Automatic Signature Generation Final Report: Addressing Limitation of Approach for Self-Propagating Attacks

Version 1.1

Todd Heberlein
Net Squared, Inc.

27 Aug 2004

Report developed under SBIR contract for Topic No. AF03-094. One of the biggest threats facing the Internet and large organizations are self-propagating attacks (e.g., worms), and of special concern are so-called zero-day worms targeting previously unknown vulnerabilities. The automatic signature generation project is designed to quickly generate a signature for a previously unknown attack, and when used in conjunction with intrusion prevention devices (IPDs) can provide a rapid countermeasure to a newly discovered attack. However, unless IPDs are distributed throughout an organization, a potentially expensive proposition, our proposed solution may have only limited success against a worm that initially gets past the perimeter IPD. This paper addresses the critical flaw in the design. By properly using the filtering capability of commodity routers we create a defense in depth capability that allows us to contain and stop a zero-day worm.

Table of Contents

List of Figures

1 Introduction

Computer science researchers typically work with formal abstractions allowing them to "prove" certain features about their models. However, in real operational environments with real users, real missions, and real constraints (training, time, costs, etc.), "formal" or "obvious" efforts frequently break down, are unknown, or are ignored for more pragmatic solutions. For example, a researcher may begin "assume the existence of a trusted computing base", at which point the rest of the presentation's practical applicability in real world operations becomes suspect. As researchers we should strive to bridge the world of research with practical operations.

Two of our current research efforts focus on automatic signature generation to address fast-moving zero-day worms and attack graph analysis to identify optimal changes needed to harden a network against cascading attacks. While we have made progress towards developing solutions for these threats, we have also identified a number of challenges to our solutions, and we have been looking for alternative approaches (outside our specific lines of research) to lessen the effects of these challenges on our solutions.

Interestingly, we identified a single approach that helps address many of the challenges to both research efforts, and not only does this approach help the research efforts produce better results, the approach can also be effectively used as an inexpensive stand-alone solution. Furthermore, organizations can deploy the approach today without needing to purchase expensive and exotic hardware, and we believe it will fit relatively well into operational real-world networks. For example, we have already deployed the approach in our own operational networks. Finally, we believe the approach can provide a baseline against which other solutions to the threats can be measured.

Section 2 presents two important threats (fast spreading server-based worms and cascading attacks that allow an attacker to penetrate deeply into an organization), the defense strategy others and we are researching to address these threats, and challenges to the proposed defense approaches. Section 3 presents the simple but potentially very effective approach that by themselves can help address the threats presented in Sections 2.1.1 and 2.2.1 but also can make the solution under development even better. Section 4 shows how the approach presented corresponds very closely with good operating system design principles. Section 5 summarizes the paper. Finally, Appendix A examines in detail how the approach provided in this paper works with our automatic signature generation system and intrusion prevention devices to provide a defense in depth capability against zero-day worms.

2 Motivations

The approach presented in this paper is not the focus of our research. We are focusing on the threats posed by worms, in particular server-based worms against previously unknown vulnerabilities, and maximally hardening a network given a limited number of changes that can be made. In this section we summarize each of these threats, briefly discuss the defense strategies we (and others) are taking to address these threats, and present some of the challenges that can limit the effectiveness of the proposed solutions.

It is the challenges to the proposed defense strategies that motivates us to present this work, because the simple approach presented in Section 3 ameliorate some of the difficulties facing the research solutions and provide value in their own right.

2.1 Worm-Based Threats

2.1.1 Salient Features of Worms

For the purpose of this discussion we focus on server-based worms, and we define this class of worms as ones that attack network servers and do not need human interaction to complete the process. For example, most email-based worms require (1) a client application to download a malicious email message from a server and (2) a user to open a malicious attachment. A server-based worm, on the other hand, takes advantage of some flaw in a server application (e.g., a web server) to infect the server and then move on without requiring a client application or a user to perform any additional steps.

One of the most dangerous aspects of a worm is the speed in which it can propagate across the network, and the primary reasons behind this speed are that the worm attacks in parallel and each infected system becomes an attacker itself. To illustrate this issue Figure 1 shows three common attack patterns. In each subfigure we track the penetration of eight hosts over time. A circle represents a host, and time progresses from left to right. A single row represents a single host as it progresses from a non-compromised state to a compromised state. We color the circle black on the round that the host is penetrated, and for all subsequent rounds, during which the host is still considered compromised, we color the circle grey.

