

Old Threats Never Die

Why Protection for Old Vulnerabilities can never be Retired

Published: IBM ISS 2007

With year-on-year increases in vulnerabilities, malware and new threat vectors, businesses must deal with an expanding barrage of attacks. As threats mount, businesses place greater pressure on alerting and protection technologies designed to identify and block threats before they cause damage. Under pressure, protection performance and defense robustness can visibly weaken.

In the physical world, when pressures mount, civil engineers consider their options. As the water level rises behind a concrete dam, engineers select a course of action—whether it is the dam with additional material, making the dam taller or opening the spillways in order to reduce mounting pressure. In many cases, the time, energy and expense of building physical improvements factors heavily in determining a course of action.

In contrast, the world of IT security software engineering includes few barriers to constructing a “stronger” or “taller” dam to stop the rising tide of Internet threats. The relative ease of adding new attack signatures and new protection algorithms means that intrusion detection and prevention systems are becoming stronger all the time. Which is fortunate, since few businesses would ever contemplate the equivalent of opening the spillways when under attack.

Unfortunately, many organizations have mistakenly opened spillways in their IT security defenses, allowing entire classes of attack to penetrate the network. The misguided decision to allow certain malicious traffic stems from underestimating the duration of a particular threat, or investing in protection technologies unable to cope with mounting pressure— technologies now rendered obsolete in the face of advanced threats.

Businesses must understand basic aspects of the lifecycle of Internet threats in order to apply the proper security strategy. In particular, organizations need to be aware that old threats never actually retire from the digital landscape. Rather, they tend to become background noise on the Internet— ready to burst into life with each new software update, host recovery, device deployment or embedded system release.

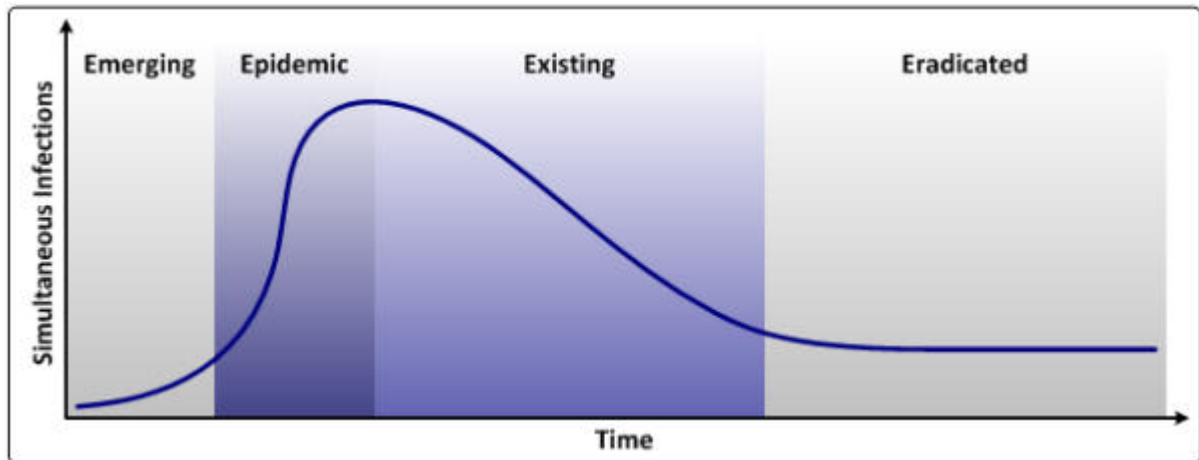
Threat Lifecycle

Internet threats come in many shapes and sizes, but in generally the largest and most visible threats are those that exploit vulnerabilities and weaknesses over a

network. With literally thousands of unique software vulnerabilities meeting these criteria each year, it is important to understand the lifecycle of a typical threat.

Malware Threat Lifecycle

'Malware' is the most frequently encountered Internet threat, and consequently, the most widely studied. Worms—the most insidious form of malware—are self-propagating and network-centric, and typically evolve through four sequential phases; as observed in the following graph.



