that were blocked in our survey of 794 firewall rulesets. The high and low numbers for vulnerabilities indicated in our two parsing methods are also shown. The Nessus data is from the database current on April 7, 2008.

Rank	% Blocked in Survey	Protocol	Port	Common service	# of vulns. (high/low)
1	92.4%	TCP	80	HTTP	2359/2300
2	2.2%	TCP	445	Windows SMB	1792/896
3	17.5%	TCP	139	Windows SMB	896/755
4	3.7%	TCP	21	FTP	176/171
5	89.4%	TCP	25	SMTP	119/118
6	20%	TCP	161	SNMP	90/81
7	9.4%	TCP	8080	HTTP	86/29
8	88.2%	TCP	23	Telnet	65/63
9	86.4%	TCP	143	IMAP	71/63
10	87.4%	TCP	22	SSH	55/48
11	87.4%	TCP	110	POP3	42/32
12	1.3%	ICMP	(ICMP)	Ping	36/21
13	2.1%	TCP	8000	HTTP	29/29
14	96.1%	TCP	443	HTTPS	25/17
15	1.1%	TCP	3306	MySQL	23/17
16	0.0%	TCP	111	Sun RPC	21/3
17	0.1%	TCP	69	TFTP	20/8
18	0.5%	TCP	3128	Squid Proxy	19/13
19	0.2%	TCP	79	Finger	16/16
20	87.5%	UDP	53	DNS	15/8

(open/closed) of the two firewalls that we tested as our example for each port.

The top 10 or so vulnerabilities clearly stand out from the rest, and indeed, the vulnerabilities associated with ports 80, 445, and 139 (HTTP and Microsoft Windows SMB, respectively) dwarf the remaining vulnerabilities by an order of magnitude.

Table 3: Top 10 attacked ports according to the SANS Internet Storm Center, using data from October 27, 2006 to July 25, 2008.

Rank	% Blocked in Survey	Port	Common service	# of targets
1	0.1%	1434	Microsoft SQL	283,076,767
2	0.5%	135	Windows RPC	282,561,882
3	0.5%	1433	Microsoft SQL	263,007,700
4	1.1%	137	Microsoft NetBEUI	254,521,251
5	2.9%	445	Windows SMB	253,503,686
6	92.4%	80	HTTP	184,847,744
7	2.1%	139	Windows SMB	158,451,995
8	0.0%	9898	Sasser worm backdoor	110,535,147
9	87.4%	22	Secure Shell (ssh)	108,875,526
10	0.0%	5554	Sasser worm FTP server	102,437,806

Table 3 compares the ports in the firewall rulesets to the most *attacked* port numbers as indicated by the SANS Internet Storm Center. This list coincides somewhat (but not entirely) with the list of vulnerabilities. Two key distinctions are ports 9898 and 5554, which are commonly attacked (due to worm activity) but do not have a large amount of vulnerabilities. Conversely, port 80 (HTTP) are ports 139 and 445 (SMB) are both commonly attacked and vulnerable. Note that both the border and internal firewalls have the HTTP, SMB, and SSH ports open. Indeed, these three protocols are among the most commonly used services by legitimate users.

5 Discussion

The number of vulnerabilities and attacks for port 80 is not surprising. These vulnerabilities do not indicate that the web server programs Apache and Microsoft IIS themselves are extremely vulnerable (indeed, they are not especially vulnerable); looking more closely at the vulnerabilities in the Nessus database, most occur because of the CGI scripts and other third-party applications that run in conjunction with a web server. Additionally, given the complexity of SMB (not only the protocol but also the large variety of services that run over the protocol, such as NetBIOS and DCE/RPC), the number of vulnerabilities and attacks for ports 139 and 445 are also unsurprising. Several of the ports have been used by services that have been the targets of large-scale worm attacks, including port 1434 (targeted by the Slammer worm [MPS+03]) and port 9898 (targeted by the Sasser worm [SEVS04]), two substantial and widespread attacks during the time period of our analysis. These results are also in line with the ports most commonly open and listening for connections on hosts themselves. As recently reported in a study of millions of Internet nodes [Fyo08], the most commonly open ports are 80 (HTTP), 23 (Telnet), 22 (SSH), and 443 (HTTPS), in that order.

One interesting outcome of our study is that while there is some overlap between top attacks and vulnerabilities, there are also key distinctions where there there are commonly attacked ports that are ordinarily not especially vulnerable. Those include ports 9898 and 5554, due to worm attacks.

