



The McAfee Gateway Anti-Malware Engine

Protecting Users from Emerging Malware Threats

Table of Contents

Executive Summary	3
Major Malware Trends from 2007 to 2008	4
Financially motivated attacks	4
Compromise of legitimate websites	6
Multimedia software exploitation	7
Preemptive Malware Defense	8
Exact detection	9
Generic detection	9
Heuristic detection	10
Behavioral heuristics	11
Structural heuristics	12
Conclusion	14
Glossary	14

Executive Summary

Malware is a sophisticated business. Unlike the good old days when attacks were perpetrated by individual malcontents, today's malware developers are organized into sophisticated, disciplined teams. What's more, they are constantly probing and refining their attacks to inflict maximum damage or reap the highest profitability—or both. This white paper, authored by the McAfee® network security business unit's anti-malware team, recaps the latest threats seen throughout 2007 and the first half of 2008, and introduces you to the McAfee gateway anti-malware engine and the technologies it uses to combat today's and tomorrow's threats.

The McAfee gateway anti-malware engine (or McAfee anti-malware engine) is a multi-platform engine incorporated into McAfee's web gateway products, such as McAfee Web Gateway (formerly Webwasher). The engine detects and blocks malware threats—everything from viruses and worms to adware, spyware, and riskware. It is designed especially for deployment at the network perimeter. While anti-virus products running on desktop computers can use on-access scanning to effectively detect threats in unknown document or archive formats, the McAfee anti-malware engine inspects exotic formats, such as WordML and Office Open XML, at the gateway—before they enter the network. To further protect end users against emerging malware threats, zero-day threats, and targeted attacks, the McAfee anti-malware engine focuses on generic and heuristic detection of malware. Patent-pending technologies enable the McAfee anti-malware engine to spot malicious behavior, preemptively detect new variants of known malware families, and enhance detection overall by combining capabilities with the McAfee TrustedSource™ web database.



As a founding member and leading supporter of the Anti-Malware Testing Standards Organization (AMTSO), McAfee is committed to fair and independent anti-malware testing. The McAfee anti-malware engine is tested regularly by the independent AV-Test GmbH test labs, one of the most widely respected virus testing organizations in the world. Furthermore, our McAfee anti-malware engine is a vital contributing technology to the respected VirusTotal.com site—a free web service that was cited as one of the 100 best “Security Website” products by PC World Magazine in 2007.

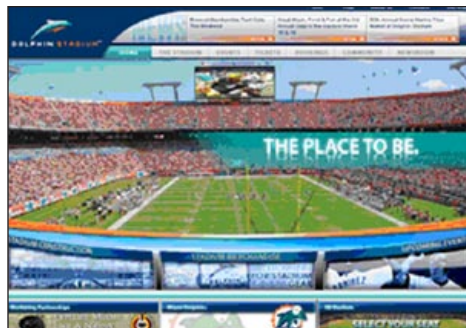
A four-year track record of high-profile, zero-day threats being caught preemptively underscores the McAfee anti-malware engine's exceptional preemptive detection capabilities. Customers using our anti-malware engine have been protected against many zero-day threats right from the start, including high-profile remote code execution vulnerabilities, such as the HTML Elements Vulnerability (MS04-040) of Microsoft Internet Explorer, Microsoft Outlook and Microsoft Outlook Express, which allowed remote code execution by passing specially crafted data to FRAME or IFRAME elements; the DHTML Method Call Memory Corruption zero-day vulnerability (MS06-013) in Internet Explorer, which allowed remote execution of arbitrary code through the createTextRange() method; the DirectAnimation ActiveX Controls Memory Corruption vulnerability (MS06-067), which allowed attackers to remotely execute arbitrary code; the Windows Animated Cursor Remote Code Execution vulnerability (MS07-017), which was heavily exploited in the wild; and many others.

