The Effect of Identifying Vulnerabilities and Patching Software on the Utility of Network Intrusion Detection[*]

Richard Lippmann, Seth Webster, Douglas Stetson
Lincoln Laboratory MIT, 244 Wood Street, Lexington, MA 02173-9108
Email: lippmann@ll.mit.edu

Abstract. Vulnerability scanning and installing software patches for known vulnerabilities greatly affects the utility of network-based intrusion detection systems that use signatures to detect system compromises. A detailed timeline analysis of important remote-to-local vulnerabilities demonstrates (1) Vulnerabilities in widely-used server software are discovered infrequently (at most 6 times a year) and (2) Software patches to prevent vulnerabilities from being exploited are available before or simultaneously with signatures. Signature-based intrusion detection systems will thus never detect successful system compromises on small secure sites when patches are installed as soon as they are available. Network intrusion detection systems may detect successful system compromises on large sites where it is impractical to eliminate all known vulnerabilities. On such sites, information from vulnerability scanning can be used to prioritize the large numbers of extraneous alerts caused by failed attacks and normal background traffic. On one class B network with roughly 10 web servers, this approach successfully filtered out 95% of all remote-to-local alerts.

Keywords: intrusion detection, vulnerability, attack, network, signature, false alarm, scan, probe, DoS, worm, exploit, Internet, patch

1 Introduction

Intrusion detection systems are not used in isolation. They are almost always used in conjunction with defensive mechanisms including frequent installation of software updates or patches to eliminate known vulnerabilities, firewalls, and vulnerability scanning to find known vulnerabilities on protected machines. In recent years, the cost and difficulty of using these defenses has decreased dramatically and they have become more widespread. Low-cost personal firewalls are available to protect individual hosts. Scanners that catalog vulnerabilities in large networks have become more capable and comprehensive with almost daily updates in rules used to detect vulnerabilities. In addition, software patches have become easier to install and many modern

---

operating systems now include mechanisms to automatically apply security patches and other updates. The purpose of this paper is to assess the impact these defensive techniques have on the utility of network intrusion detection systems.

This paper focuses on network-based intrusion detection systems that detect known attacks using separate signatures for each attack and on the use of these systems to detect successful attacks where remote attackers compromise monitored machines (remote-to-local attacks). It was motivated by anecdotal evidence suggesting that these important attack types are rarely detected by network-based intrusion detection systems. On well-protected sites, network-based intrusion detection systems detect scans, some denial of service (DoS) attacks, and failed remote-to-local attacks, but rarely do they detect successful remote-to-local system compromises. This occurs even though there are many signatures for such attacks, there are 100's to 1000's of unsuccessful remote-to-local attacks per day, and analysts spend hours each day examining alerts.

This paper has two goals. The first is to analyze how defensive techniques interact with network intrusion detection to determine when these systems will detect successful remote-to-local attacks. The second is to demonstrate that knowledge of the defensive posture of a site can enhance the utility of network intrusion detection by reducing the number of alerts that analysts must examine to detect system compromises.

The remainder of this paper first reviews important characteristics of the most common type of network-based intrusion detection system. It also describes the use of firewalls to protect internal machines and to isolate, in a Demilitarized Zone (DMZ), systems that offer network services to the public. Following this, the importance of detecting and responding to remote-to-local attacks is discussed in relationship to the utility of detecting and responding to other types of attacks that are commonly found by network intrusion detection systems, including DoS and reconnaissance or scan attacks.

The concepts of "windows of vulnerability" and "windows of visibility" are then introduced. These represent time intervals when protected hosts can be compromised by exploiting recently discovered vulnerabilities and when these compromises will be detected by network intrusion detection systems. A detailed timeline analysis follows for eight important vulnerabilities. Each timeline indicates when vulnerabilities were announced, when software patches, intrusion detection signatures, and vulnerability scanner rules were available, and when vulnerabilities enabled major Internet worm incidents. The implications of these timelines for well-protected hosts in a DMZ where no hosts have known vulnerabilities are then discussed and statistics on the frequency of discovery of serious vulnerabilities are presented. These show it is feasible to update server software whenever serious vulnerabilities are discovered for a small number of hosts. The implications of these analyses for poorly protected sites and for normal sites with many protected hosts are then discussed. The paper then focuses on an analysis of intrusion detection alerts at one site to demonstrate that knowledge of known vulnerabilities for protected hosts can be used to prioritize alerts and focus on those that may represent successful systems compromises. This is followed by a summary and discussion of further issues.

2 Network-Based Intrusion Detection Systems

Network-based intrusion detection systems that use signatures to detect known attacks are pervasive because many hosts can be monitored by one intrusion detection system and no changes are required on monitored hosts. Descriptions and evaluations of research, commercial, and open-software network-based systems are available in [13,15,18,25]. These systems rely on signatures or rules to detect known attacks by comparing the contents of captured packets to signatures or rules. For example, to determine if an attacker is attempting to access a backdoor file left behind by the code-red worm [2], the string "scripts/root.exe" could be searched for in incoming traffic on TCP port 80. Signature-based systems normally do not detect new attacks. In some instances a new attack may be detected using an old signature [13,15], but this appears to be infrequent. It usually occurs only for attacks such as buffer overflows that exploit similar vulnerabilities. A new attack is typically not detected unless a new signature is developed for that attack and this signature is downloaded and installed in the intrusion detection system. Network-based intrusion detection systems also often do not distinguish between successful attacks and failed attempted attacks. For example, for the above code-red backdoor signature, the signature may create an alert whenever an external user attempts to execute the backdoor "root.exe" file whether or not the executable exists or whether it runs successfully.

The evaluations of intrusion detection systems referenced above noted many practical limitations of these systems including failing to match signatures when the network traffic load is too high, producing many false alarms, and being vulnerable to the insertion and evasion attacks described in [20]. The initial analysis in this paper assumes an ideal intrusion detection system without these practical limitations. It will be assumed that a network-based intrusion detection system issues an alert whenever an attacker attempts to exploit a known vulnerability, that there are no alerts for attacks that exploit novel new vulnerabilities, that there are no false alarms, and that after a new attack signature is installed, all instances of that attack are detected.