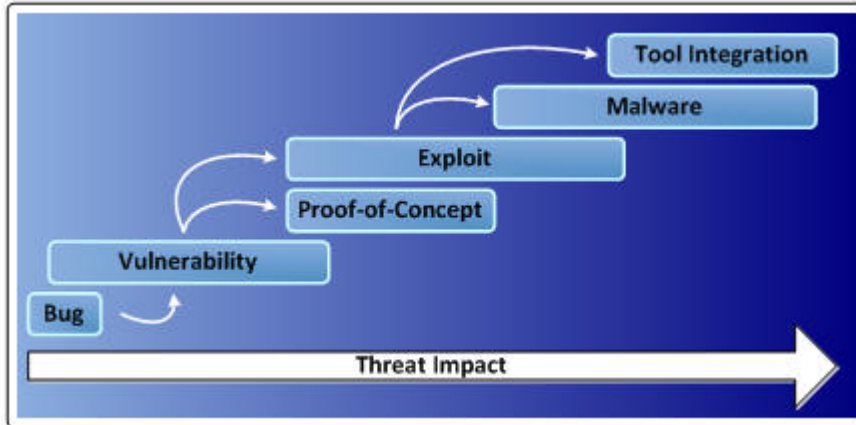
The phases an Internet worm goes through include:

- Emerging – the malware begins to surface as a potential threat.
- Epidemic – there is either a sharp increase in infections, or if there is a slow continuous increase of infections over time.
- Existing – protection is developed and gradually applied; the malware is removed, and infection rates decrease. Eradicated – the malware ceases to be a 'practical' threat.

Although a malware threat may be “eradicated,” it does not mean that all infections have been removed or that the threat ceases to exist. Instead, it assumes that detection, protection and removal technologies are aware of the particular piece of malware and sufficiently deployed to prevent the threat from ever again exhibiting the same epidemic infection rates.

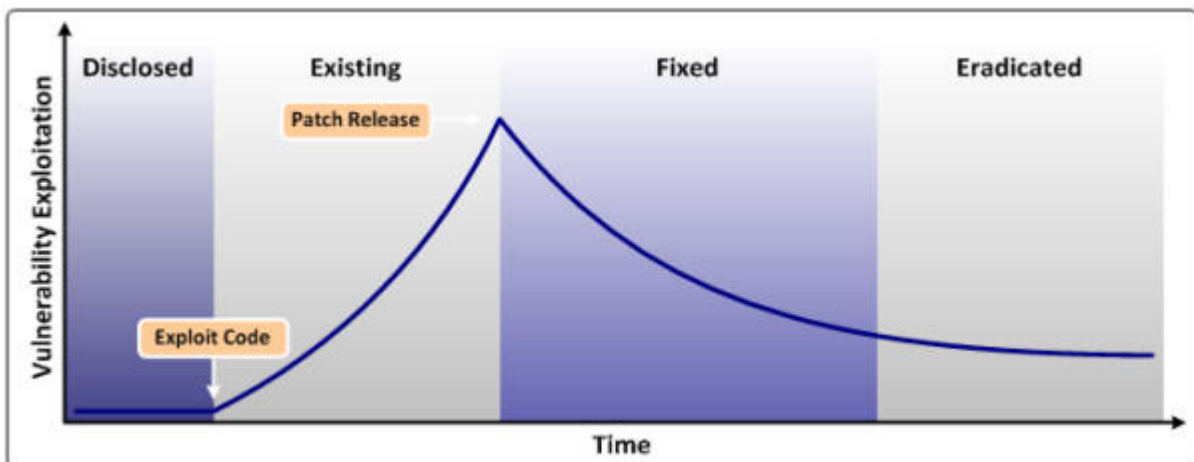
Vulnerability Lifecycle

Many forms of malware exploit vulnerabilities as part of their initial installation process. However, the lifecycle of a vulnerability is distinct from that class of threat – with malware only being a subset of vulnerability threats.



In the graph above, the bug is the precursor to a vulnerability. Bugs are not typically considered a threat at this stage. If the bug can be reproduced reliably and leveraged in a security context (e.g. for bypassing authentication, to cause a memory stack overflow, to allow access to ‘restricted’ content, etc.) it will subsequently be classified as a security vulnerability. By itself, a vulnerability represents a marginal threat. Real threats materialize only after proof-of-concept or related exploit code is available. Once exploit code is generally available the threat escalates, and its rapid integration into malware or security assessment and penetration tools sees the threat reach maximum potential.