The left-most pattern represents a human attacker working his was across multiple systems. Starting at the initially penetrated host, the attacker attacks a single host in round 2 (the top-most host). It is from this newly penetrated host that the attack penetrates the next host, and so on. The attack pattern is very serial. This was the common tactic used by attackers until about 1993 and is typified by Cliff Stoll's "wily hacker" [Stol 90] The middle pattern represents a scan-based attack. In this approach the attacker, usually with automated software, attacks each host from a single system (or small number of systems), but each newly penetrated host is not used to automatically propagate that attack (at least not initially). Around 1993 attackers started using this technique to quickly scan and initially penetrate a large number of systems at an organization, and attackers still use this method for initial penetration or to gather resources (e.g., zombies) to be used for other purposes (e.g., DDOS attacks or for sending spam mail).

The right most pattern represents a server-based worm attack, and it essentially combines the strategies of the first two patterns. Like a scan-based attack a worm-based attack can quickly attack a number of systems. Like a human-based attack of the first pattern, each newly penetrated system is used to attack additional systems. The result is a fast parallel attack with an exponential spread rate.

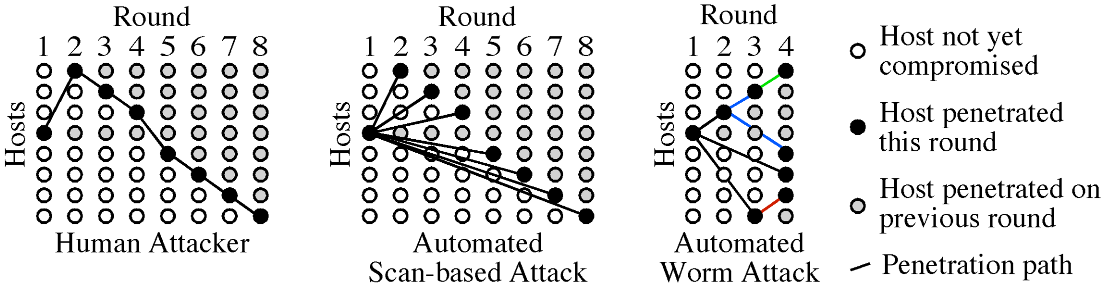


Figure 1: Penetration Mechanisms and Patterns

While the simple graphs of scan-based and worm attacks in Figure 1 seems to imply that the worm attack is only twice as fast as the scan-based attack, the penetration of the worm attack is much faster (with exponential growth) than the scan-based attack (with linear growth). While the difference is minimal after just four rounds, the difference is dramatically more apparent after just 16 rounds. Figure 2 demonstrates the difference between the two attack methods: each cell represents the total number of systems penetrated by each method (vertical axis on the left) by a given round (horizontal axis across the top). By round 16 the scan-based attack has only penetrated 16 systems, while the worm has penetrated over 32,000 systems. The primary threat from worms is the rate at which they infect systems, and that speed is the result of using each newly penetrated system to attack new systems.

	Rounds									
	1	2	3	4	5	6	7	8	···	16
Worm	1	2	4	8	16	32	64	128	···	32,768
Scan Attack	1	2	3	4	5	6	7	8	···	16

Figure 2: Rate of Penetration of Worms vs. Scan-based Attacks

2.1.2 Defense Strategy to Server-based Worms

Our primary research focuses on fast moving attacks, including server-based worms and scanning attacks, which exploit unknown vulnerabilities. Because the vulnerability is unknown, no patch is available, thus the attack will have free reign for at least several hours and maybe several days before vulnerable systems can be patched. As we have seen in the case of worms, most vulnerable systems are penetrated within a few hours. For systems that are not mission critical, simply turning off the service or not using the vulnerable application may be a reasonable response, but for mission critical services this is not an option.

Without a viable host-based solution we are developing techniques to take advantage of the network infrastructure. In particular, we are taking advantage of the so-called “intrusion prevention” devices, essentially integrated intrusion detection and firewall systems, that can drop packets and terminate connections when an attack signature is detected. The challenge is creating a signature for the newly detected attack that is of high enough quality (high true positive and low false positive rates) so that network managers will have the confidence to deploy the technology and fast enough (a few seconds at most) to be effective against fast moving attacks.

2.1.3 Challenges to Automatic Signatures and Intrusion Prevention Devices

While we have confidence in our research approach, it still face challenges. First, given that our time to develop a signature is relatively fixed, the exponential spread rate of worms tell us that a much greater percentage of our network will be infected before a defense strategy can be deployed than would be the case for a scan-based attack with its linear spread rate. Second, since the attack is new and against an unknown vulnerability, the first instance of the attack will get through to a potentially vulnerable system before we can deploy a new signature (i.e., the attack must occur before we can detect it), so if intrusion prevention device is only deployed at the perimeter of an organization, a worm attack will be able to continue attacking internal systems inside an organization even after a signature is deployed to the perimeter intrusion prevention device.