The initial analysis of this paper also focuses on secure networks where internal machines are behind a well-configured firewall, where only a small number of machines in a DMZ provide public networked services, and where a network intrusion detection system monitors traffic between the protected network and the Internet. This configuration is common at many government, educational, and commercial sites. It also influences system administrators to be concerned most about security for the small number of vulnerable machines in the DMZ, instead of the many more machines located behind a firewall.

Table 1. Three categories of remote attacks.

Attack Type	Description
Remote-to-Local	Remote attacker obtains privileges on a protected local machine that are normally reserved for local users and inaccessible to remote users
DoS	Deny access to a network service
Scan	Obtain information about hosts and network services

3 The Importance of Successful Remote-to-Local Attacks

The three categories of attacks shown in Table 1 are generally considered the most important types that can be detected using network-based intrusion detection. Remote-to-local or privilege-gaining attacks permit remote users to obtain privileges they are normally denied. The most serious remote-to-local attacks provide attackers with root-level privileges on Unix hosts and Administrator privileges on Microsoft Windows hosts. Denial of service (DoS) attacks deny access to a network service. Episodic DoS attacks send a small number of packets that induce a software or protocol fault while continuous DoS attacks consume a resource by sending a continuous stream of packets. Scan or reconnaissance attacks provide an attacker with information about hosts and network services. Other important attack categories including "abuse of Internet privileges" and "poor security practices" (e.g. connecting to restricted web sites, downloading restricted material, using telnet instead of ssh) are not included in Table 1 because these behaviors are site-specific and can be detected using traffic monitoring.

Table 2 shows the importance of these three attack types by considering the potential damage, the most common local site-specific response, the response cost, and the effect of the response on other, future attackers, who launch an identical attack against the same victim. This table applies only to known old attacks where software patches are available to prevent exploitation of the known vulnerability. The first row of this table analyzes successful remote-to-local and episodic DoS remote-to-local attacks. Episodic DoS attacks are included in this row because they have similar characteristics to remote-to-local attacks.

This paper focuses on detection of successful remote-to-local attacks in the first row of Table 2 because these are the most damaging and have enabled recent worldwide Internet security incidents including many worms and DDoS attacks [2,3,6,7,8,12,23]. As indicated, detecting these attacks as they occur allows system administrators to react by shutting down and cleaning up the compromised systems and protecting against future attacks by installing software upgrades and patches. Such rapid action can prevent theft of corporate data, monetary loss from theft of financial information, and much longer down times caused by further compromises launched from the initial compromised machine. The dollar amount of losses due to remote-to-local attacks reported in [19] is roughly 5 to 35 times greater than that of DoS attacks

Table 2. The effectiveness of detecting and responding to known attacks.

Attack	Attack Damage	Immediate Local Response	Response Cost	Effect on Future Attackers
Successful Remote-to-Local and Episodic DoS	High	Shutdown, Analyze, Cleanup, Patch	High	Block This Attack Type
Failed Remote-to-Local and Episodic DoS (Probe)	None	Record/Block Source	Low	None
Scan	None	Record/Block Source	Low	None
Continuous DoS	Mod-High	Block Attacker(s)	Moderate	None

depending on whether the cost of "Theft of proprietary information" is added to the cost of "System penetration by outsider" when assessing the cost of remote-to-local attacks. While these attacks can cause the most severe damage, they are also the easiest to prevent from reoccurring. Intrusion detection systems can detect the initial remote connection for these attacks or post-compromise actions including communications from the attacker, attempts to compromise other hosts, scanning from the compromised host, and packet streams sent from compromised hosts participating in DoS attacks. Detecting the initial remote connection simplifies the response and lessens the damage.

The next two rows in Table 2 contain failed remote-to-local and failed episodic DoS attacks (often called probes) and scan or reconnaissance attacks. Both types of attacks provide an attacker with information about remote hosts and network services but do no damage to the victim. Responses are also often ineffective in preventing future scans or probes. These attacks are ubiquitous and are enabled by many automated tools. Our experience on three separate class B address spaces, is that it is common to observe more than 40 separate scans per day from as many different sources with from hundreds to thousands of packets sent from each source to non-existent IP addresses. The Snort intrusion detection system [21] also often detects thousands of remote-to-local attacks each day. On one well-protected DMZ, careful hand verification of these alerts demonstrated that all were either caused by failed attacks or normal background traffic. Detecting scans and failed remote-to-local attacks provides situation awareness and sometimes indirectly detects previously successful remote-to-local attacks by identifying internal compromised machines that are scanning external hosts. The often-suggested response of blocking packets from external scanning or probing hosts may or may not be effective. It will block a novice attacker who scans and then attacks from the same source address but not more capable attackers who scan from compromised machines. It will also not block an attacker who uses public lists of IP addresses to find target machines or an attacker who falsifies the source IP address for probes. A non-local response not shown in Table 2 is to contact the system administrator of the external scanning machine. This can be ex-

tremely effective when possible, but requires identifying the location of the scanning machine and gaining the cooperation of an unknown administrator. Detecting probes and scans thus provides some situation awareness and protection from simple attacks from the same source, but it does not usually protect against future attacks of the same type from different sources.

Continuous DoS attacks are the third category in Table 2. Distributed DoS (DDoS) attacks where the attacker first compromises many hosts using remote-to-local attacks and installs agents that simultaneously send packets to one victim have become a major concern in the past few years [6]. Although they don't typically damage the target site, they prevent others from completing online purchases and obtaining information and can cause loss of revenue [19].

Intrusion detection systems are useful to detect continuous DoS attacks, however such attacks are relatively easy to detect at the victim by traffic monitoring tools including many tools used for network management. Blocking packets from the attack source(s) stops these attacks. Local blocking at a victim site will stop the current attack but not necessarily prevent future attacks of the same type. More global responses that involve modifying the Internet infrastructure can limit the severity of these types of attacks, but this requires cooperation across many sites. Intrusion detection systems are thus not necessary to detect DoS attacks and stopping an ongoing DoS attack provides little protection against future DoS attacks. Alternatively, detecting the original remote-to-local attacks used to create a group of agents can prevent these attacks.