“Professionalization” of this criminal activity has led to modularization and reuse of malicious code components, as well as more silent and stable exploitation. In the past, malicious sites simply fired their whole arsenal of exploits against all visitors. Today, however, they are more likely to probe the client for vulnerabilities and then only run the specific exploits that are most likely to succeed. This increases their rate of infection and reduces the likelihood of detection. Malicious sites also remember IP addresses of clients to which exploits have been delivered and deliver either benign or empty content on later visits. Security companies that research malicious sites or botnets may subsequently suffer denial-of-service attacks as a scornful thank-you.

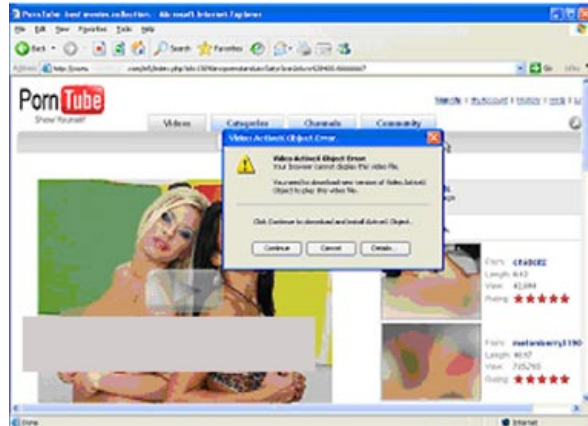


Playing on the Internet is anything but risk free. In March 2008, the Crysis game suffered a format-string vulnerability in its online multiplayer mode.

In 2007, financially motivated attacks even made their way into the virtual worlds of Second Life, World of Warcraft, and others, as virtual achievements can be traded for real dollars on the web. This is a high-profit opportunity for malware authors since the market for online gaming is expanding quickly, especially in China, where its growth is parallel to broadband Internet access availability. The PSW. OnlineGames family of password-stealing spyware targets online role-playing games and was among the most prevalent malware of the year. Software vulnerabilities didn't spare the games industry either. In September 2007, an identity-theft vulnerability in the Second Life client was disclosed that allows malware authors to steal user credentials through the client's proprietary secondlife:// URL scheme. Three months later, Second Life, which uses the QuickTime player to play video sequences within the game, was found to be susceptible to the QuickTime Real-Time Streaming Protocol (RTSP) vulnerability, which was actively exploited in the wild at that time (dubbed Exploit.RTSP-CodeExec.gen by the McAfee anti-malware engine).



The Dolphin Stadium website – not exactly “the place to be” in February 2007.



A fake video portal, with Zlob exploring the missing video codec social engineering trick.

Continuing their prowl for highly ranked websites, in March 2008, Zlob-affiliated attackers applied their misguided creativity to exploit a cross-site scripting vulnerability in the search functionality of popular websites, such as News.com, Wired, and ZDNet Asia, and injected malicious IFRAMEs into the search results. As these results were indexed by search engines, subsequent queries on Google would present links to the legitimate domains, but when following any such link, the cached search result with a malicious IFRAME was executed.⁴ Other attackers take a similar infiltration approach by using malicious Adobe Flash-based ad banners to penetrate ad banner networks, distribute the malicious offering onto legitimate websites that show the advertisement, and then mislead the users to the actual malware. SWF.AdHijack turned out to be the most notorious new malware family in this latest group of indirect attacks.⁵

Multimedia software exploitation

Microsoft's Security Development Lifecycle (SDL) process, and new security methods such as Data Execution Prevention (DEP), User Account Control (UAC), and Address Space Layout Randomization (ASLR), have led to a decrease in the number and severity of vulnerabilities affecting the Microsoft Windows operating system, Microsoft Internet Explorer, and Microsoft Office applications.

Attackers are always probing for vulnerabilities and, as you would expect, they have an affinity for the weakest link. As a result, they are moving on to riper targets, namely popular third-party software, such as multimedia players and platforms.⁶ Microsoft Office document formats had been the primary targets for vulnerability exploitation in 2006 and the first half of 2007, followed by Microsoft's legacy graphics formats WMF, ANI, and ICO. However, exploits based on non-Microsoft formats such as Apple QuickTime, are growing in prevalence now.