2.2 Cascading Penetration Threats

2.2.1 Introduction to Cascading Attack

Whereas the primary threat of worms is the rate at which they can infect a large number of systems, cascading penetrations, often the result of a clever human adversary, allow an attack to reach deep inside an organization by taking advantages of new capabilities acquired at each stage. Figure 3 illustrates the basic concept. An attacker beginning outside an organization penetrates a web server that is available through the firewall. Once inside the firewall the adversary takes advantage of a buffer overflow in ssh to penetrate a second system. This second system is not accessible to the outside world, so the adversary would not be able to attack it directly. From this second host the adversary takes advantage of a trust relationship to penetrate a third host – a host that neither is accessible from the outside nor has any programming vulnerabilities.

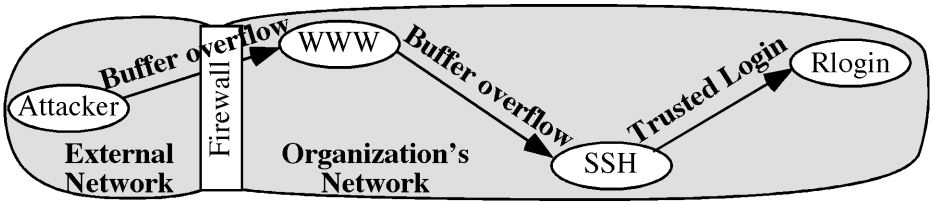


Figure 3: Cascading Attack Inside Organization

One of the primary reasons that cascading attacks are often successful in the wild is the large number of unpatched vulnerabilities in organizations. While the initial solution to the problem is to simply patch all the vulnerabilities, there exist many reasons that this does not occur in practice. Some reasons include: system administrator's time is limited and they often cannot patch everything all the time; applying patches often require taking systems offline for periods of time; patches often introduce instabilities, incompatibilities, and new vulnerabilities; people responsible for an organization's overall network security often do not have the authority to make changes in individual systems; and untrained administrators (i.e., a user) are often responsible for installing and maintaining personal computer operating systems and applications that are as complex as the mainframe systems of a decade ago. For these and other reasons, moderate to large operational networks frequently have large numbers of vulnerabilities.

A number of strategies exist for prioritizing which vulnerabilities should be addressed first. Point-based strategies, typically provided by vulnerability scanner vendors, look at the capability that a vulnerability provides an attacker and recommends first fixing the vulnerabilities that give the attackers the greatest capabilities. For example, the system would recommend fixing a vulnerability that allows a remote attacker to run arbitrary commands as root or administrator before fixing a vulnerability that allows a local user to read a protected file. A probability-based strategy, typically provided by intrusion monitoring services, ranks the vulnerabilities by how frequently they are attacked. The vulnerability that is attacked most frequently will be given a higher priority than a vulnerability rarely or never attacked. Goal-based strategies, a topic of much research today, examines how one vulnerability can interact or enable other vulnerabilities and how these interactions allow an adversary to achieve some particular goal. Vulnerabilities are ranked on their importance with respect to allowing an adversary to achieve a particular goal. Research in this area is often referred to as "attack graph" analysis, and is the focus of the next section.

2.2.2 Attack Graph Analysis Approach to Cascading Attacks

While attack graph analysis may be considered a branch of fault-tree analysis [Gart 99], we trace the current work back to Robert Baldwin's Kuang system developed in the late 1980s [Bald 94] and distributed as part of the COPS security package [Farm 90]. This system looked for a single path on a single Unix computer from some an initial condition (e.g., an unprivileged user) to a final condition (e.g., becoming root) by exploiting configuration files (e.g., leaving an important file world writeable). Later Dan Zerkle and others extended the basic Kuang concept to a network of systems [Zerk 96]. More recently there has been a flurry of research to extend attack graph concepts to look for all paths from an initial condition to an end goal and to include vulnerability exploits in the generation of those paths [Shey 02] [Amma 02] [Noel 03].

In general attack graphs describe how an adversary, either an individual at a keyboard or automated software, can through a series of steps acquire new capabilities. These steps usually involve the exploitation of a vulnerability. Four our discussions here, using [Bish 02] we define a vulnerability as follows:

> *A vulnerability is a set of conditions that, when all are true, can allow for a violation of policy.*

Figure 4 provides a graphic display of this definition. An example of a vulnerability using this definition might be: code in a network server does not perform an array bounds check (i.e., it has a buffer overflow bug) (condition 1); the code for the server is running (condition 2); the CPU and operating system allows instructions to be run from the stack (condition 3); and firewalls and routers allow the adversary's host to send data to the server (condition 4); thus the adversary can execute arbitrary commands with the server's UID (violation of policy).