In summary, remote-to-local attacks are the single most important category of known attacks that a network-based intrusion detection system can detect. They have the potential to inflict the most damage and installing software patches prevents other attackers from exploiting the same vulnerability. Detecting probes and scans provides situation awareness, but these attacks do no damage. Detecting continuous DoS attacks is important, but often possible without intrusion detection systems and does not lead to as great a financial loss as successful host compromises [19].

4 Windows of Vulnerability and Visibility

The ability to detect and respond to successful remote-to-local attacks depends on details concerning when vulnerabilities are discovered and publicized, when software patches or other fixes are made available and installed, and when signatures are made available and installed in intrusion detection systems. Figure 1 shows the primary events that determine if successful remote-to-local attacks will be detected and two of the many possible orderings of events. In both time lines, a new vulnerability is discovered and an exploit is developed that makes use of this vulnerability. After this, a fix or software patch is made available that makes exploits using the new vulnerability impossible and a signature is developed for intrusion detection systems to detect attempted attacks that exploit the vulnerability. It is up to a system administrator to install the published signature and/or the patch and these can be installed in any order or never.
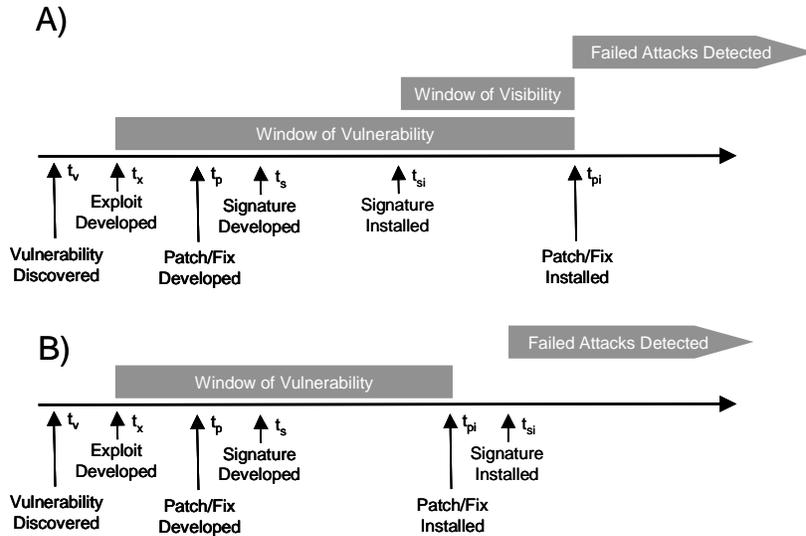
A)

B)

Figure 1. Hosts are vulnerable to new exploits and there always is a window of vulnerability from the time a new exploit is developed until vulnerable systems are patched. There is a window of visibility, as shown in (A), only if a signature is installed in an intrusion detection system before a patch is installed and there is no window of visibility, as shown in (B), if the signature is installed after the patch.

Independent of the order of other events, systems are always vulnerable to attacks from the time an exploit is developed until the time software patches or other fixes are installed. This time interval always exists and it will be called the "window of vulnerability" as in [1]. In addition, there may be a time interval when successful known attacks are detected by intrusion detection systems. This interval, if it exists, will be called the "window of visibility". The upper time line of Figure 1 labeled (A) shows a situation where a new signature is installed before the software patch and there is a window of visibility when new attacks are detected. The lower time line shows a situation when the signature is installed after software patches and there is no window of visibility when successful attacks are detected. There will only be a window of visibility when the time signatures are installed (t_{si}) is before the time patches are installed (t_{pi}). It is difficult to predict if there will be a window of visibility in practice because this depends on how long it takes to develop patches and signatures and on the strategy used by system administrators. There will be a window of visibility if patches are developed after signatures. There will not be a window of visibility if patches are developed before signatures and administrators place a priority on minimizing the window of vulnerability by installing patches before signatures.

Table 3. Eight remote-to-local attacks used for analysis.

Name	CVE Number	Who-Discovered	Publish Date	Who-Patched/Fixed
IIS MDAC RDS Vulner-ability	CVE-1999-1011	G. Gonzalez	7/19/1999	Microsoft
rpc.statd Buffer Overflow	CVE-2000-0666	D. Jacob-witz	7/16/2000	Linux
IIS Unicode Directory Traversal	CVE-2000-884	Anonymous	10/14/2000	Microsoft
BIND TSIG buffer over-flow	CVE-2001-0010	COVERT Labs	1/29/2001	Unix
RPC snmpXdmid Buffer Overflow	CAN-2001-0236	Job de Haas	3/15/2001	Job de Hass, Sun
IIS ISAPI Extension Buffer Overflow	CAN-2001-0500	eEye	6/8/2001	Microsoft
Telnet Buffer Overflow	CAN-2001-0554	TESO	7/18/2001	Unix
Windows XP UPNP Buffer Overflow	CAN-2001-0876	eEye	12/20/2001	Microsoft

5 A Detailed Analysis of Eight Important Remote-to-Local Attacks

A detailed analysis was made of the eight remote-to-local vulnerabilities shown in Table 3 to determine likely sequences for those events shown in Figure 1. The first six of these vulnerabilities were selected from among the 20 most important security incidents cataloged in [22]. The last two are serious recently-discovered vulnerabilities. Information on these attacks is available from the NIST ICAT meta database [14] and in a recent security advisory [10]. Table 3 contains the attack name, the Common Vulnerabilities and Exposures (CVE) number [4], the person or organization that discovered the vulnerability, the date the vulnerability used by the attack was published, and the identity of the organization that published the patch. Dates of attacks shown range from mid 1999 to late 2001, both individuals and security companies discovered vulnerabilities, and the affected software developers provided patches. In this table, the terms "Unix" and "Linux" in the last column indicate that many different software developers of these operating systems developed patches.