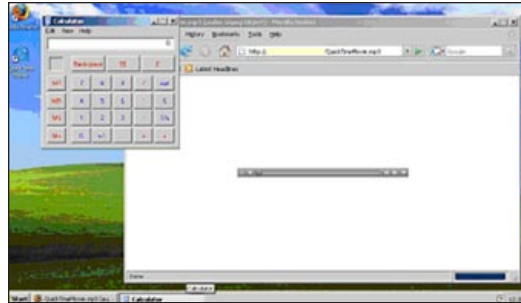
For example, in September 2007, a privilege escalation zero-day vulnerability surfaced that affected users with both Apple QuickTime and Firefox installed.⁷ An attacker would only need to place a specially crafted QuickTime link file on a disguised website and lead victims there. Just one month later, another zero-day vulnerability (CVE-2007-5601) in the RealPlayer media player was exploited in targeted attacks in the wild. The Metasploit Framework, released in January 2008, consistently showed an alarming growth in the number of exploits targeting multimedia applications such as RealPlayer, QuickTime, and iTunes.

4 Dancho Danchev: "ZDNet Asia and TorrentReactor IFRAME-ed," March 2008, <http://ddanchev.blogspot.com/2008/03/zdnet-asia-and-torrentreactor-iframe-ed.html>.

5 Dennis Elser, Micha Pekrul: "Inside Rogue Flash Ads," January 2008, Virus Bulletin Magazine. http://www.trustedsource.org/download/research_publications/SCJan08.pdf.

6 Christoph Alme, Dennis Elser: "Blow Up Your Video," December 2007, Virus Bulletin Magazine. http://www.trustedsource.org/download/research_publications/SCDec07.pdf.

7 Secure Computing Corporation: "Apple QuickTime Player Zero-Day Vulnerability," September 2007, <http://www.trustedsource.org/TS?do=threats&subdo=blog&id=4>.



Simple access to a .mp3 URL results in the Windows calculator popping up in the foreground, with even more “magic” taking place in the background.

In addition, vulnerabilities in legacy graphics formats are still prevalent. For example, a classic stack overflow vulnerability in Windows’ GDI component (MS08-21), fixed by Microsoft in April 2008, was actively exploited in the wild just a few days after the patch release.⁸ The exploit would download a “Poison Ivy” backdoor variant from a server hosted in Korea that in turn placed the victim’s PC under the remote control of the attackers.

Preemptive Malware Defense

The McAfee anti-malware engine combines multiple detection approaches to offer preemptive malware defense. This section outlines the underlying technologies and philosophy behind them.

```
push {ebp+ip$start}
push offset aMailFrom$ : "MAIL FROM:<id>@ic'n"
push {ebp+h$len}
call w$printA
add esp, 0Ch
push {ebp+h$len}
call int$lenA
push 0
push eax
push {ebp+h$len}
push ebx
call $end
push 0Fh
push 400h
push {ebp+var_4}
push ebx
call sub_401A00
test eax, eax
jc loc_402436
push {ebp+var_4}
call sub_4020FD
cmp eax, 201552h
jnz loc_402436
push offset aRcptTo$ : "RCPT TO:<id>@ic'n"
push {ebp+h$len}
call w$printA
add esp, 0Ch
push {ebp+h$len}
call int$lenA
push 0
push eax
push {ebp+h$len}
push ebx
call $end
```

The Bagle, a worm, starting to propagate and mail addresses harvested from a user’s address books.

⁸ Secure Computing Corporation: “Latest GDI vulnerability now exploited in the wild,” April 2008, <http://www.trustedsource.org/TS?do=threats&subdo=blog&id=33>.

Exact detection

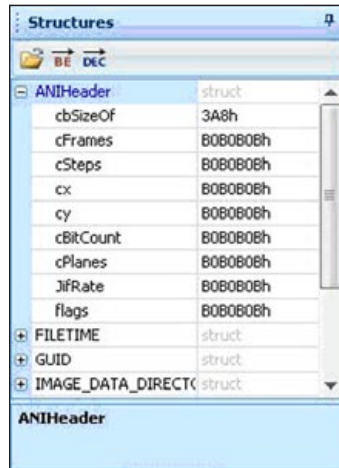
In contrast to the important generic and heuristic detection techniques to be explained later on, which are largely based on experience and assumptions rather than proof, exact detection has the benefit of blocking malware because it knows it's malware. Usually based on signature scanning, exact detection and naming of malware results in the best possible accuracy, performance, and user confidence. False positives are kept to an extreme minimum, usually below 0.005 percent, thereby giving network administrators almost no whitelist maintenance work.