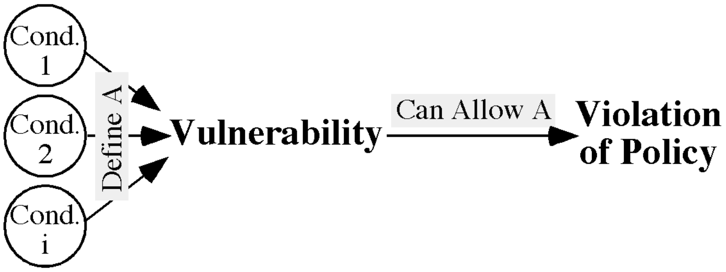
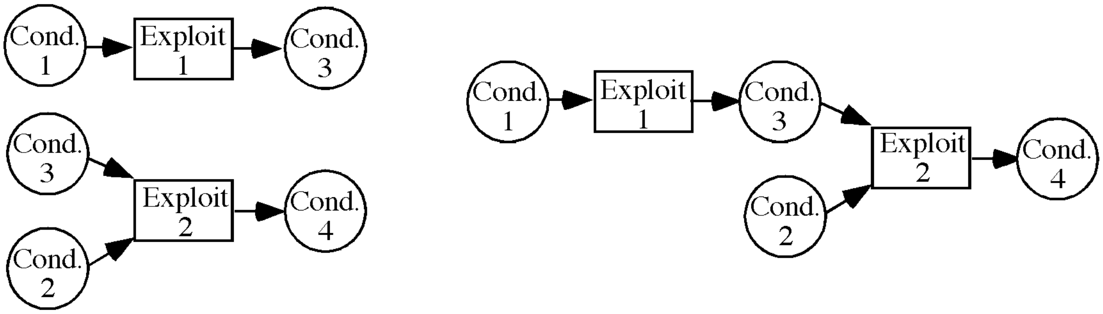


Figure 4: Vulnerability Definition

To negate the vulnerability, only one of the conditions needs to be removed. For example, future CPU and OS designs will probably prevent the CPU from executing code located on the stack, and this will block a large number of vulnerabilities. Also, in the early days of UNIX (e.g., SunOS and Ultrix) and later with Linux, the default installation for these operating systems include many servers that simply did not need to be running. By shifting to a paradigm where only required services are active by default (a direction more vendors are taking), many vulnerabilities are removed from the system.

Attack graph analysis identifies how a set of conditions (a vulnerability) can be exploited, and the result of that exploitation (a new condition) contributes to a new vulnerability that can be exploited. In other words the second vulnerability does not exist until the first vulnerability is exploited.



Two Individual Exploits **Chaining Two Exploits To Achieve Condition 4**

Figure 5: Attack Graph Analysis

Figure 5 demonstrates the concept. In the left column are two potential vulnerabilities that can be exploited to create new conditions. If in an operational system *Condition 3* is not present, then the second vulnerability does not exist. However, the right column shows how by exploiting the first vulnerability an adversary can "activate" the second vulnerability. The adversary can then exploit this second vulnerability to achieve *Condition 4*.

Our attack graph research primarily focuses on developing the capabilities to identify the chains of conditions and exploits (the adversary's "attack graph") and then identifying optimal strategies for disrupting the adversary's attack graph.

2.2.3 Challenges to Attack Graph Approaches

There are a number of challenges for the research including data acquisition, efficient graph building algorithms, and ensuring that you have adequate coverage of the exploit space. There are also more fundamental problems. The attack graph strategy is based on the assumption that we know the attacker's initial conditions (what privileges and on what systems they will initially have, what tools or knowledge they possess, etc.) and what the adversary's goal might be. For example, if we assume the adversary has more capability than one the network will actually encounter, our solution for changes to the network will be less than optimal (i.e., more work was done than was required). More importantly, the graph-based strategy presumes the adversary has a particular goal (e.g., achieve administrator access on the host secrets.mybase.mil), and if we guess the wrong goal or the adversary does not have a specific goal, the proposed changes to the network may have little relevance. Finally, today's attack graph research and development is based around *known* vulnerabilities and exploits, so if the adversary has knowledge or capabilities that we did not model, the attack graph recommended strategy may be irrelevant.

While our research is examining a number of approaches to address these challenges, we also looked at a very simple strategy that addresses many of these concerns. Interestingly, the strategy also addresses the concern affecting our research on stopping fast moving worms.

3 A Simple Strategy To Addressing Worms and Cascading Attacks

This section presents a simple strategy that organizations can apply today using commodity hardware and based on well-known design principles that can have substantial impact on worms and cascading attacks. For many organizations that do not have the need or resources to deploy automatic signatures to address zero-day worms or perform deep and complex analysis of their network to determine the ways vulnerabilities and configuration settings can interact, these simple steps may be enough. For those with the needs and resources to deploy these advanced technologies or perform the deep analysis, these simple steps can greatly increase the effectiveness of these technologies and analyses.

Section 3.1 presents the primary motivation for the strategy. Section 3.2 presents the two simple steps that organizations can apply today. When applied these steps can have a dramatic impact on worms and cascading attacks and increase the performance of a number of research activities.