Since vulnerabilities are almost always associated with a particular software flaw, they are usually remediated via a software patch (or update). The patching process affects the vulnerability lifecycle significantly (see below).



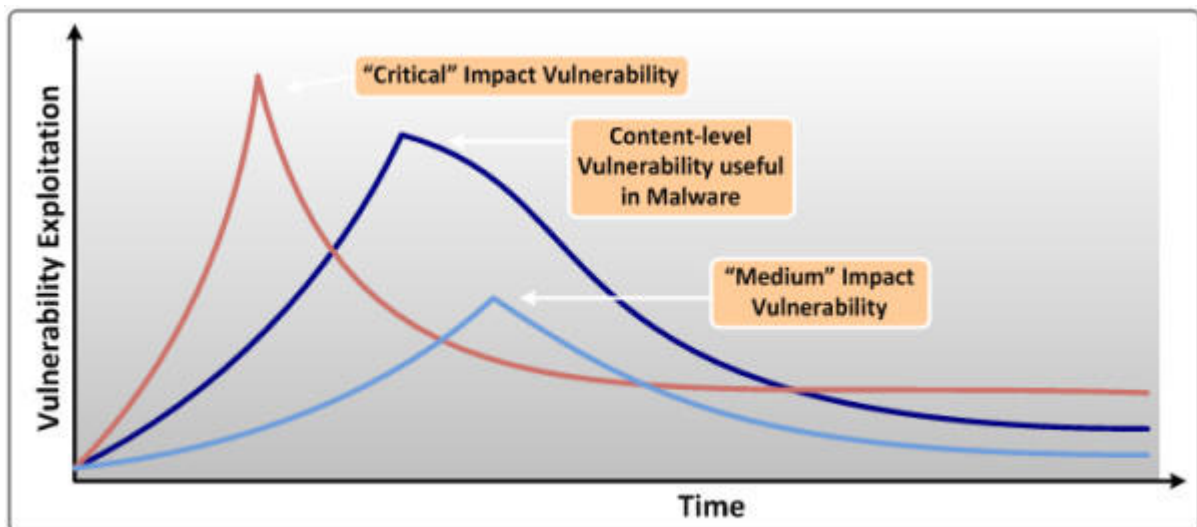
The phases of the vulnerability lifecycle include:

- Disclosed – the vulnerability is disclosed, but not exploited because exploit code is not commonly available.
- Existing – proof-of-concept or exploit code appears and there is an increase in the rate of exploitation.

- Fixed – vendor patches are developed and gradually applied; vulnerable systems are patched and exploitation rates decrease.
- Eradicated – the vulnerability and its corresponding exploits cease to be a practical threat.

Several notable differences between the vulnerability and malware lifecycle include:

- In the Disclosed phase, vulnerability exploitation is nil or minimal (i.e. the vulnerability may have been disclosed, but there is no public guidance for exploiting it). For malware, the Emerging phase implies that exploitation techniques already exist. In the case of vulnerabilities, exploit code availability signals the Existing phase of the lifecycle. For zero-day threats (vulnerabilities that make their public *début* in the form of exploit code only) the Disclosed' phase will be non-existent.
- The shape of the vulnerability lifecycle curve is pinched (i.e. 'saw-toothed') — contrasting with the gradual upswing (i.e. 'whale-fin') of malware. The peak in the vulnerability lifecycle becomes more pronounced depending on the popularity or widespread deployment of the vulnerable software, and exploitability of the vulnerability (i.e. it can be reliably exploited, it doesn't require authentication, it provides system access, etc.).
- The apex of the 'saw-tooth' typically occurs immediately prior to the vendors software patch or fixes availability. The Eradicated phase in the vulnerability lifecycle ties more closely to the duration of time in which patches have been available, whereas malware's Eradicated phases refers to the infection rate. For vulnerabilities, the period between patch availability and the Eradicated phase is typically measured in days or weeks.