Paradoxically, the analysis required to create such sound detection, resulting in the accuracy described above, is also this method's Achilles heel. With targeted and short-span distributed malware on the rise, samples may not appear on an anti-malware vendor's radar at all, as they have slipped through and left the scene long before manual analysis is done and signature updates are shipped. This can be partly countered with automation and fast response times. At McAfee, threat information—such as samples of new potential malware or malicious sites—is gathered from customer feedback, as part of a global, real-time McAfee TrustedSource network, and from anti-virus partners. These samples are analyzed and classified, signatures are created, and, upon successful testing, shipped with the next database update. All of this happens automatically, 24 hours a day, seven days a week. Analysis of submitted malware samples is performed by an automated system that investigates behavior of suspected programs or websites in an isolated lab environment. Very much like a neural network, an array of weighted voters computes an overall malware probability and associates the new sample with the closest malware family. Each voter is trained on a certain type of malware behavior and detection.

Generic detection

Generic detection is the "seen one, seen 'em all" method that extracts key characteristics from one or more samples of a malware family or exploits and creates a one-size-fits-all detection rule that catches as many variants as possible. Generic detection rules use techniques that are similar to those used by exact detection. With only a slight increase of false-positive risk, generic detection can provide a significant gain in preemptive protection.

While the first reaction to new malware is usually a fingerprint—or signature-based detection, more generic detection rules should follow quickly to combat any upcoming variants of a new malware family or exploits. In February 2008, attackers were alarmingly fast in taking advantage of a remote code execution vulnerability in the popular Adobe Acrobat Reader that had been fixed by Adobe just days before. Using a built-in, proprietary malware detection language, McAfee's malware researchers were the first to extend automatically created detection signatures with a more generic detection for the vulnerability quickly upon the first in-the-wild sightings. Further variants exploiting the same vulnerability would all be caught proactively as "Exploit.PDF.ZoneBac.gen"⁹—such as the malware used in targeted attacks against various pro-Tibetan organizations in March 2008.¹⁰



Exploited “anih” header structure opened in File insight.

Sometimes, the quality of generic detection becomes readily apparent months or even years after its implementation. This was the case with the prominent Animated Cursor vulnerability, probably the most exploited vulnerability in 2007. A variant was based on the very same bug that had already led to the Animated Cursor and Icon Format Handling vulnerability (MS05-002) in 2005. Two years later, attackers simply altered their methods slightly and were able to trigger the vulnerability that apparently was still present in Windows’ `user32.dll`.

A generic detection, intended for the original ANI vulnerability in 2005, also protected McAfee Web Gateway (Webwasher) customers proactively against the Windows Animated Cursor Remote Code Execution vulnerability (MS07-017). This vulnerability also affected users that had just invested in the new Windows Vista operating system. Fortunately, Microsoft released a patch for this critical vulnerability ahead of their patch day schedule. As a side note, one of the very first servers that hosted these zero-day ANI exploits was the same Chinese server that was involved in the Dolphins Stadium hack a few weeks before.

Heuristic detection

Where there’s smoke, there’s fire. Heuristics combine a few known facts with experience to make an assumption on the classification at large. In general, this family of detection techniques provides outstanding preemptive detection capabilities against new malware variants and families and even protects against yet-to-be-discovered vulnerabilities and exploits.

Given such outstanding benefits, the slightly increased perceived false-positive rate associated with any heuristics does not spoil the attractive cost/performance ratio. With a moderate false-positive rate ranging between 0.05 and 0.1 percent running the web anti-malware module with maximum heuristics enabled, required maintenance by network administrators is relatively infrequent. However, the gain in proactive security is tremendous. While blocking of a legitimate, user-requested download is usually perceived as a false positive, a careful examination of the detected behavior may justify a different conclusion. Specifically, are anti-disassembling protection, anti-debugging tricks, and rootkit-like file-hiding behaviors suspicious? Sure they are, and one might even call them malicious, too. But when such behaviors are performed by a game’s copy-protection module or a multimedia software’s digital rights management component—both applications themselves being totally benign—do such behaviors thus become benign in general? This is the gray area of perceived false positives.

