WHITE PAPER

No Signature Required: The Power of Emulation in Preventing Malware

TABLE OF CONTENTS

Organizations are facing an expanding and increasingly complex attack surface, and current models of security are failing to keep pace. Emerging threats have surpassed the known, with 70% to 90% of malware unique to a single organization[1] and impossible to detect with signature-based technology. Resource-constrained security teams can be crippled by this—stuck in a constant cycle of firefighting new intrusions to their endpoint systems. A new approach is necessary to move from firefighting to strategic defense, one which eliminates the vast majority of emerging, zero-day threats from the internet before they even have a chance to reach their destination.

KEY CAPABILITIES

- Emulation of Microsoft Windows executables
- Generic unpacking of obfuscated web content
- Behavioral proactive detection of "heap-spray" and Flash exploit attacks
- Proactive detection of PDF and scareware (FakeAV) threats
- Integration with Skyhigh Security Advanced Threat Defense for in-depth malware detection

Emerging Threats Require a New Approach to Protection

Today's web browser environments provide powerful scripting functionality to create feature-rich, user-friendly, and customizable browsing experiences through dynamic web content generation. Unfortunately, this also creates an excellent environment for cybercriminals to create web scripts that, though appearing innocuous, are actually carrying malicious code inside, designed to ultimately infect the endpoint. Malicious JavaScript may be conducting reconnaissance, checking to identify which browsers are installed, 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 endpoint.

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 current security technologies.

Simply evaluating JavaScript and other malicious mobile code for visibly known patterns would not flag these obfuscated scripts as being malicious in their own right. A different approach is required to reveal the true intent of active web content.

A new level of sophistication has also emerged in exploit toolkits, raising the bar for detecting emerging threats. These toolkits cleverly circumvent most security prevention techniques, such as traditional signature-based defenses and malicious code analysis. Security teams need the ability to proactively identify the behavior of exploit toolkits, zero-day threats, and advanced malware to prevent their intended actions.

1. Verizon 2015 Data Breach Investigations Report. (http://www.verizonenterprise.com/verizon-insights-lab/dbir/)

Real-Time Behavior Emulation of Web Content

The Gateway Anti-Malware Engine is the industry's first behavior-based, web content emulation technology that completes all of the following tasks in-line before the code has been delivered and infects an endpoint system:

- Detects zero-day attacks by safely emulating code, not just scanning the script text
- 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 exploits
- Continuously shares zero-day insights with other Skyhigh Security customers through Global Threat Intelligence (GTI) and, in turn, uses GTI-collected data to produce new behavior detections 24/7

In addition to version releases, Gateway Anti-Malware Engine is updated on a regular basis by Skyhigh Security machine-learning systems, which "re-train" the engine's detection logic to improve classification. Updates are delivered to in-production deployments, providing the most up-to-date behavior detection for emerging threats.

The following sections will detail a selection of threat use cases organizations face today, and the methods by which the Gateway Anti-Malware can prevent their success.