Figure 2 shows time lines for all eight attacks. Each separate horizontal region corrresponds to one attack, the horizontal axis shows dates from October 1998 to February 2002, and symbols show the dates of different events. Open squares indicate when vulnerabilities were published and plus signs indicate when fixes or patches were available. These dates were obtained by going back to the original mailing lists or web postings where the vulnerabilities were announced. The patch/fix dates correspond to
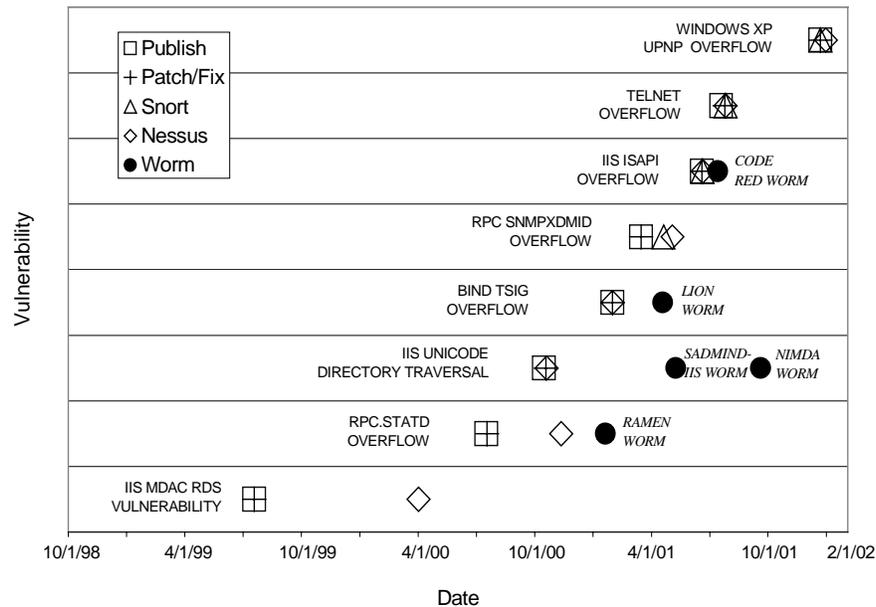
Figure 2. Dates eight important vulnerabilities were publicly released, software patches or fixes were made available, a Snort signature was available, a Nessus vulnerability-testing rule was available, and (if applicable) when major Internet worms exploited the vulnerability.

the dates patches were released for all but the RPC snmpXdmid Buffer Overflow vulnerability. Here, the date is that of the fix suggested by the vulnerability discoverer and Sun released a patch more than five months after the vulnerability was discovered. The availability date for signatures was determined by finding two dates. One was when a signature was available for a widely used open-source intrusion detection system named Snort [21] with performance comparable to that of many commercial systems [18]. The second date was when a security test was available in an open-source vulnerability scanner named Nessus [17] that also performs comparably to commercial vulnerability scanners [9].

The Snort and Nessus dates represent those that might be provided by a user group that carefully tracks publication of new vulnerabilities. Dates were found by searching postings of new security test listings for Nessus and both software repositories and new signature postings for Snort. It was found that Snort signatures were not being updated frequently until roughly April 2001. Times for Snort signatures were thus left out if they occurred before this date while dates for all Nessus security tests are listed. If an attack in Figure 2 was part of a major Internet worm incident, then the date of the

worm release is shown by a solid dot. Information on the worms shown is available in [2,3,7,8,12,23].

Figure 2 shows that patches or software fixes were almost always available soon after vulnerabilities were publicly released. This agrees with the analysis of three older attacks performed in [1]. For many vulnerabilities, the discoverer notified software developers and held back public release until a software patch was available. In others the discoverer presented a fix to prevent the vulnerability from being exploited or software developers rapidly produced patches following the vulnerability announcement. Patches were available on the day the vulnerability was announced or the next day for five of the eight vulnerabilities and within ten days for all vulnerabilities. In almost all cases, the discoverer publicly noted the significance of the vulnerability and this information was widely distributed across many security mailing lists and web sites.

Figure 2 also shows that Snort signatures and Nessus security tests typically were not available until after patches were developed. For five vulnerabilities, Nessus tests were available within roughly one week, but for the others, tests weren't available for from two to nine months. For three vulnerabilities Snort rules were available within a week, but for one, a rule wasn't available for roughly a month.

6 Implications for Well Protected DMZ subnets

If the timelines shown in Figure 2 generalize to other vulnerabilities, then these results suggest why successful remote-to-local attacks are rarely detected at security conscious sites. At such sites, software patches or other fixes are applied to a few critical hosts as soon as they are available. Since patches are always available before or simultaneous with signatures, this implies that there is no window of visibility and only unsuccessful remote-to-local attacks or probes will be detected as shown in Figure 1B. Signature-based network intrusion detection system will never detect successful remote-to-local attacks at such sites. They will simply verify that failed attempts occur.

The strategy of rapidly installing software updates on critical servers would be too costly for widespread adaptation if new vulnerabilities were discovered too frequently. Figure 3 shows that this is not the case. It shows the dates of high severity remote-to-local vulnerabilities over six years as recorded in the NIST ICAT meta-database [14]. These are vulnerabilities that can be exploited remotely and have been rated "severe" by NIST. They include remote-to-local attacks that gain root or administrator privileges, episodic DoS attacks, and other attacks that a security-aware system administrator would want to prevent. These dates are taken from the ICAT database and not from the original vulnerability announcements so they differ slightly from the dates in Figure 2. Vulnerabilities that were discovered within one day are counted as single vulnerabilities because they are typically patched simultaneously.
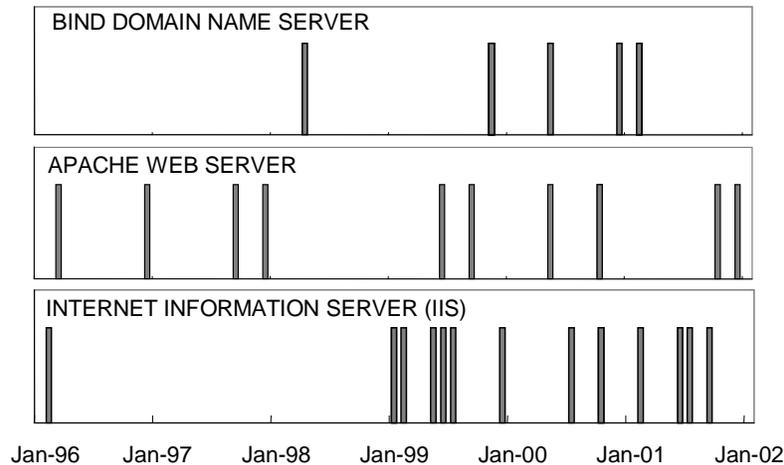
Figure 3. Dates for high severity remote-to-local attacks recorded in the ICAT database for bind DNS server, Apache web server, and Internet Information Server (IIS) Software.