The conclusion is that worms sometimes attack obscure services, and so blocking commonly used ports should not necessarily take precedence over blocking obscure ports, or even ports used by security software. For example, the "Witty" worm—"the first widespread Internet worm to attack a security product" [SM04]—attacked the BlackICE security suite's ICQ parser, and carried a destructive payload, resulting not just in a denial-of-service, but also frequently in permanently-crashed machines.

Unfortunately, there appear to be few ways in which to improve the effectiveness of current firewalls can improve effectiveness significantly because institutions use these ports to provide services that they cannot shut off. Therefore, closing the ports and blocking those services is not acceptable.

There are two less "binary," and also less effective solutions that could be implemented. One such solution is to use multiple layers of firewalls with the border router configured as a limited filter and interior firewalls to provide more specific protection against particular services. A second solution is distributed firewalls [IKBS00], which can be helpful at defeating attacks from the "inside" (though if all the firewalls have the same rules, the distributed firewalls will be less useful, since all firewalls enable the same services to run, and therefore allow the same attacks through). This might also be particularly helpful in protecting legacy systems running non-secure protocols that cannot be patched [KP07]. Both the multi-layered and distributed firewall approaches require careful identification of the services running on each device and coordination among firewalls. Finally, any of these solutions may simply provide a false sense of security, since the most vulnerable services are still likely to be left open on firewalls. If users or administrators do not remember this and assume that the firewall will protect them, then they will be more vulnerable if some element of security fails.

Our results suggest that while network firewalls are suitable for protecting against *unknown* network processes running on unpredictable ports, they are generally not effective at protecting against *known* or unknown vulnerabilities running on common ports. As a result, administrators might consider leaving protection against known vulnerabilities to host-based firewalls.

An interesting embellishment of this idea arises from the observation that a firewall is an additional layer of security. But when and how are they appropriate and when are they hurtful, because they provide a false sense of security [Sin03, Sin05]? Other layers that have partially overlapping function and benefits are software tools such as anti-virus programs, wrappers, network and host intrusion detection systems, and so forth. These extra layers also can have security vulnerabilities; in fact, a number of vulnerabilities have been discovered in firewalls [Sec00, Sec01, Sec02a, Sec02b, Sec04a, Sec04b, Sec07a, Sec07b] as well as in other types of security-related software, including anti-virus program sand intrusion detection systems. Our conclusion is that firewalls are useful when applied appropriately, because they can they add another layer of security. If the inside tools have vulnerabilities, the firewall may block an exploit that otherwise would succeed, and if a firewall has a vulnerability, another tool may block it. But coordinating these tools, their redundancy, protection domains, and failure modes is very hard and is rarely done.

6 Designing and Using Firewalls More Effectively

This paper used empirical data to show that firewalls do not block many network-based attacks, and suggested ways in which existing firewalls could be used more effectively. How can we use our results to design a better firewall? Addressing two key issues would ameliorate the problems discussed earlier. The first is that many significant vulnerabilities lie at the application layer, and

so application-level firewalls would be more effective for them rather transport- or network-layer firewalls. The second issue is encryption, because many attacks that involve a local exploit occur over an encrypted remote connection.

A firewall that does application-level analysis could address attacks that involve not just communication with a particular port (e.g., port 80, running Apache or Microsoft IIS), but the contents of the packets that are being sent to the program running on that port (e.g., CGI scripts). This technology already exists. Many network-based intrusion prevention systems can reassemble and analyze packets all the way from the network layer (e.g., IP) to the application layer (e.g., HTTP), and can also address client-side attacks (e.g., cross-site scripting attacks) against systems behind the IPS, as well as server-side attacks [KKVJ06]. "Application-layer firewalls" also reassemble packets, and are closely related to intrusion prevention systems, which generally do application-level packet content analysis by default, even when the intrusion prevention system (IPS) is running as a network device rather than directly on an end host. The lines between application firewalls and IPSs have been blurred even further in recent years [LWKS05].

One of the challenges of using both IPSs and firewalls is coordinating the two devices. An administrator should be able to configure all these devices so that overlap is minimized when desired, redundancy is maximized when desired, and that all gaps are known (and, if possible, closed). Our experience is that this is rare. Further, depending on the devices, there may not even *be* a way to coordinate these devices, given that many IPSs and firewalls use proprietary languages to describe their rulesets. In this case, an administrator may have a broad sense of what a given IPS rule does, but may find it challenging to identify with enough precision what the rule does in order to coordinate with the firewall rulesets.