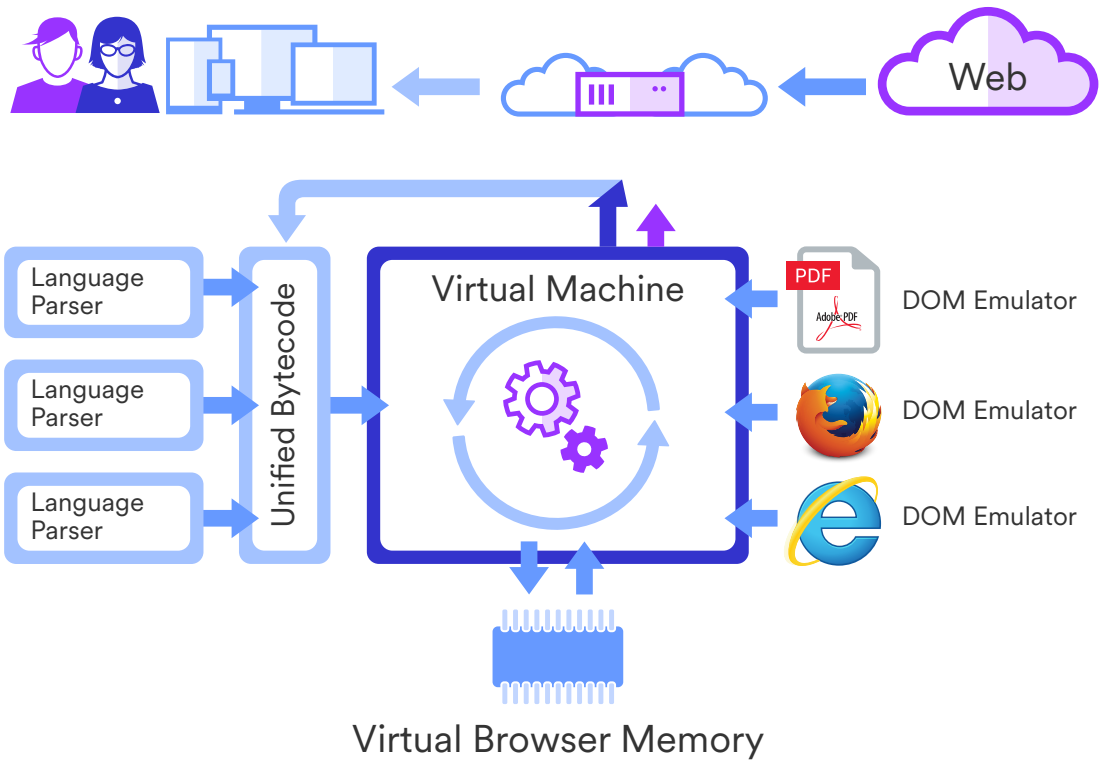

Figure 1. Browser emulation and behavioral profiling help to deliver protection from contemporary web attacks.

Use Case 1: Emulation of Windows Executables

When users download new software packages for their Windows desktop environment, these downloads get inspected by Gateway Anti-Malware. Equipped with powerful Intel CPU emulation technology, Gateway Anti-Malware dissects the download and simulates its most likely run-time execution path—long before it can reach the user's desktop. This helps to protect from new zero-day threats, including spyware, scareware (FakeAV), and ransomware such as CryptoLocker and Locky.

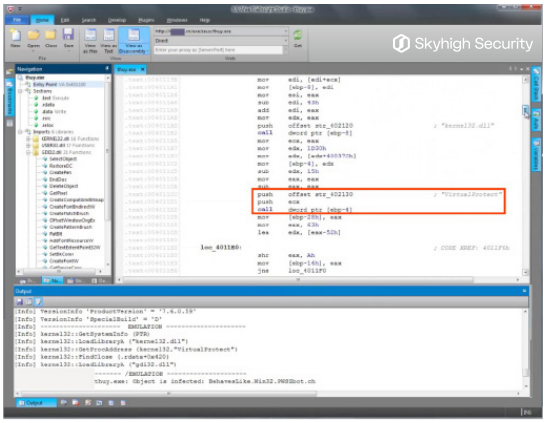

Figure 2. Zeus password-stealer (a.k.a. Zbot) running within Gateway Anti-Malware emulation environment, currently calling "VirtualProtect" Windows system function. (This visualization is based on an engineering front end to Gateway Anti-Malware.)