This figure shows the number of severe vulnerabilities for the two most popular web servers (Apache and IIS) that together account for roughly 85% of all web servers [16] and for the BIND software package that is widely used as a domain name server [11]. These three software packages account for half of the vulnerabilities shown in Figure 2. All packages have had 5 or more serious vulnerabilities over six years with at most 15 serious vulnerabilities over six years for any one package, and at most 6 in any one year. Rates of high-severity remote vulnerabilities for other servers (e.g. WU-FTPD and Sendmail) are similar. Figure 3 shows that installing software patches on a small number of machines in a DMZ with limited services (e.g. web, mail, FTP, and DNS servers) requires daily monitoring of security alerts and substantial effort only every few weeks to months. It is thus a practical strategy for securing a small number of machines. A signature-based network intrusion detection system on a well-maintained DMZ provides backup protection, but should never detect successful remote-to-local attacks.

7 Implications for Small Poorly Protected Sites

At poorly protected sites with few hosts, where patches are not installed rapidly, Figure 2 suggests that signatures would have been available to detect attacks that were part of major Internet worm incidents and there would have been a window of visibility where successful attacks were detected, as shown in Figure 1A. Snort signatures or Nessus security tests were always available before vulnerabilities were exploited by worms. They also may have been available before individual attackers exploited vulnerabilities. The widespread nature of many of the worms in Figure 2 suggests that

there are many vulnerable sites. For example, at its peak, the Code Red worm infected more than 359,000 sites [2]. A recent survey of vulnerabilities on web servers [16] found many vulnerable servers, even after many were patched for the Code Red IIS ISAPI vulnerability. This survey found that roughly 10% of servers tested in October 2001 still had back doors left behind by the Code Red II worm. These observations suggest that many small sites are poorly maintained and have little security. Such sites could use network intrusion detection systems to identify vulnerabilities that are successfully exploited and should be patched. This strategy, however, would require continuous monitoring and analysis. In most cases it would be simpler to perform frequent automated scans of hosts using any of the vulnerability scanners discussed in [9] followed by software/security upgrades to prevent found vulnerabilities from being exploited. As discussed above, such upgrades are practical for small sites without too many hosts. For the Nessus scanner, a strategy of automated scanning and patching performed every two weeks would have prevented hosts from being compromised by the Internet worms shown.

8 Implications for Normal Large Sites

Most sites are not as completely protected as the above secure DMZ or as poorly protected as sites used by worms that exploited year-old vulnerabilities. Most of us probably work at sites with known, but recent, vulnerabilities. This is supported by our own vulnerability testing and that of others (e.g. [16]) that finds substantial percentages of vulnerable hosts. It is also supported by a recent survey [19] where 40% of sites responding reported that one or more systems were compromised from a remote attacker over the preceding year. Vulnerabilities exist because it is difficult to eliminate all known vulnerabilities. At sites with 100's to 1000's of hosts, servers, routers, switches, and other equipment that is protected by firewalls, it is practically impossible to maintain software patches on all systems and also enforce firewall filtering and proxy rules required to prevent new vulnerabilities from being exploited. Although software management tools are being developed to simplify this task, they are not in widespread use yet and patches are frequently not installed until they are fully tested because they sometimes do not work or they disable important capabilities. It is also difficult to coordinate security concerns across many system administrators, to keep track of existing hardware, and to make sure that installing new equipment does not create new vulnerabilities. Software upgrades or fixes for known vulnerabilities also can sometimes not be installed because they disable an essential network capability or make software incompatible. There will thus almost always be machines with known vulnerabilities at any large site and intrusion detection systems will issue alerts when these machines are successfully compromised. A major problem, however, is that there are often so many alerts due to failed remote-to-local attacks and normal background traffic that alerts for successful attacks are missed.
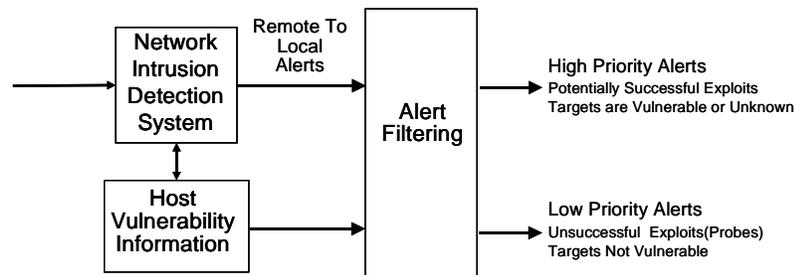
Figure 4. Alert filtering based on vulnerabilities of protected hosts separates remote-to-local alerts into high and low priority bins

9 Using Vulnerability Information to Prioritize Alerts

Our experience is that network intrusion detection systems often produce 100's to 1000's of remote-to-local alerts per day on a class B network. Many of these correspond to failed attacks, some are false alarms caused by normal network traffic, and a very small number are caused by successful attacks. It is essential to find the few successful attacks.

It would be trivial to prioritize attacks if intrusion detection systems reliably indicated whether attacks succeeded or failed. Unfortunately, most don't. It is possible to determine the success of well-known scripted attacks, such as the code-red worm, by detecting actions performed after a successful compromise. It is also sometimes possible to monitor the response from web and other servers for evidence of a compromise. In general, however, it is difficult for a network monitor to determine when attacks succeed. In addition to the issues described in [20], many other characteristics of network monitoring make this difficult. First, some intrusion detection systems analyze only single packets or the packet stream from outside to inside addresses and either do not analyze the response from the attacked machine or do not correlate the response with the incoming request. Second, even if the response is analyzed, sometimes incoming and outgoing packets travel over separate paths and either only incoming or only outgoing packets are visible to the intrusion detection system. Third, many attacks produce no visible immediate response (e.g. episodic DoS attacks or attacks that install backdoors) or communicate back to the attacker using a different transport mechanism than was used to launch the attack. Communications back to the attacker can be sent hours or days after the attack, encrypted or "tunneled" through a common TCP service, and sent back to a different address than was used to launch the attack (e.g. see the HTTP tunnel attack in [13]).