3.1 Network as Kernel

The primary insight leading to the simple approach presented below is that a multi-user operating system is designed with the assumption that individual users and programs cannot be trusted to behave, and we should design our networks with a similar assumption. In other words, we want to design our network infrastructure assuming the individual elements using it cannot be trusted. Both the operating system kernel and the network infrastructure facilitate access between elements (e.g., processes for an OS and hosts for a network), and they also (or at least should) mediate the ways one element can interact with another element.

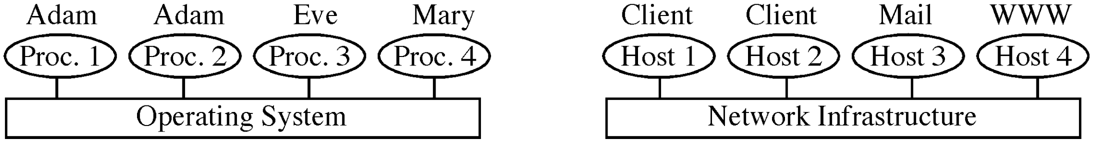


Figure 6: Host OS and Network Infrastructure Roles

Figure 6 demonstrates the similarities of the roles of an operating system kernel and network infrastructure. On the left are several processes (Proc. 1-4), each "owned" by a particular user. The operating system provides a number of ways for each of these processes to communicate (e.g., sending data or signals), but the operating system also enforces limits on these interactions. For example, process 1 and process 2 are both owned by the user Adam, so the operating system may allow process 1 to "kill" process 2. However, because process 3 is owned by Eve, the operating system should prevent process 1 from killing process 3. On the right, are several hosts, each playing a different role, and the network infrastructure provides a similar capability to these hosts that the OS kernel plays on the single computer. Because Host 1 is a client, it should be able to initiate a communications with the Mail server and Web server; however, in general the Web server should not be able to initiate connections to Host 1.

Operating systems are rarely developed and deployed (only a handful of operating systems control the vast majority of systems in the world), and they are typically developed by a team of experts trained in the various aspects of operating system design. Network infrastructures, on the other hand, are newly built for each individual site, and these network infrastructure developers rarely have the depth of training or the time of those developing operating system kernels. As a result, the security provided by the network infrastructure "kernels" are more similar to the security provided by the old Microsoft DOS operating system than they are to the more modern Windows XP or UNIX-class operating systems. Our goal, therefore, is to develop an approach, or rules of thumb, that is simple enough to be easily applied yet powerful enough to provide many of the desired capabilities.

3.2 The Two Simple Steps

The approach that helps slow the exponential growth rate of worms and makes it harder for an attacker to penetrate deeply into an organization has two steps: separate functionality onto separate hosts and apply the simplest of rules to the router to limit a host's actions to its necessary functionality.

3.2.1 Separating Functionality

The first step to providing the additional security that prevents the exponential spread rate of worms and deep penetration by cascading attacks is to separate the activity performed by a single host. A router understands IP addresses; a router does not, in general, distinguish the process or application that generates the packets. This means that if a host performs activities that are difficult for the router to distinguish from an attack, then the router probably cannot block the attacks.

Figure 7 illustrates the problem and solution. On the left side we have a single host running both a web client and web server. The web client routinely performs outbound client requests, which the router must allow. However, an inbound attack (dashed line) compromises the server and then performs outbound connection requests to spread the attack. The router cannot distinguish the legitimate outbound web requests and the attack, so the router allows everything.

The right side of Figure 7 shows our new and preferred network architecture. The web client and web server are placed on two different hosts (with two different IP addresses), and the hosts placed on two different networks. Because the expected activity (i.e., “the roles”) of the two hosts are different, the router can enforce access control rules (ACLs) that allow legitimate activity to continue while preventing the adversary from using the penetrated server to spread the attack.

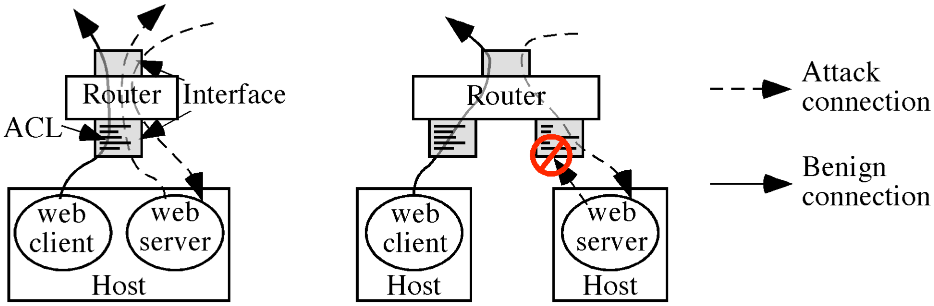


Figure 7: Separating Functionality

3.2.2 Limiting Functionality

By separating the roles of the client and server on different hosts (and on different network interfaces), we can apply some simple access control list rules to the two interfaces to prevent the clients from being attacked and prevent the compromised server from being used to attack additional hosts. The Cisco IOS operating system (as well as virtually all router and firewall vendors) supports the application of separate access control lists to packets leaving the router (the “out” list) and entering the router (the “in” list) (see Figure 8). For our purpose, the configuration of the interfaces is simplified by (1) our separation of roles for the two interfaces and (2) the Cisco IOS reflect table feature that allows us to tie together the in and out packet flows.