One way to deal with encrypted traffic is to decrypt packets on the host (the end point), send them back to the firewall, let the firewall filter the packets after decryption, and then return them to the host. However, the latency and additional communication of doing this is prohibitive. A second possibility, which avoids a roundtrip of the traffic in question, is to use the firewall (or something very close to the firewall) as a proxy, have the firewall capture and buffer the traffic, obtain *only* the session key from the host, and then decrypt and pass the decrypted traffic to the host. By passing only the session key from the host behind the firewall to the proxy, the latency would be reduced dramatically. Similar arrangements could be used with SMB and the variety of applications that run on top of it.

The issue in dealing with both of these areas is speed. One can punch holes in firewalls to deal specifically with high-bandwidth, low-latency applications such as multi-party videoconferencing. But these applications have had large numbers of vulnerabilities. For example, RealPlayer, Windows Media, and QuickTime all support third-party codecs, and therefore act as a conduit for vulnerabilities just as web servers do for CGI scripts. Since the third-party codecs are not necessarily validated by the media players' vendors, these codecs might be less trustworthy and more vulnerable than if they came directly from the media players' vendors.

Though speed is a challenge, it is partially surmountable. Specialized hardware exists that can reassemble packets, decrypt streams, and process high-bandwidth network traffic all the way up to the application layer, and in contexts where bandwidth is (relatively) low this hardware may provide the support firewalls and IPS systems need. However, in contexts where bandwidth is particularly high, or where attacks may even span multiple very high bandwidth types, and intrusion alerts must be correlated before determining that an attack exists [VVKK04, ZHR+07], the use of these firewalls or IPSs may not be acceptable.

7 Conclusions and Future Work

The introduction to this paper posed four questions. The answers depend on the environment being defended. As with computer security in general and indeed all of computer science, there are good answers for specific environments, but there are few good answers, general for all, or even most environments.

One result is clear: the current use of firewalls as a protection for highly-vulnerable services is fundamentally flawed. The complexity of configuration, the insider problem, and a false sense of security leading to lax internal security policies, are all serious issues.

What *are* firewalls—particularly border firewalls—useful for today? Certainly, they can protect specific machines or sets of machines from access. That is, they can be used to create not just a "demilitarized zone" ("DMZ") between the Internet and the internal network, but an actual partition of the internal network, blocking many insider attacks as well.

Also, firewalls can rate-limit the effects of fast-moving worms and denial-of-service attacks from spreading behind the firewall. They can work with an intrusion prevention system to block specific attacks (e.g., dictionary attacks against an `sshd` daemon), or do so themselves. Clearly, automated methods of configuring multiple layers of network and host-based firewalls can be useful, providing they take into account the machines behind those firewalls and the services running on those machines. However, simply running fewer services and/or restricting those services with network-based access control mechanisms (e.g., TCP Wrappers [Ven92] or `hosts.allow`/`hosts.deny`) can be an equally effective, and potentially less vulnerable, than increasingly complex firewalls.

Not all damaging events can be predicted in advance, and it is important to allow employees to do their jobs and to allow customers and partners to access necessary resources. So simply allowing events to happen may be appropriate. On hosts, a very systematic and careful approach to forensic logging enables the reconstruction of a system in case something goes wrong [Pei07, PBKM05, PBKM07a, PBKM07b]. Similar techniques can work on a network, or a network of machines, by logging packets as they pass through firewalls. Firewall logs already contain significant security information ([Bac00], p.74),[7] but if a systematic approach to logging were used, then one could determine if the right information is being recorded.

Firewalls—application-based, distributed, or otherwise—will never be a panacea for defending against attacks, but they can certainly be built, configured, and used more effectively than they are now.

Empirical studies, such as ours, can also always be broader, deeper, and can analyze data over a longer period of time. Our future work involves all of these areas. One set of questions that we intend to examine is: are there any bugs in firewall implementations, and how many attacks actually get through? This requires gathering a certain amount of empirical forensic information. We also intend to study a broader collection of firewall rules.

In this paper, we have focused on analyzing a large set of attacks and vulnerabilities and a small set of firewall rules. An ideal future study would look at a larger set of firewall rules from a variety of different enterprises, security domains, and system administrators to determine how different policies are implemented, and which are more or less effective at preventing attacks. These studies can help us better understand the environments in which specific defenses are appropriate and useful.