Use Case 2: Unpacking Obfuscated or Server-Polymorphic Web Threats

Today's exploit toolkits, such as 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 rapidly 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 3). The script accesses the elements by utilizing the getElementsByTagName function of the browser's DOM. A loop then iterates through these nodes, concatenates the hidden script parts, and stores the entire exploit code within a variable.

```
function getShellCode() {
    return
"%u4141%u4141%u8366%ufce4%u
a37e%u205e%uf31b%ua34e%u14
d%u5ea3%u2b08%u1bdd%u61a1%
a324%u2bf0%ua3f5%ua32c%ued
8%u4068%u28d7%u2828%uab78%
2cc4%u2829%ua528%u0c74%uef
f%u5c4a%u6ce5%u2d35%u4c06%
24c3%dd77b%u2c7e%uebab%uc3
2%u3373%u6eee%u1e51%u0732%
}
function sp17() {
    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 < 21)) {
        var fname = "
        var Flash_obj = "<object classid='clsid:d27cdb6e-ae6d-11cf-96b8-44455354
        Flash_obj += "<param na
        Flash_obj += "<param na
        Flash_obj += "<param na
        Flash_obj += "<embed s                    id='swf_id' name='swf_id'";
        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 3. Top obfuscation layer of a Blackhole malware sample—malicious payload is obfuscated and spread over multiple elements.

As the Gateway Anti-Malware Engine 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 4).

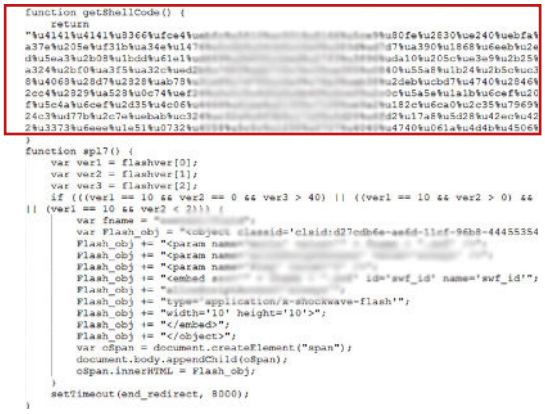

Figure 4. Accurate emulation generically unpacks obfuscation layers and reveals the underlying payload.

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

Use Case 3: Zero-Day Behavioral Detection of Heap-Spray Attacks, Flash Exploits, and More

Emulation also profiles the simulated browser memory to identify suspicious memory manipulation activities, such as heap-spraying buffer overflow. This is a widespread generic technique for exploiting web browser vulnerabilities. Since it is challenging for an attacker to exploit a vulnerability in order to transfer control to the attacker's code, heap spraying improves the chance of successful remote code execution drastically by 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 5 shows a memory area before and after a heap-spray modification.

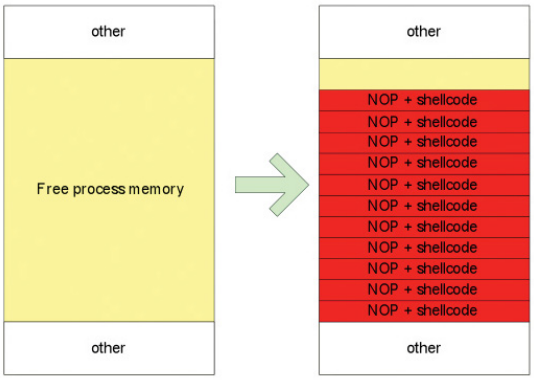

Figure 5. Browser memory sprayed with NOP slides and shellcode.

Behavior-based exploit detection helps to generically identify suspicious memory manipulation or usage within interactive web content, including Flash videos. Shortly after the 'Hacking Team' group was hacked and their zero-day exploits leaked to the public in Summer 2015 (http://krebsonsecurity.com/2015/07/adobe-to-patch-hacking-teams-flash-zero-day/), major exploit toolkits—such as Metasploit, Neutrino, Angler—started to deploy the (Flash) vulnerabilities within their frameworks. A typical exploited Flash video would, for example, contain ActionScript code to download some malicious payload, decode x86 shellcode (obfuscation techniques here include RC4 encryption), and trigger a vulnerability, such as the CVE-2015-5119 Use-after-free vulnerability in order to get the x86 shellcode executed.

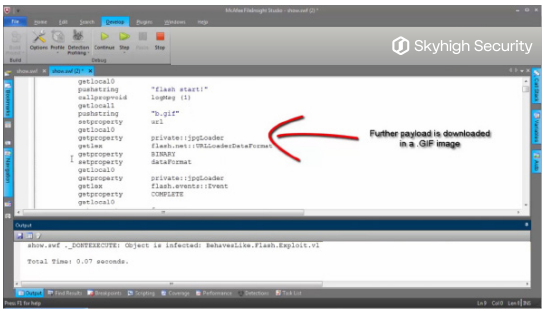

Figure 6. Exploit for CVE-2015-5119 downloading its malicious payload and here masquerading as a .GIF image.

Use Case 4: PDF and Scareware (FakeAV or FakeAlert) Malware Detection

Adobe Reader's features and capabilities are often a challenge to information security professionals 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.

To counter this threat, Gateway Anti-Malware applies its emulation technology to Adobe Reader by simulating Adobe Reader's JavaScript execution and profiling 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-20-22-78-61-22-3b-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-6a-5f-
```

Figure 7. Obfuscated script code hidden in a PDF document.