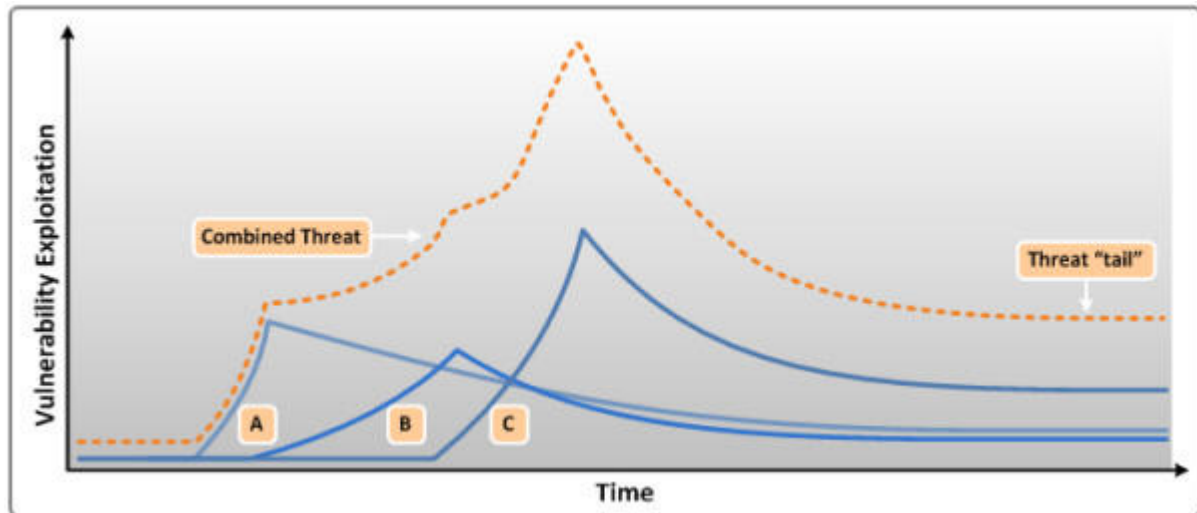


Vulnerabilities that prove highly useful in malware installation or propagation will exhibit a less pinched lifecycle curve around the patch release date. Vulnerabilities that are disclosed simultaneously with security patch or update releases and are still useful to malware authors will have a similar curve to the malware lifecycle.

Lifecycle Tails

Even though both vulnerability and malware lifecycles include an Eradicated phase, the threats never disappear completely—meaning the volume of new infections and exploitation never reduces to the level observed during the initial phase.

The end of the vulnerability lifecycle—or the “tail” in the graph below—has implications for security teams and protection technologies.



The graph depicts three independent vulnerability lifecycles (A, B and C)—each with its own peaks and asymmetry occurring at different points in time. The peaks indicate exploitation and the “tail” refers to the line sloping downward from the peak. By combining all three vulnerability lifecycles into a cumulative dotted line, it becomes clear that the ongoing exploitation that occurs in the Eradicated phase may affect more hosts than the peak exploitation of any subsequent single vulnerability.

Vulnerability Retirement

Since the year 2000, IBM Internet Security Systems (ISS) has observed an exponential rise in the discovery and disclosure of new vulnerabilities. In 2006 more than 600 new vulnerabilities were disclosed per month on average, with the monthly average expected to rise each year. Very few of these vulnerabilities ever warrant a high degree of media interest, however. In fact, only the most critical vulnerabilities in the most widely deployed software ever reach the limelight—accounting for approximately four or five new vulnerabilities making media headlines each month.

Once patches are available and the media headlines die down, businesses lose visibility of the threat that the vulnerability (and its exploitation) represents. After a short period of time, many businesses assume that the threat has not only been mitigated, but that it has been exterminated—never again to pose a risk to the organization or the Internet in general.

Unfortunately, time and experience have proven the opposite. The tail of the vulnerability lifecycle represents a continued threat and the continued exploitation of insecure hosts over time. Old vulnerabilities (and their exploits) never disappear or retire gracefully; instead they become dangerous background noise on the Internet. The examples that follow will illustrate the disturbing effects vulnerabilities can have over the long term.

Login Bypass Vulnerabilities

Malicious attackers and malware authors prize vulnerabilities that enable them to bypass authentication processes through flawed logic within a networked service. Such vulnerabilities never require the authors to write complicated lines of exploit code. The Sun Solaris telnet “-f” vulnerability (CVE-2007-0882) is a prime example from February 2007.