[7] Firewall logs are also used as evidence in court ([SB03], pp. 247–249).

Acknowledgements

The authors wish to thank Dr. Johannes Ullrich at the Internet Storm Center for providing valuable attack data used in this paper. The authors also wish to thank the anonymous providers of the firewall rulesets examined in this paper.

Sean Peisert was supported in part by the National Science Foundation under Grant Number CNS-0831002. Matt Bishop was supported in part by the National Science Foundation under Grant Number CNS-0716827.

The opinions, findings, and conclusions contained in this document are those of the authors and should not be ascribed to and do not necessarily reflect the views of the funding sources of any author.

References

[App07] Apple Computer, Inc. Mac OS X 10.5 Leopard: About the Application Firewall. `http://docs.info.apple.com/article.html?artnum=306938`, November 15 2007.

[ASH04] Ehab S. Al-Shaer and Hazem H. Hamed. Discovery of Policy Anomalies in Distributed Firewalls. In *Proceedings of IEEE INFOCOM*, pages 2605–2616, March 2004.

[Bac00] Rebecca Gurley Bace. *Intrusion Detection*. Macmillan Technical Publishing, 2000.

[BEF+10] Matt Bishop, Sophie Engle, Deborah A. Frincke, Carrie Gates, Frank L. Greitzer, Sean Peisert, and Sean Whalen. A Risk Management Approach to the 'Insider Threat'. In *Insider Threats in Cybersecurity*, Advances in Information Security Series. Springer Verlag, Berlin, September 2010.

[BEP+08] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. We Have Met the Enemy and He is Us. In *Proceedings of the 2008 New Security Paradigms Workshop (NSPW)*, Lake Tahoe, CA, September 22–25, 2008.

[BEP+09] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. Case Studies of an Insider Framework. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS), Cyber Security and Information Intelligence Research Minitrack*, Waikoloa, HI, Jan. 5–8, 2009.

[Bis05] Matt Bishop. The Insider Problem Revisited. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 75–76, Lake Arrowhead, CA, October 20–23, 2005.

[BMNW04] Yair Bartal, Alain Mayer, Kobbi Nissim, and Avishai Wool. Firmato: A Novel Firewall Management Toolkit. *ACM Transactions on Computer Systems (TOCS)*, 22(4):381–420, November 2004.

[CCBGA05] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquín García-Alfaro. Detection and Removal of Firewall Misconfiguration. In *Proceedings of the 2005 IASTED International Conference on Communication, Network and Information Security (CNIS)*, pages 154–162, 2005.

[EZ01] Pasi Eronen and Jukka Zitting. An Expert System For Analyzing Firewall Rules. In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec)*, pages 100–107, Copenhagen, Denmark, Nov. 2001.

[Fyo08] Fyodor. "telnet still tops open port list". `http://www.securityfocus.com/brief/794?ref=rss`, August 10 2008.

[IKBS00] Sotiris Ioannidis, Angelos D. Keromytis, Steven M. Bellovin, and Jonathan M. Smith. Implementing a Distributed Firewall. In *Proceedings of the 7th ACM International Conference on Computer and Communications Security (CCS)*, pages 190–199, Athens, Greece, November 2000.

[KKVJ06] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic. Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks. *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 330–337, 2006.

[KP07] Angelos D. Keromytis and Vassilis Prevelakis. Designing Firewalls: A Survey. In Christos Douligeris and Dimitrios N. Serpanos, editors, *Network Security: Current Status and Future Directions*, pages 33–49. Wiley - IEEE Press, April 2007.

[LWKS05] Michael E. Locasto, Ke Wang, Angelos D. Keromytis, and Salvatore J. Stolfo. FLIPS: Hybrid Adaptive Intrusion Prevention. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID 2005)*, pages 82–101, Seattle, WA, Sept. 7-9 2005.

[MPS+03] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer Worm. *IEEE Security & Privacy*, 1(4):33–39, July/August 2003.

[MwL+07] Rich Mogull, windexh8er, Rob Lee, Chris Pepper, Apple Computer, and FreeBSD, Inc. ipfw Rules. `http://securosis.com/publications/ipfw.html`, December 12 2007.

[MWZ06] Alain Mayer, Avashai Wool, and Elisha Ziskind. Offline Firewall Analysis. *International Journal of Information Security*, 5(3):125–144, 2006.