One approach to determine which intrusion detection alerts correspond to successful remote-to-local attacks is to use information concerning known vulnerabilities and hosts to filter alerts into high and low priority bins as shown in Figure 4. This requires site-specific data that can be obtained from vulnerability scanners and also by recording operating system and server software versions to associate known vulnerabilities with as many of the hosts, routers, and other equipment that can be cataloged and analyzed. In addition, it requires network intrusion detection systems to be positioned both on the interface to the Internet and internally behind any firewalls to monitor traffic to and from monitored machines. Monitoring behind firewalls is required because machines behind firewalls are often not updated as frequently as those exposed to network traffic and are thus often vulnerable to many more attacks, because network address translation and load balancing obscure the identity of internal machines, and because many attacks originate from behind firewalls [19].

Figure 4 illustrates how alerts for remote-to-local attacks can be filtered into high and low priority bins. The high-priority bin shown at the top right of this figure contains alerts for vulnerable hosts. These are alerts corresponding to vulnerabilities that are known to exist and alerts for hosts where no information is available concerning a particular vulnerability such as alerts from recently installed hosts. The second low-priority bin shown at the bottom right of Figure 4 contains presumably failed remote-to-local attacks against hosts not vulnerable to detected exploits. To find successful attacks, an analyst would first examine alerts in the high-priority bin and then examine alerts in the low-priority bin. When the time available for analyzing alerts is limited, and all alerts cannot be hand examined, this will result in more detections of successful attacks by focusing on more important alerts. In practice, alerts in the low-priority bin should be sampled (especially previously unseen alert types for existing hosts) because new software releases sometimes inadvertently re-enable old vulnerabilities, software tools that may contain known vulnerabilities are often installed without notifying system administrators, and vulnerability data may be incorrectly recorded or transferred. This approach will be successful if few alerts are left in the high-priority bin and the vulnerability analysis is correct.

An analysis of remote-to-local alerts from the Snort intrusion detection system at one class B site containing roughly 10 Microsoft IIS web servers was performed to determine the percentage of alerts that are left in the high-priority bin with this approach. This analysis suggests that this approach can be extremely effective in reducing the number of alerts that require immediate attention by a system administrator or security analyst. There were roughly 845 alerts per day over roughly two weeks that could have indicated successful remote-to-local attacks. The top 27 types of alerts generated more than 5 alerts per day each and together were responsible for generating roughly 830 remote-to-local alerts a day. Of these 830, roughly 95% could be placed in the lower-priority bin based on knowledge of the software patches, operating systems, services, and software versions running on the web servers. This left only 5% or 42 alerts per day from one alert type in the high priority bin. This alert was a generic signature used to detect web traversals by scanning for "..\" and "../" in web requests. It detected primarily failed attacks or probes for a variety of web traversal attacks and false alarms for relative path addressing used in web pages. A detailed analysis of these alerts involving full-time intrusion detection analysts and site system

administrators indicated that the low-priority alerts were all failed attacks or false alarms due to normal traffic.

At this site, this approach successfully reduced the number of high-priority remote-to-local alerts by roughly a factor of twenty (from roughly 800 to 40 per day). It also didn't require extensive information concerning software on protected hosts. Table 4 shows the information required to assign a low priority to the most common alerts. Information was required concerning the host operating system, the existence of web server extensions, and the installation of specific IIS patches.

It was not simple to determine specific questions that could be used to prioritize alerts. First, alert types corresponding to potential remote-to-local attacks had to be identified. Then, a detailed analysis of alerts and associated vulnerabilities was required to identify software components necessary for the success of attacks. In many cases, this was made overly complex by poor descriptions of alerts that required a detailed analysis of alert signatures themselves, poor cross referencing from alerts to information describing vulnerabilities, and non-standardized and distributing documentation of vulnerabilities. This analysis was made somewhat easier by the use of the CVE vulnerability numbering system [4] to associate alerts to vulnerabilities. Some alerts that could not be assigned to a low priority were false alarms caused by normal background traffic. Categorizing these alerts required a detailed analysis of signatures and background traffic. Ideally, an analysis to identify background traffic that can cause false alarms and also to specify conditions that must be met for an alert to stay at a high priority could be performed by intrusion detection developers and users. The result would be a shared list of conditions that must be met for an exploit that triggers an alert to succeed and also a list of normal traffic that might cause false alarms.

Table 4. Reasons for assigning a low priority to the most frequent remote-to-local alerts. Generic "http directory traversal" alerts could not be assigned a low priority.

Alert Name	Ave Alerts per Day	If Low Priority, Why
Web cgi redirect	140	Not running cold fusion
Sun RPC high port access	110	No Sun Servers
front page shtml.exe	85	No FrontPage
iis-vti_inf	67	IIS Patched
front page shtml.dll	63	No Frontpage
http directory traversal	54	
Shaft to client	30	No UNIX Servers
DDoS Mstream client to handler	25	No UNIX Servers
ISAPI Overflow ida	23	IIS Patched

The analysis presented in Figure 3 also requires accurate and timely information concerning IP addresses for hosts exposed to the Internet and knowledge of their operating systems, software, and patches. Vulnerability and network scanners simplify the task of gathering this information for large numbers of workstations. As Figure 2 shows, rules for at least one scanner are now being rapidly updated to detect new vulnerabilities. Scanners are being used at many sites to gather information on operat-

ing system and software versions as well as information on large numbers of known vulnerabilities. One limitation of scanners is that it is difficult to monitor all hosts and discover new hosts continuously. This limitation can partially be overcome by continuously performing passive analysis of traffic within network-based intrusion detection systems. The double arrow between intrusion detection and host vulnerability components in Figure 4 indicates this type of analysis. Passive analysis can be used to detect new hosts recently connected to the Internet that should be analyzed for vulnerabilities. It also can potentially identify the operating systems of monitored hosts. For example, it might be possible to determine the operating system type of protected hosts using a combination of "passive operating system fingerprinting" of TCP packet headers as described in [24], analysis of banners produced by server software, and analysis of the services offered by hosts. Although this approach has not been carefully explored, a study reported in [5] used packet header information to identify operating systems and filter remote-to-local alerts. Experiments on a few networks demonstrated that often roughly 30% of the remote-to-local alerts could be sent to the low-priority bin by simple filtering based on operating system type. This approach should only be used to detect recently connected hosts and perform a preliminary analysis of those hosts. The analysis should preferably be extended and confirmed by active scanning and consultation with system administrators.