The vulnerability lay in the telnet daemon (in.telnetd), as implemented in Solaris 10 and 11 (SunOS 5.10 and 5.11), which misinterpreted client “-f” sequences as valid requests to bypass authentication. The application logic flaw can be observed in the following two code snippets (identified by Kingcope and taken from Opensolaris source):

From /usr/src/cmd/cmd-inet/usr.sbin/in.telnetd.c

```
} else /* default, no auth. info available, login does it all
*/ {
    (void) execl(LOGIN_PROGRAM, "login",
                "-p", "-h", host, "-d", slavename,
                getenv("USER"), 0);
}
```

From /usr/src/cmd/login/login.c

```
case 'f':
    /*
     * Must be root to bypass authentication
     * otherwise we exit() as punishment for trying.
     */
    if (getuid() != 0 || geteuid() != 0) {
        audit_error =
ADT_FAIL_VALUE_AUTH_BYPASS;
        login_exit(1); /* sigh */
        /*NOTREACHED*/
    }
    /* save fflag user name for future use */
    SCPYL(user_name, optarg);
    fflag = B_TRUE;
```

The flaw meant that exploitation could occur if an existing user account name was supplied after the “-f” (e.g. bin), and could be conducted through the command line

with:

```
telnet -l -f
```

The Sun Solaris telnet “-f” vulnerability mirrors a series of login bypass vulnerabilities found more than 10 years ago. The first widely published example appeared in May 1994 as a multiple UNIX rlogin authentication bypass (CA-1994-09 and much later as CVE-1999-0113). In the early example, the “-f” flag of the rlogin service allowed the attacker to login as ‘root’ (i.e. system administrator) and was referred to as a “-froot” vulnerability.

“-froot” didn’t work with the vulnerable versions of Solaris in 2007 due to earlier security enhancements that prevent users logging in directly with the root account—hence, “-f” was more likely to work.

The “-f” option can be traced to early versions of the BSD operating system (most likely version 4.4), and authentication bypass vulnerabilities appeared semi-regularly in operating systems that re-implemented or ported the login service to other operating systems (e.g. AIX, Linux, etc.) thereafter.

Vulnerability Conclusion: Old vulnerabilities regularly reemerge in new software releases when older code is ported between operating systems and applications, or through simple programming errors. In the most recent example, earlier versions of Sun Solaris were not vulnerable, it was only later versions that included the “extended functionality” after the complete integration of ktelnet.

Businesses are unlikely to monitor every “minor” update or enhancement to the operating systems and applications they use on a daily basis, which makes reemerging vulnerabilities a significant threat vector.

Web Browser Vulnerabilities

The relative complexity of Web browser technologies has resulted in a constant stream of new vulnerabilities over the last decade. Many Web browser vulnerabilities have been classed as low or medium impact threats, and vendors have made patches available for most of them.

While Web browsers have experienced bugs and vulnerabilities since their creation, the majority of vulnerabilities prior to 2005 were usually considered nuisances and instigated by users browsing dangerous links. The following developments changed this perception:

1. Web browser technology became more important within organizations and constituted a core component in new business applications (especially for Web 2.0 technologies and application platforms that utilize .NET and AJAX).
2. A reliable method of exploiting old Web browser vulnerabilities (typically classed as denial of service prior to 2005) was uncovered through a method referred to as heap spraying.

3. Advances in protection systems that combated classical malware propagation caused malicious authors to seek alternative delivery channels for their malware, resulting in the targeted exploitation of softer Web browser technologies.

Based upon constant scanning and monitoring for malicious Web browser vulnerability exploitation in 2007, IBM ISS found that the three most frequently encountered exploits targeted the following vulnerabilities:

1. Microsoft MS-ITS CHM file code execution (MS04-013)
2. Microsoft RDS.Dataspace ActiveX (MS06-014)
3. Mozilla InstallVersion.compareTo (MSFA2005-50)

None of these represent new vulnerabilities. In fact, the most commonly targeted Web browser vulnerability was originally disclosed and fixed by Microsoft in April 2004. While patches have been available for the vulnerability, many hosts on the Internet remain unpatched and consequently targeted for reliable exploitation.