Whatever the perception, if a download is blocked or a website cleaned erroneously, whitelisting the affected URL is a hassle that lasts a minute or two. Compare that to forensic analysis of an infected computer to determine if any data has been stolen, and computer cleaning that can take hours or days. Depending on the data or identity that may have been stolen, the financial damage per infection ranges from roughly \$50 for a very simple infection up to millions of dollars when corporate officers' PCs are compromised by information-stealing malware.

Behavioral heuristics

The McAfee anti-malware engine employs behavioral heuristics by combining rule-based and weight-based methodologies. This enables the module to estimate the functionality that a program may perform at runtime. However, since mobile code cannot be inspected at runtime on a gateway, such estimates are always probabilistic. To ensure the best possible gateway performance, emulation is kept to an absolute minimum.

Attack vectors usually divide into several stages. Especially on the web, the first stage is a web page trying to exploit one or (usually) more vulnerabilities on the client computer. The McAfee anti-malware engine inspects HTML and script code for obfuscation techniques, potential buffer overflow exploitation, and shellcode injection, as these are common characteristics of this first stage of attack. Buffer overflowing slides can be detected using statistical methods, and the shellcode is usually located nearby.

```

...
jmp    shellcodeEntryPoint
getPayloadAddr:
; Absolute address now in ESI
pop    esi
xor    ecx, ecx
; Length of payload
mov    cx, 259
decoderLoop:
; Backwards XOR decoder, key 97h
xor    byte ptr [esi+ecx-1], 97h
loop   decoderLoop
...
shellcodeEntryPoint:
call   getPayloadAddr
; Encoded payload follows here

```

Decoder loop, taken from an exploit of the WMF "Graphics Rendering Engine" (MS06-001 vulnerability).

When scanning the potential shellcode, certain characteristic behaviors can be detected. One common one is CALL-TO-POP. It is used to determine the location where the shellcode has been written into memory. Another is a decoder loop. As a shellcode's malicious payload is often encoded, a decoder loop will be used right upon execution of the shellcode (See example in the figure above). The actual implementations of decoder loops vary. For example, the encoded payload may be bit-rotated rather than XOR'ed, or JNZ might be used for iteration instead of the LOOP instruction, as in the Ginwui exploit, which was used for an extremely targeted attack against only one specific company. It took advantage of the Microsoft Word Malformed Object Pointer (MS06-027) zero-day vulnerability that was first used by this targeted attack.

Upon successful exploitation, a drive-by infection can be performed by the attacker. That is, the desired malicious executable file—such as a keylogger or other sort of spyware—can be installed on the victim's PC without their knowledge.



Localized mail with an invoice Trojan carrying a fake Adobe Acrobat Reader icon.

An alternative attack vector starts with the use of social engineering tricks that fool users into believing an application is benign, useful, or at least interesting. The first malicious executable to show up in this attack vector usually is a small “jetty” that can be mass mailed to targets without consuming much bandwidth—known as a Trojan downloader. Like the Trojan Horse in Homer’s Iliad, a Trojan downloader pretends to be something appealing—a gift too hard to resist. For example, executables can be disguised to look like something they’re not with their file extension, version information, or icon.

- A .SCR file extension suggests the executable is a screensaver
- The executable’s version information may make the file appear to originate from a trustworthy software vendor, such as Microsoft, or appear to be a security update
- An executable may carry a familiar icon, such as the Adobe Acrobat Reader icon

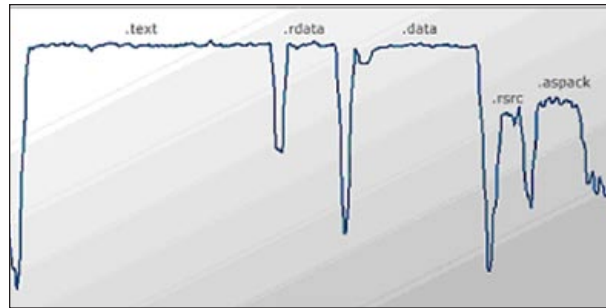
But will they behave like they’re supposed to? Not likely.