Figure 7 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 8 shows server-side PHP code running on the toolkit's web server and producing unique PDFs on the fly. The fact that each PDF is unique and therefore has a unique signature makes it difficult, if not impossible, to rely solely on signature-based detection.

Figure 8. 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. Gateway Anti-Malware utilizes both generic unpacking techniques and proactive behavioral emulation to protect against today's web threats.

To improve accuracy, Skyhigh Security 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 9) versus legitimate documents (green dots in Figure 9) is revealed, as expected.

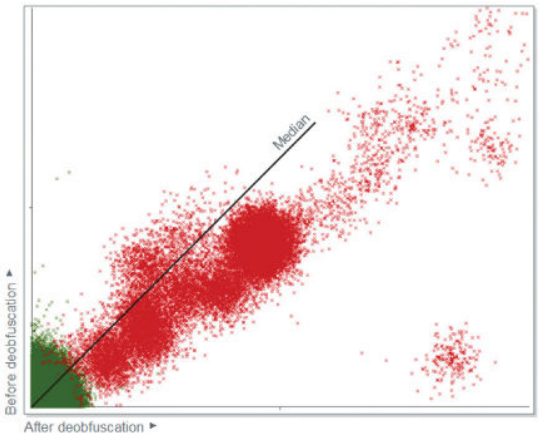

Figure 9. Visual representation of the prevalence of malware hiding malicious fragments under obfuscation.

Similarly, FakeAV scareware threats (Figure 10) are detected 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 the Gateway Anti-Malware Engine to overcome this challenge and provide accurate detection and protection capabilities.

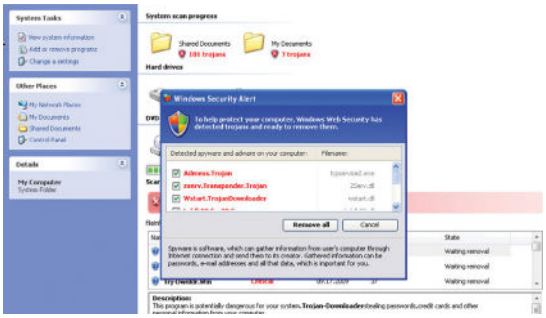

Figure 10. FakeAV and scareware that are difficult for signature-based detection methods are detected and blocked before reaching endpoint system.

Summary

The Gateway Anti-Malware Engine is a powerful in-line technology designed to protect against contemporary threats delivered via HTTP and HTTPS channels, taking web exploit detection, zero-day, and targeted threat prevention to the next level. This technology is featured in the following solutions.

Skyhigh Security Web Protection

The flagship secure web gateway that protects every device, user, and location from sophisticated internet threats. The Gateway Anti-Malware Engine originated in this solution and provides full protection against malicious files and web content. Skyhigh Security Web Protection is the combination of both the on-premises Skyhigh Security Web Gateway and Skyhigh Security Web Gateway Cloud Service.

Learn More

For more information on Gateway Anti-Malware technology, visit: https://skyhighsecurity.com/en-us/solutions/secure-web-cloud.html.

About Skyhigh Security

Skyhigh Security Secure Web Gateway is the cloud-native web security solution that provides the most advanced layered protection from threats and data loss with integrated RBI, CASB, and industry leading DLP capabilities in the web and cloud. It enables organizations to implement a simplified SSE architecture that delivers the security, scalability, and availability that is required for a distributed and remote workforce.

Learn More

For more information visit skyhighsecurity.com.

Skyhigh Security

6220 America Center Drive
San Jose, CA 95002
888.847.8766
skyhighsecurity.com