10 Summary and Discussion

It is has recently become easier to scan for known vulnerabilities in hosts and to obtain software patches designed to eliminate these vulnerabilities. This capability has had two seemingly contradictory effects on the usefulness of network intrusion detection systems. On small sites, such as small DMZ networks, network intrusion detection systems might never detect successful systems compromises because vulnerabilities can be patched before intrusion detection systems have been updated to detect associated attacks. On small sites, vulnerability scanning and patch installation delegates network intrusion detection into a backup role. On large sites, it is too expensive to install patches and network intrusion detection systems may detect system compromises. These important detections, however, are often hidden among thousands of unimportant alerts caused by failed attacks and normal background traffic. On such sites, information on vulnerabilities and protected hosts can be used to prioritize alerts and focus on those that might represent successful exploitation of known vulnerabilities. On large sites, vulnerability scanning and information on protected hosts can thus make network intrusion detection useful and practical.

The analyses that led to these conclusions focused on dangerous remote-to-local attacks where a remote attacker achieves restricted privileges on protected hosts and compromises those hosts. It also focused on the common approach to network intrusion detection where signatures are used to detect known attacks. The first part of this paper explored the time sequence for availability of software patches, intrusion detection signatures, and vulnerability scanner rules following announcements of new vulnerabilities. It was found that software patches are almost always available before new

intrusion detection signatures. "Windows of visibility" where intrusion detection systems detect successful system compromises will never occur if software patches are installed as soon as they are available. In addition, it was found that vulnerability-scanning rules are available soon after new vulnerabilities are announced and, so far, they have been available well before a vulnerability was used as part of a widespread worm attack. Timelines from eight recent important vulnerabilities support these sequences of events. Signature release times were for the Snort network intrusion detection system and vulnerability scanner rule times were for the Nessus security scanner. It is expected that this order of events will continue and that intrusion detection signatures and rules to scan for vulnerabilities will be distributed simultaneous with the public disclosure of new vulnerabilities or only slightly after. This makes it possible to protect against exploits that use new vulnerabilities by scanning a site to detect vulnerable hosts and installing software patches whenever a new vulnerability is announced. Such a strategy would minimize the "window of vulnerability" where a host is susceptible to exploitation using a new known vulnerability. As noted above, it would also mean that there would never be a "window of visibility" where a network intrusion detection system using signatures detects successful remote-to-local attacks. One limitation of this analysis is that few attacks were examined and non-malicious security groups or individuals discovered all attacks. Even if vulnerabilities were discovered and exploited by a malicious group, the analysis suggests that software developers would develop and release patches soon after these "in-the-wild" exploits were discovered and that intrusion detection signatures would be provided simultaneously with patches. This work also only analyzed signature and rule development dates for one network intrusion detection system and one vulnerability scanner. Since highest priority is normally placed on developing patches, it is likely that similar results would be obtained for other intrusion detection systems and scanners.

The strategy of installing patches as soon as they are available was demonstrated to be practical for small numbers of hosts in DMZ's because the discovery rate for major vulnerabilities of Internet server software packages was shown to range from zero to seven per year. With three or four server software packages in a DMZ, serious security software patches may be required at most a few times a month. On such well-protected sites, network intrusion detection systems should never detect successful remote-to-local attacks. They could, however, serve as a backup and provide some "defense in depth." They can verify that patches are installed correctly by detecting failed remote-to-local attacks (probes), detect DoS attacks, and provide situation awareness by detecting scans.

Intrusion detection systems are more useful on large sites with known vulnerabilities. It is often necessary to use hosts with known vulnerabilities because it may be too costly or time consuming to update all the required software, because using vulnerable software may be the only solution to satisfy operational or compatibility requirements, or because no centralized authority is available to scan and enforce a security policy across all vulnerable hosts. On sites containing systems with known vulnerabilities, network intrusion detection systems can detect successful system compromises, but this requires constant monitoring and labor-intensive analysis of thousands of alerts per day caused by failed attacks and normal background traffic. An analysis of alerts from the Snort intrusion detection demonstrated that knowledge of the operating sys-

tem, software packages, and vulnerabilities of monitored hosts provides information that can be used to prioritize these alerts. Of roughly 830 remote-to-local alerts that occurred each day, 95% corresponded to vulnerabilities that did not exist on a site with roughly 10 web servers and could be categorized as low priority alerts. This left only 5% or roughly 40 high priority alerts per day that could have been caused by successful system compromises. Prioritization didn't require knowing all details such as patch levels of all software running on monitored hosts. It often required only being able to prove that the host isn't susceptible to a vulnerability by knowing that is is not using a specific operating system (e.g. it is not UNIX) or that it is not running a particular server extension (e.g. it is not running FrontPage). Useful prioritization does not necessarily require maintenance of a large database including detailed patch level information. It requires lists of the identities of monitored hosts, of their operating systems, and of server software. Even knowledge of operating systems and major server types can help prioritize alerts.

In summary, vulnerability scanning and applying software patches should always be a component of site security. On small sites, such as small DMZ networks, these tactics can make network intrusion detection systems serve a secondary role because vulnerabilities can be patched before intrusion detection systems have new signatures required to detect attacks. On large sites, it is too expensive to patch all systems and network intrusion detection systems can serve a useful function by detecting successful compromises. On such sites, information on vulnerabilities and protected hosts is required to prioritize alerts and focus on those that might represent successful exploitation of known vulnerabilities.