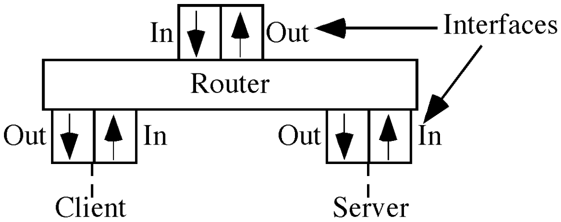


Figure 8: Configuring In and Out Interfaces

A common configuration for networks that support hosts that only act as clients is to allow packets into the network that are in response to outbound requests. By default, network appliances that support dynamic Network Address Translation (NAT) (or what Cisco describes as NAT with Port Address Translation (PAT)) provide this capability by default. To provide this capability for a router supporting the Cisco IOS command set we create the following access controls and add them to the appropriate router interface (we call it client_interface).

```
ip access-list extended out_of_client_network
   permit tcp any any reflect client_session_req

ip access-list extended into_client_network
   evaluate client_session_req

interface client_interface
   ip access-group out_of_client_network in
   ip access-group into_client_network out
```

Figure 9: Protecting Clients From External Attacks

The first ACL, named "out_of_client_network", allows all outbound connection requests from the client and then records the requests in the table "client_session_req". The second ACL, named "into_client_network", allows in any packet that matches an outbound request but blocks everything else. These two ACLs are then applied to the appropriate interface.

To stop the server from contributing to the exponential spread of a worm or allowing it to contribute to cascading penetration into an organization we apply nearly identical filter rules to the server interface; the primary difference is that we are protecting the rest of the world from a potentially compromised server.

```
ip access-list extended into_server_network
   permit tcp any any reflect server_session_req

ip access-list extended outof_server_network
   evaluate server_session_req

interface server_interface
   ip access-group out_of_server_network in
   ip access-group into_server_network out
```

Figure 10: Preventing Server From Attacking Other Systems

The first ACL, named "into_server_network", allows all inbound connection requests to the server and then records the request in the table "server_session_req". The second ACL, named "outof_server_network" only allows packets out of the network that are in response to a specific request (i.e., matches an entry in the server_session_req table).

Using commodity hardware, these two simple ACL rules, one applied to inbound server requests and one applied to outbound server replies, provide a powerful level of defense against two of the most important threats facing the Internet and individual organizations. Certainly additional refinement can be applied these rules (e.g., limiting inbound connections into the server only to a specified port, or allowing limited access to other hosts such as network time protocol (NTP) servers), but these simple rules provide a sound foundation.

4 Saltzer and Schroeder's Design Principles

As mentioned in Section 3 our goal is to, in effect, use the routing infrastructure as a network kernel (i.e., providing and *mediating* access between elements in the network). This prevents bad elements in the network (e.g., a compromised server) from compromising more of the network. Indeed, the simple strategy presented here of separation of roles and mediated access follow closely to the design principles for operating systems specified nearly three decades ago by Saltzer and Schroeder [Salt 75]. We summarize some of those principals and how this network approach is mapped to them:

- Economy of mechanism. The network design presented in this paper is very simple, using two simple ACL rules available on commodity hardware. Other efforts to control the exponential spread rate of worms or prevent cascading penetrations typically require much more complex solutions or analysis.
- Fail-safe defaults. By default, any outbound packets from the server network that are not explicitly matched to an inbound request are dropped. By default, the design drops packets (a property of the implicit "deny" at the end of any ACL). Another ACL rule can be added to report any of these failed packets so there are no silent failures. This can also serve as a detection mechanism if the server is compromised and the attack tries to attack additional systems.
- Complete mediation. As presented in this paper, all packets to and from the server are mediated by the ACL rules in the router. We suspect most sites will have multiple servers on the same side of the mediated interface (e.g., in a DMZ LAN), and in this case the router cannot prevent the servers on the same LAN from infecting each other. However, we hope that by using commodity hardware most sites will place a router as close the server possible.
- Open design. The design and router features used in this design are well known.
- Separation of privilege. In our case, we are using "separation of duty" (placing the client and server roles on different hosts), which is often considered a type of separation of privilege. The separation of duty is what allows us to apply simply rules but gain a considerable amount of security.
- Least privilege. In the design provided here, the server can only send out packets in reply to a request from a client. The server cannot send out any other packets unless explicitly allowed by additional rules.