Vulnerability Conclusion: The cyclical advances in attack and defense mean that attackers are constantly seeking ways to leverage vulnerabilities in new and more reliable ways. As additional attack vectors are uncovered and new exploit techniques developed, vulnerabilities considered medium or low impact before can become critical overnight.

Similarly, as doors close on popular exploit delivery channels, attackers seek the next in the line of low-hanging fruit. Vulnerabilities previously described as limited in scope may be exploited through a combination of other lesser vulnerabilities. Applications that have non standard or infrequent patching processes make increasingly popular targets.

In the case of Web browsers and similar technologies, attackers can easily serve exploit code for hundreds of known vulnerabilities with a single page request until one is found to work. Consequently, old vulnerabilities within Web browsers will continue to be automatically targeted because it costs the attacker little to do so.

Worm Infestations

Both security professionals and businesses greatly fear self-propagating worms capable of independently traversing the Internet and infecting millions of hosts in minutes. Thankfully, worms have been diminishing due to better defense-in-depth security technology deployments (and more profitable routes for developers with the skills necessary to develop devastating worms). But worms have not been eradicated. The SQL Slammer worm—considered by many to be the most damaging worm in history—is a primary example.

The SQL Slammer worm (Slammer for short; but also known as Sapphire, Worm.SQL, W32/SQLSlam-a and Helkern) rapidly infected networks by exploiting a buffer overflow vulnerability in the Resolution Service of Microsoft's SQL Server

2000 and Desktop Engine (MSDE) 2000 software products (see Microsoft MS02-039) in the early morning of January 24th 2003—even though patches for the vulnerabilities had been available since July 2002. SQL Slammer loaded the DLL's Kernel32.dll and WS2_32.dll, calling GetTickCount to create a seed for a random IP address routine, and then proceeded to continuously send 376 bytes of exploit and propagation code across port 1434/UDP until the affected SQL Server process was shut down.

Unlike many worms (both before and after), Slammer's main function was pure propagation. It did not infect or modify files, nor did it incorporate any Distributed Denial of Service (DDoS) or backdoor functionality, and it only existed in memory. It also did not have a preference for scanning local subnets (such as the functionality found in the Nimda worm) which meant that its propagation across local networks was not as quick as it could have been. Even so, Slammer spread rapidly, with reports of 75,000 infections within the first 10 minutes. While the infection could be removed by rebooting the infected host, the scanning method did generate enormous amounts of traffic that quickly overwhelmed networks. Without protection in place, vulnerable servers were quickly re-infected.

Four-and-a-half years later, attacks from Slammer-infected hosts targeting port 1434/UDP are constantly observed by IBM Managed Security Services (MSS) and are still the most frequently encountered attack on the Internet.

Vulnerability Conclusion: History has repeatedly proven that malware can explode upon the scene many months after fixes for the vulnerabilities they exploit have been commonly available. Even if no attacks have been observed and the vulnerability has faded from an administrator's memory, attacks or new vectors for attack can surface a year or more later. With increases in enterprise connectivity and bandwidth, the explosion of malware across the Internet can be now be measured in seconds.

Malware outbreaks like the SQL Slammer worm reveal that many organizations do not know precisely which software is deployed within their environments. Selective patching strategies focusing only on hosts known to have vulnerable software installed may leave large swathes of an enterprise network unprotected.

Protection Retirements

Businesses expect their security technologies to protect them against each and every new threat with the same efficiency and level of performance as the day the solution was first purchased and installed. However, with so many new threats uncovered on a monthly basis, protection technologies are under increasing strain.

IBM ISS alone catalogued 7,247 newly disclosed vulnerabilities in 2006, and has catalogued more than 32,000 vulnerabilities since 1995. IBM ISS also identified and analyzed more than 300,000 new and unique malware captures in 2006 alone. With

such high numbers, it is no surprise that some protection technologies are struggling to keep pace.

New threats are not the only thing protection technology must combat. Older threats still plague the Internet and must also be prevented—whether they are old login bypass vulnerabilities, Web browser exploits, worm infestations or new vectors for previously misclassified vulnerabilities. Though many of these threats have been in the Eradicated phase of the vulnerability lifecycle for many years, they continue to threaten business.