[PBKM05] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Principles-Driven Forensic Analysis. In *Proceedings of the 2005 New Security Paradigms Workshop (NSPW)*, pages 85–93, Lake Arrowhead, CA, October 2005.

[PBKM07a] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Analysis of Computer Intrusions Using Sequences of Function Calls. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(2):137–150, April–June 2007.

[PBKM07b] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. Toward Models for Forensic Analysis. In *Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 3–15, Seattle, WA, April 2007.

[Pei07] Sean Philip Peisert. *A Model of Forensic Analysis Using Goal-Oriented Logging*. PhD thesis, Department of Computer Science and Engineering, University of California, San Diego, March 2007.

[SB03] Fred Chris Smith and Rebecca Gurley Bace. *A Guide to Forensic Testimony: The Art and Practice of Presenting Testimony As An Expert Technical Witness*. Addison Wesley Professional, 2003.

[Sec00] SecurityFocus. Multiple Firewall Vendor FTP Server Vulnerability. `http://www.securityfocus.com/bid/979`, February 9 2000.

[Sec01] SecurityFocus. Check Point Firewall-1 RDP Header Firewall Bypassing Vulnerability. `http://www.securityfocus.com/bid/2592`, July 9 2001.

[Sec02a] SecurityFocus. Symantec Enterprise Firewall SMTP Proxy Information Leak Vulnerability. `http://www.securityfocus.com/bid/4141`, Feb 20 2002.

[Sec02b] SecurityFocus. Symantec Raptor / Enterprise Firewall FTP Bounce Vulnerability. `http://www.securityfocus.com/bid/4522`, April 16 2002.

[Sec04a] SecurityFocus. Microsoft Windows Internet Connection Firewall Filter Bypass Vulnerability. `http://www.securityfocus.com/bid/10930`, August 12 2004.

[Sec04b] SecurityFocus. Microsoft Windows XP Firewall ACL Bypass Vulnerability. `http://www.securityfocus.com/bid/12057`, December 20 2004.

[Sec07a] SecurityFocus. Apple Mac OS X 10.5 Application Firewall Misleading Configuration Weakness. `http://www.securityfocus.com/bid/26461`, November 15 2007.

[Sec07b] SecurityFocus. Microsoft Windows Vista Teredo Interface Firewall Bypass Vulnerability. `http://www.securityfocus.com/bid/24779`, July 10 2007.

[SEVS04] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated Worm Fingerprinting. In *Proceedings of the 6th ACM/USENIX Conference on Symposium on Operting Systems Design & Implementation (OSDI)*, pages 45–60, Berkeley, CA, December 2004.

[Sin03] Abe Singer. Life Without Firewalls. *;login:*, 28(6):34–41, December 2003.

[Sin05] Abe Singer. Tempting Fate. *;login:*, 30(1):27–30, February 2005.

[SM04] Colleen Shannon and David Moore. The Spread of the Witty Worm. *IEEE Security and Privacy Magazine*, 2(4), 2004.

[Smi94] Richard E. Smith. Constructing a High Assurance Mail Guard. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, MD, October 1994.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[UC07] Tomás E. Uribe and Steven Cheung. Automatic Analysis of Firewall and Network Intrusion Detection System Configurations. *Journal of Computer Security*, 15(6):691–715, 2007.

[Ven92] Wietse Venema. TCP WRAPPER: Network monitoring, access control, and booby traps. In *Proceedings of the 3rd USENIX Security Symposium*, Baltimore, MD, September 1992.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 1(3):146–169, July–September 2004.

[Woo01] Avishai Wool. Architecting the Lumeta Firewall Analyzer. In *Proceedings of the 10th USENIX Security Symposium*, pages 85–97, Washington, D.C., August 2001.

[Woo04] Avishai Wool. A Quantitative Study of Firewall Configuration Errors. *IEEE Computer*, 37(6):62–67, June 2004.

[YMS+06] Lihua Yuan, Jiannina Mai, Zhendong Su, Hao Chen, Chen-Nee Chuah, and Prasant Mohapatra. FIREMAN: A Toolkit for FIREwall Modeling and ANalysis. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 199–213, Oakland, CA, May 2006.

[ZHR+07] Jingmin Zhou, Mark Heckman, Brennan Reynolds, Adam Carlson, and Matt Bishop. Modelling Network Intrusion Detection Alerts for Correlation. *ACM Transactions on Information and System Security (TISSEC)*, 10(1), February 2007.