5 Conclusions

Two of the biggest threats facing the Internet and large organizations are worms and cascading penetration. Our own research efforts in attack graphs and automated signature generation, as well as similar research performed by a number of organizations, are designed to identify optimal strategies for reducing these threats, but they often use complex analysis and expensive hardware. Furthermore, the solutions under research today face a number of challenges that limit their potentials for success.

The key to both of the threat of exponential growth of worms and the cascading penetration into an organization is that each penetrated system can be used to attack additional systems. By addressing this critical fact using separation of roles and enforcing those roles with simple ACLs in commodity routers, we greatly reduce the twin threats of server-based worms and cascading attacks without deploying any new hardware or performing complex analysis and improve the potential effectiveness of research under development in these areas.

In the final analysis we have only rediscovered an approach that was described in a seminal paper in 1975, and we identified how this solution to the threats can ameliorate the challenges to some of the more exotic solutions currently being researched. Ultimately, we believe that the approach presented here will serve as a baseline when quantifying the value provided by more advanced research. For example, we might compare the penetration rates for a worm against (1) a network with no additional security, (2) a network using this simple separation of roles and router-based enforcement of roles, (3) a network deploying some newly proposed approach, and (4) a network using the role separation and enforcement along with the newly proposed approach.

6 References

[Amma 02] Paul Ammann, Duminda Wijesekara, Saket Kaushik, “Scalable, Graph-Based Network Vulnerability Analysis”, *Proceedings CCS02: 9th ACM Conference on Compuer and Communication Security*, pp. 217-224, Washington, DC, Nov 2002.

[Bald 94] Robert Baldwin. Kuang, “Rule based security checking”, Technical report, MIT Lab for Computer Science, Programming Systems Research Group, May 1994.

[Bish 02] Matt Bishop, *Computer Security: Art and Science*, Addison-Wesley Publishing, Dec. 2002.

[Farm 90] Dan Farmer, Eugene H. Spafford, “The COPS Security Checker System”, *USENIX*, 1990.

[Gart 99] Felix Gartner, “Fundamentals of Fault-tolerant distributed computing in asynchronous environments”, *ACM Computing Surveys*, Vol.31, No.1, March 1999.

[Noel 03] Steven Noel, Sushil Jajodia, Brian O’Berry, Michael Jacobs, “Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs”, *19th Annual Computer Security Applications Conference*, pp. 86-95, Las Vegas, NV, Dec 8-12, 2003.

[Salt 75] Jerome H. Saltzer, Micael D. Shroeder, “The Protection of Information in Computer Systems”, *Proceedings of the IEEE*, pp. 1238-1308, Sept. 1975.

[Shey 02] Oleg Sheyner, et al., “Automated Generation and Analysis of Attack Graphs”, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 254-265, May 2002.

[Stol 90] Cliff Stoll, *The Cuckoo’s Egg*, New York, Bantam Books, 1990.

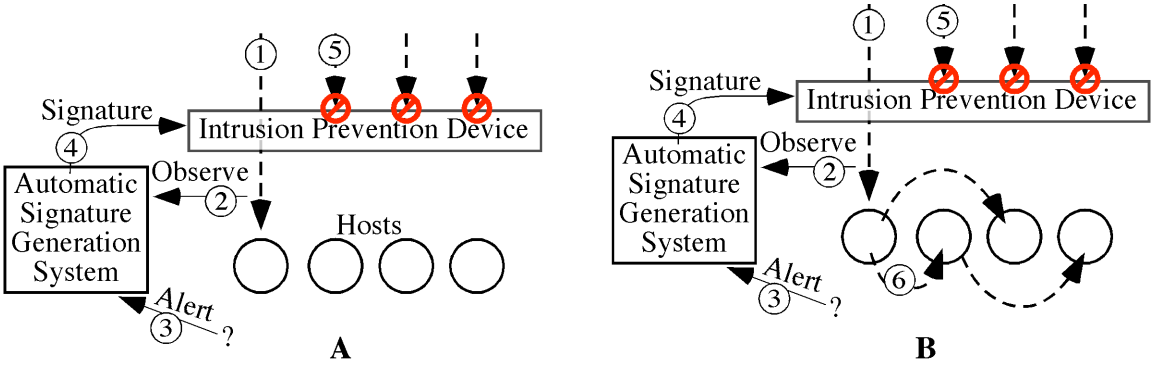
[Zerk 96] D. Zerkle, K. Levitt , "NetKuang--A Multi-Host Configuration Vulnerability Checker". *Proceedings of the 6th USENIX Security Symposium.* San Jose, California, July 22-25, 1996, pp. 195-204.