11 Limitations and Future Work

Further analyses can be performed to explore the interrelationships between vulnerability scanning, software patching, gathering information on protected hosts and intrusion detection. This paper was limited to known attacks, primarily to remote-to-local attacks, and to network intrusion detection systems that use signatures to detect known attacks. Conclusions do not apply to network intrusion detection systems that detect novel new attacks without signatures. Such systems could provide an early warning function by guiding forensic analysis of new attacks that could lead to new patches and signatures to block and detect future exploitation of newly detected vulnerabilities. As noted above, further remote-to-local attacks could be analyzed and timeline information could be obtained for other intrusion detection systems and vulnerability scanners. Such new results are unlikely to change the major conclusions except when patches are so difficult to develop that they are distributed after signatures are available.

Some of our current work focuses on passive analysis of network traffic to determine host information required to filter alerts. Passive analysis can provide a list of protected hosts and alert analysts to newly installed unauthorized hosts that may be running software with known vulnerabilities. It might also be able to provide informa-

tion concerning operating systems and server software by examining software banners and packet headers.

Acknowledgements

References

1. Arbaugh, W.A., W.L. Fithen, and J. McHugh, Windows of Vulnerability: A Case Study Analysis, IEEE Computer, 2000. 33,(12), 52-59, http://www.cs.umd.edu/~waa/pubs/Windows_of_Vulnerability.pdf.
2. CAIDA, Code-Red Worms: A Global Threat, Cooperative Association for Internet Data Analysis (CAIDA), 28 November 2001, http://www.caida.org/analysis/security/code-red/.
3. Chien, E., W32.Nimda.A@mm Worm, Symantec Corporation, 18 September 2001, http://www.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html.
4. CVE, Common Vulnerabilities and Exposures, The MITRE Corporation, 2002, http://www.cve.mitre.org/.
5. Dayioglu, B. and A. Ozgit, Use of Passive Network Mapping to Enhance Signature Quality of Misuse Network Intrusion Detection Systems, in Proceedings of the Sixteenth International Symposium on Computer and Information Sciences, 2001, http://www.dayioglu.net/publications/iscis2001.pdf.
6. Dittrich, D., Distributed Denial of Service (DDoS) Attacks/tools, University of Washington, Seattle, 2001, http://staff.washington.edu/dittrich/misc/ddos/.
7. Dougherty, C., S. Hernan, J. Havrilla, J. Carpenter, A. Manion, I. Finlay, and J. Shaffer, CERT Advisory CA-2001-11 sadmind/IIS Worm, CERT Coordination Center, 8 May 2001, http://www.cert.org/advisories/CA-2001-11.html.
8. Fearnow, M. and W. Stearns, Lion Worm, SANS Institute, 29 March 2001, http://www.incidents.org/react/lion.php.
9. Forristal, J. and G. Shipley, Vulnerability Assessment Scanners, Network Computing, 8 January 2001, http://www.networkcomputing.com/1201/1201f1b1.html.
10. Hassell, R., R. Permeh, and M. Maiffret, UPNP - Multiple Remote Windows XP/ME/98 Vulnerabilities, eEye Digital Security, 20 December 2001, http://www.eeye.com/html/Research/Advisories/AD20011220.html.
11. Internet Software Consortium, ISC Berkeley Internet Name Domain (BIND) Domain Name System (DNS), January 2002, http://www.isc.org/products/BIND/.
12. Lestat, M., The Ramen Worm and its use of rpc.statd, wu-ftpd and LPRng Vulnerabilities in Red Hat Linux, SANS Institute, 7 February 2001, http://rr.sans.org/malicious/ramen.php.
13. Lippmann, R.P., J.W. Haines, D.J. Fried, J. Korba, and K. Das, The 1999 DARPA off-line intrusion detection evaluation. Computer Networks, 2000. 32: pp. 579-595.
14. Mell, P. and T. Grance, The ICAT Metabase CVE Vulnerability Search Engine, National Institute of Standards and Technology, January 2002, http://icat.nist.gov.

15. Mueller, P. and G. Shipley, To Catch a Thief, Network Computing, 2001,
    http://www.networkcomputing.com/1217/1217f1.html.
16. Netcraft Web Server Survey, Netcraft Ltd., Bath England, October 2001,
    http://www.netcraft.com/survey/index-200110.html.
17. Nessus, The Nessus Security Scanner, 2002, http://www.nessus.org.
18. NSS Group, Intrusion Detection Systems Group Test (Edition 2), Oakwood House, Wen-
    nington, Cambridgeshire, England, December 2001, http://www.nss.co.uk/ids/.
19. Power, R., 2001 CSI/FBI Computer Crime and Security Survey, Computer Security Insti-
    tute, Spring 2000, http://www.gocsi.com/forms/fbi/pdf.html.
20. Ptacek, T.H. and T.N. Newsham, Insertion, Evasion, and Denial of Service: Eluding Net-
    work Intrusion Detection. Secure Networks, Inc., 1998,
    http://secinf.net/info/ids/idspaper/idspaper.html.
21. Roesch, M. Snort - Lightweight Intrusion Detection for Networks, in USENIX 13th Sys-
    tems Administration Conference - LISA '99. Seattle, Washington, 1999,
    http://www.snort.org.
22. SANS, The Twenty Most Critical Internet Security Vulnerabilities (Updated). Bethesda,
    MD, System Administration, Networking, and Security (SANS) Institute, 2001,
    http://www.sans.org/top20.htm.
23. SANS, NIMDA Worm/Virus Report – Final, System Administration, Networking, and
    Security (SANS) Institute, October 2001, http://www.incidents.org/react/nimda.pdf.
24. Spitzner, L., Know Your Enemy: Passive Fingerprinting, Honeynet Project, January 2002,
    http://project.honeynet.org/papers/finger/.
25. Yocom, B., K. Brown, and D.V. DerVeer, Review: Intrusion-Detection Products Grow
    Up, Network World Fusion, 2001, http://www.nwfusion.com/reviews/2001/1008rev.html.