Legacy Protection Technologies

Many of the protection technologies regularly employed by organizations to combat today's threats are based on decades-old technologies. For example, signature-based anti-virus relies on two older technologies—unique-hash lookups and regular-expression string matching.

Why would such technologies only now start to suffer from the escalating volume of threats? When these technologies first became popular, the threat landscape was relatively small and unsophisticated. In the early 1990s, the number and variety of unique malware lay in the 10^3 realm. By 2000 this figure had reached the 10^5 realm, and by 2006 more malware was uncovered in a single year than during the first 20 years of malware history.

Signature-based protection engines are very fast and efficient at identifying small lists of known threats. But with each new signature added there is a minute decrease in performance due to system overheads (whether storage capacity, memory occupation or CPU clock cycles). Consequently, as new threat volumes have increased by 50-100 percent per annum, the prospect of 50-100 percent decreases in protection performance is not acceptable to most organizations.

Signature-based protection engines have been incrementally improved over time, but the escalation in system overheads and the drain on performance will continue if comprehensive threat prevention is to be maintained. If an organization is prepared to remove old signatures at the rate new signatures are added, performance of the protection engine will remain static.

Many protection vendors have decided to silently depreciate signatures for older or infrequently encountered threats—either making them optional or removing them entirely from their products.

Protection Advice

As evidenced by the emergence of old and previously forgotten vulnerabilities, the depreciation of old protection can be a recipe for disaster. In addition, the cumulative tail of each vulnerability lifecycle has necessitated the development of newer, more efficient technologies capable of handling entire classes of threat whose volume will continue to grow at high rates.

For organizations to successfully combat the threats they will encounter on a daily basis, they must maintain protection for not only tomorrow's threats—but for every threat ever identified. The selective application of protection for older threats cannot be recommended.

Old vulnerabilities may reappear and present a threat to an organization at anytime as the following scenarios will show:

- An old system is restored from backup after a data failure and is missing current security patches.
- A vendor adds new functionality to a popular software application by incorporating code from other packages and fails to identify classic application logic flaws.
- A group of vulnerabilities had been classed as denial of service because they cannot be exploited in a reliable fashion. After a few years a clever researcher identifies a new exploit vector that suddenly makes all of these vulnerabilities trivially exploitable.
- A software package encapsulates or repurposes components from a third-party. At some point, the third-party releases security fixes for the component. However, because the authors of the main software package are unaware of the update, they fail to provide their customers with the necessary fixes.
- An attacker can automatically cycle through application vulnerabilities without being noticed in very little time. Consequently, it requires no effort on behalf of the attacker to try every vulnerability known to potentially affect the application—regardless of age.
- Embedded and certified systems may contain older operating systems or applications for simplicity and stability reasons. The systems are either very difficult to update (for example, the operating system on a multi-function printer) or are frequently forgotten.

Conclusions

The vulnerability lifecycle should affect how organizations manage the security of their networked infrastructure. The Eradicated phase of the lifecycle does not mean that vulnerabilities are gone forever or dormant. Instead, the tail of the vulnerability lifecycle represents a constantly circulating threat to vulnerable systems. Once a vulnerability reaches the Eradicated phase, it simply becomes part of the background noise of Internet traffic.

The exponential increase in vulnerabilities and the threats that flow from them have resulted in a combined threat that is much larger than any new threats that may arise. As time goes by, it will become increasingly difficult for organizations to differentiate one attack from another if they depreciate older protection systems.

If an organization is struggling with system performance from legacy signature-

based protection systems, they should evaluate newer technologies that provide scalable protection against entire classes of threat. For example, vulnerabilities such as cross-site scripting (XSS), directory transversal and SQL injection account for a sizable percentage of software vulnerabilities. However, the way an attacker is likely to exploit them now and in the future is predictable. Instead of a one-for-one signature protection model, more advanced heuristic engines can be used to protect the entire threat class. The same engines will continue to perform whether 100 or 100,000 new vulnerabilities are discovered.

Protection for old vulnerabilities should never be depreciated or removed. Instead, organizations should seek more efficient protection engines that can prevent entire classes of threat