Appendix A

As mentioned previously, the work that led to the router configuration recommendations in the paper has been partially inspired by limitations with our automatic signature generation project. In this Appendix we examine these issues in more detail.

Our automatic signature generation project generates optimally high quality network content signatures that can be used to detect a newly observed attack. The signature can be deployed in so called intrusion prevention devices, essentially firewalls with intrusion detection capabilities that can block traffic matching an attack pattern, so hosts can be protected from network attacks even if the hosts are not patched. This is particularly important in cases for which there is no available patch (e.g., a new attack against an unknown vulnerability). The automatic signature generation's primary benefit is that it can generate a high quality signature very quickly, so a countermeasure can be put in place before most of the network is affected.

Figure 11A demonstrates how our system is designed to work. (1) The initial attack passes the intrusion prevention device (IPD) and infects the host. Because this is a previously unknown attack, the IPD lets the traffic through. (2) The Automatic Signature Generation System (ASGS) observes the attack; although, at this point it is unaware that the traffic represents an attack. (3) Some security element outside our system (e.g., a host-based IDS) detects the attack and alerts the ASGS. (4) The ASGS uses the traffic of the observed attack in step (2) to generate a signature that optimally represents that attack (high true positive and low false positive rates) and sends the signature to the IPD. (5) Subsequent attacks are now blocked by the IPD, so the remaining systems are protected. This is how the automatic signature generation system can stop fast-moving attacks.



1
5
Signature
4
Intrusion Prevention Device
Automatic Signature Generation System
Observe
2
Hosts
Alert
3
?
A
1
5
Signature
4
Intrusion Prevention Device
Automatic Signature Generation System
Observe
2
Alert
3
?
6
B

Figure 11: Performance Against Scan-based and Worm-based Attacks

Unfortunately we face two potential problems: the IPD can only block attacks that pass through it and we can only generate a signature for an attack once the first instance has occurred. For scan-based attack like the one shown in Figure 11A these are not major problems. The first host is compromised, but the other hosts are protected and once the first infected host recovers (e.g., by a reboot from a clean version of the OS) it cannot be re-infected. However, for self-propagating attacks (i.e., a worm) the limited coverage of the IPD becomes a major problem.

Figure 11B illustrates the problem. The first five steps are the same as the previous example: (1) the attack infects the first host; (2) the ASGS observes the attack; (3) the ASGS is notified that the connection was an attack; (4) the ASGS creates a signature to represent the attack and send it to the IPD; and (5) future attacks that must pass through the IPD are blocked. However, because the attack is self-propagating, in step (6) the host that was infected in the first attack can now infect the other hosts in the organization, or at least all the hosts it can reach without passing through an updated IPD. It is this threat that led us to the work in this paper.

In Section 3.2 we show that by limiting the roles that each host plays we can use filtering capabilities available on commodity routers to enforce those roles. Using this approach we can achieve a defense in depth strategy that addresses the problem of self-propagating attacks. Figure 12 illustrates how the filter rules in routers combine with our IPD approach to protect the network against self-propagating attacks; the design can also help detect the unknown attack. (1) The unknown attack against an unknown vulnerability infects the first host in the network, and (2) the ASGS observes the traffic. (3) The infected host launches an attack against other systems, but the filter rules in the router blocks the attack. (4) The router can notify the ASGS (not shown in the router rules in this paper), and (5) the ASGS creates a signature and forwards the signature to the IPD. (6) Future attacks from the outside are blocked by the IPD. Thus the combination of the IPD device (with signatures from our ASGS) and the routers with the appropriate filter rules contain the initial attack and prevent future attacks.

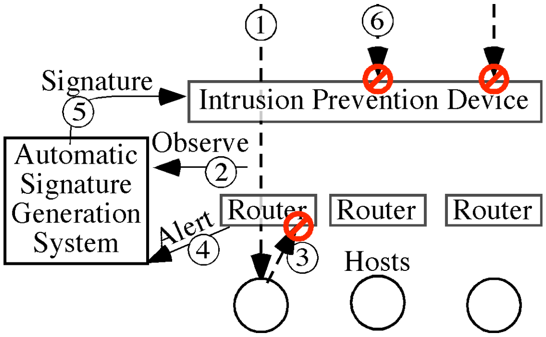

1
6
Signature
5
Intrusion Prevention Device
Automatic Signature Generation System
Observe
2
Router
Router
Router
Alert
4
3
Hosts

Figure 12: Worm-based Attack And Security In-Depth

The focus of our research is the development of the Automatic Signature Generation System. For the intrusion prevention device we plan to use a product from the emerging commercial market (e.g., products by NetScreen, now owned by Juniper), but this design is not effective against self-propagating attacks unless IPDs are distributed throughout the organization in front of every server, a potentially expensive proposition. However, this paper shows how commodity routers with the right filter rules address the limitation of our initial design.