The McAfee anti-malware engine’s behavioral heuristics can detect many Trojan-typical behaviors, and block the suspicious code or “jetty” from landing. A behavior of downloading an executable file, storing it locally, and then executing it can be used to heuristically classify an executable as a Trojan downloader. And the behavior that an executable feigns can be compared to the most likely runtime behavior as determined by analysis. If behaviors deviate from the norm, the Trojan can be stopped right at the gateway. If only the good citizens of Troy had been so lucky!

Structural heuristics

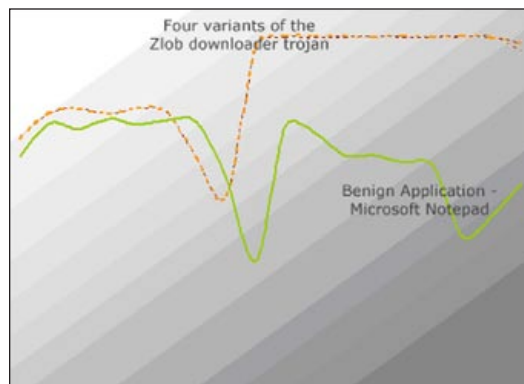
Structural heuristics, also known as geometrical heuristics, analyze the file format, the layout of headers and data, the variation of data, and the use of compression, encryption, or obfuscation. Generic detection of runtime-packed Windows executables falls into this area. Runtime-packers can be used to minimize the download size of an executable, but in most cases they are used to add an obfuscation layer over the actual program code. Upon execution, the executable runs a decoder loop that decodes the packed section and then transfers execution to the unpacked section (in memory).

Sole usage of a runtime-packer usually does not provide enough evidence to classify a program as malicious, since benign applications use such packers as well. Nonetheless, their presence should raise a red flag in any anti-malware engine, and when any other suspicious behavior is encountered, the resulting malicious classification is justified more often than not.



Distribution of information entropy in the NetSky.C worm.

Generic detection of runtime-packers means they are not detected exactly by signature but rather by common characteristics of runtime-packed executable files, such as the information entropy of sections, presence of decoder loops close to the executable's entry-point, or jumps that transfer code execution into writable (data) sections. The chart above, for example, shows the distribution of information entropy along the sections of "Worm.NetSky.C," a variant that is runtime-packed with ASPack. Execution would start in the .aspack section, which contains the decoder loop that unpacks the actual malicious content in the three sections to the left. After unpacking, execution is transferred into the unpacked .text section.

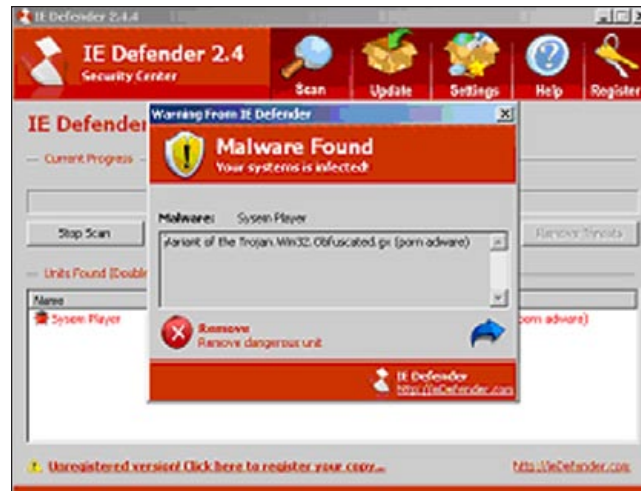


Variants in a malware family tend to be similar, but they differ substantially from benign software.

The ongoing commercialization of malware leads not only to reuse of runtime-packer tools, but also to reuse of code portions or even large parts of a binary, with only small data areas being changed with every variant—such as a download URL or command-and-control server address. The McAfee anti-malware engine's ability to heuristically associate new malware variants with known malware families, specializing in the ad/spyware and Trojan malware types, provides the protection needed to combat this trend. Whenever a new variant appears that reuses code portions from a known ancestor, or acts very similar to a known ancestor, it most likely will be caught preemptively as just another variant of a known threat. This additional preemptive protection is achieved through "fuzzy" descriptions of the malware family.

So the `storm_valentine.exe` that would have been blocked proactively as `LooksLike.XPACK` first, would turn out to be just another Storm worm threat. Other malware families, like Zlob and the various SpySheriff rogue software programs, which are notorious for disgorging new variants of their pests in short intervals, would similarly be blocked. Rogue software pretends a user's computer is infected with malware, while it actually isn't and proposes to clean the PC; users who fall prey to this

scam will not only get more malware rather than being cleaned up, they may even pay for it. When the McAfee anti-malware engine spots a previously unknown descendant of "SpySheriff.JH," it would have already been proactively blocking it as LooksLike.SpySheriff.JH before the next database update, which would exactly identify this latest threat as SpySheriff.JJ.



Rogue software is shown pretending malware was found on a new Windows installation.

Conclusion

Fighting cybercrime is a global, 24/7 battle, and it's only just begun. McAfee Avert® Labs is dedicated to helping businesses and consumers stay one step ahead of attackers, regardless of the diabolical strategies and tools that cyber crooks resort to. By developing technologies such as the McAfee anti-malware engine—and adding innovative capabilities as the team develops them—we will continue to do all we can to help our customers protect their people and property.

Glossary

- | | |
|-----------------|--|
| Buffer overflow | This is a condition that can result when an application accepts data from the user or from the Internet and writes the data into a memory buffer without properly ensuring the data fits into the buffer. Once the unchecked buffer has overflowed, the attacker may be able to overwrite the program's next return address (for stack-based buffer overflows) or linked-list pointers (for heap-based buffer overflows) in such a way that code execution will be transferred to the user-supplied data, thereby allowing remote execution of arbitrary code. |
| False positive | This is the result of an anti-malware engine that blocks a file as malware when in fact the file is benign. Users of desktop anti-virus solutions usually have to restore such files from their quarantine folder, while users of gateway anti-virus solutions have to whitelist the blocked file, or URL, and then download/access it again. The ratio of false-positive detections depends on the detection techniques applied, for example, signature-based detection is known for producing an extremely low amount of false positives, while aggressive heuristic detections lead to relatively more false positives. |

Hidden IFRAME	With the inline frame (“IFRAME”) HTML element, HTML content or script code residing on another server can be embedded into the current web page. User visiting site A unwittingly visit site B if the latter page is embedded in an IFRAME into the first page. Users have no chance to abort this, and when the IFRAME is hidden, they will not even notice the implicit action.
On-access scanning	Desktop anti-virus scanners can hook into a file system and scan a file at the time it is opened. By scanning upon access, files contained in an exotic archive or document format can be analyzed after extraction by the associated application but “just-in-time,” that is, before the user can open the extracted file.
Preemptive detection	This refers to blocking of a new threat by an anti-malware engine without requiring a database update, regardless of whether the malware is entirely new or is a variation of a known malware family or exploit.
Runtime-packer	A runtime packer compresses a given program, then adds the decompression routine in front of the compressed data and generates a new executable out of it. Upon execution, the decompression routine will be run first, unpacking the original program in memory. Afterwards, the original program will be executed.
Shellcode	Shellcode refers to machine code, such as Intel x86 code, that is executed once an attack has successfully exploited a targeted (buffer overflow) vulnerability. The term “shellcode” was coined many years ago when most exploits aimed to open a remote shell on a compromised (Unix) system.
Zero-day threat	This term refers to exploitation of a vulnerability for which no security patch exists from the vendor. PCs affected by the vulnerability, for example, those using the vulnerable client software version, are at risk of being infected simply by visiting a crafted website or opening a document. Zero-day exploits rarely require user interaction.

