websense®

USING ANOMALIES IN CRASH REPORTS TO DETECT UNKNOWN THREATS

Written By: Alex Watson and Victor Chin, Websense Security Labs

websense TRITON®

Table of Contents

Overview

One of the biggest challenges organizations face when protecting their intellectual property and other sensitive data is detecting, inspecting, and stopping attacks that are capable of bypassing their cyber defenses.

This white paper examines the “risk indicators” that can be observed in the anomalous activity taking place across a network. This anomalous activity includes behaviors (e.g., application crashes) that might indicate the presence of previously unseen attacks. While such risk indicators are rarely a smoking gun by themselves, they can be augmented with security intelligence and contextual understanding to both uncover previously undiscovered attacks and to help organizations understand the risk associated with those attacks.

Even the most advanced cyber attacks will create anomalies in network and application telemetry that can be used to detect their existence. This white paper shows how such anomalies in application crash reports — specifically, those generated by Windows Error Reporting — can be used as risk indicators to find attacks that have bypassed cyber defenses.

Today’s threat intelligence systems are combining intelligence from multiple disciplines — such as next-generation intrusion prevention system (NGIPS), next-generation firewall (NGFW) and sandboxing solutions — in an attempt to better understand advanced threats. However, most of these systems are based on expertise and signature-based technologies that can contribute valuable intelligence, but cannot detect a new, previously unseen attack. To borrow a phrase from Donald Rumsfeld, former United States Secretary of Defense, these “unknown unknowns” are our biggest concerns.

Now there is a way to reveal the unknown unknowns. Forensic analysis of anomalous behavior, such as what is recorded in crash reports, represents an entirely new approach to detecting previously undetectable attacks that are currently targeting organizations.

How Crashes Happen

Many high-profile organizations fall victim to targeted attacks despite having deployed strong perimeter-based defenses such as firewalls, proxies and anti-virus (AV) products. That’s because attackers — knowing the initial attack vector runs a high risk of detection — prefer to use zero-day exploits that are less likely to be detected by these automated systems.[1]

Many exploits work by forcing an application to perform unexpectedly, and then getting the application to jump to a section in memory containing code that allows the attacker to compromise a computer. However, such exploits (especially recent zero-days using return-oriented programming (ROP)-based techniques to gain control of the call stack) are becoming increasingly difficult for attackers to pull off, largely due to exploit-prevention technologies such as Microsoft’s address space layout randomization (ASLR) and data execution privacy (DEP). When an exploit fails, it often causes the targeted application to crash.

[1] Websense® Security Labs™ research has shown that attackers are especially vulnerable to detection during three key stages of an attack: initial exploit, which is always risky; lateral movement, where attackers feel that there is little chance of being detected; and when establishing persistence within a network.

Once they get past perimeter-based defenses, many attackers relax. They're less concerned about being detected while moving laterally through a network toward their target, because organizations rarely monitor internal network communications due to the expense. Thus emboldened, attackers can forego stealth for speed and take a direct route toward their targets — often using "noisy" exploits that might crash services running on the network.

Websense® Security Labs™ postulated that, if it were possible to distinguish between normal program crashes and applications that crash due to exploit activity and code injection associated with an attack, and then use threat intelligence to gain a more complete understanding of the attack, we could have a way to both retroactively detect when attacks happened, and detect anomalies indicative of new zero-day attacks.

Finding Failed Exploits

Websense researchers have developed a way to use this crash telemetry — specifically, Windows Error Reporting (WER) telemetry — to identify patterns of crashes resulting from zero-day exploits and code injection techniques that are targeting key business applications.

WER (aka "Dr. Watson") is a network-enabled application crash and telemetry reporting technology introduced in Windows XP that Microsoft uses to identify applications and hardware that are causing Windows computers to crash. WER is used on 80 percent of all network-connected PCs — more than one billion endpoints worldwide.[2] Crashes and other system events trigger the sending of detailed information to Microsoft. Microsoft and other vendors use WER to identify and prioritize application bugs and fixes.

WER reports include:

» Specific information about applications, services and hardware, including the specific machine ID operating system on each PC (to the specific update and service pack).

» The BIOS version of the computer.

» All browsers (and their versions, extensions, apps and plug-ins) on these devices.

» A list of every application and their versions installed on the computer.

» Events such as:

» Failed application updates.

» USB device and smartphone insertions.

» TCP Timeouts between computers on the network.

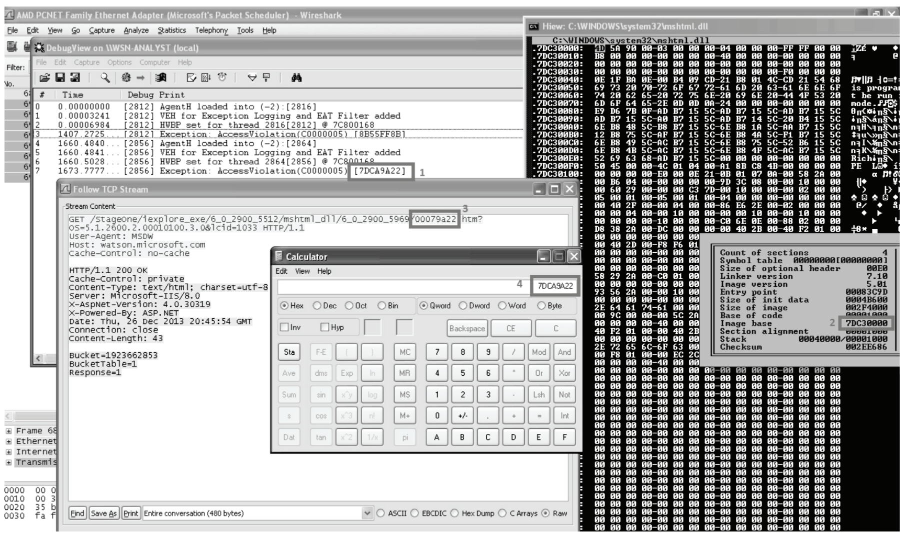
Unfortunately, hackers can access these reports because they usually are sent in unencrypted, clear-text format. The sensitive and finely detailed information they contain enable an attacker to quickly create a precise blueprint of a target's hardware and software environment, and use this information to create tailored attacks.

Using WER to Detect the "Unknown Unknowns"

Before diving into how we can use WER to discover anomalies that might indicate attacks, it's important to understand where WER sits in the operating system stack and what it sees during an exploit attempt. We'll run a known exploit on Internet Explorer to illustrate (see Figure 1).[3]

[2] "1 Billion PCs At Risk As Windows Error Reporting Sends Reports In Clear," Jan. 1, 2014, www.forbes.com/sites/timworstall/2014/01/01/1-billion-pcs-at-risk-as-windows-error-reporting-sends-reports-in-clear/

[3] While very detailed information is generated by the endpoint WER debugger, only key data is sent in the initial (i.e., "StageOne") crash report.



```
AMD PCNET Family Ethernet Adapter (Microsoft's Packet Scheduler) - Wireshark
DebugView on \\WSN-ANALYST (local)
#  Time           Debug Print
0  0.00000000     [2812] AgentH loaded into (-2):[2816]
1  0.00003241     [2812] VEH for Exception Logging and EAT Filter added
2  0.00006984     [2812] HWBP set for thread 2816[2812] @ 7C800168
3  1407.2725...   [2812] Exception: AccessViolation(C0000005) [8B55FF8B]
4  1660.4840...   [2856] AgentH loaded into (-2):[2864]
5  1660.4841...   [2856] VEH for Exception Logging and EAT Filter added
6  1660.5028...   [2856] HWBP set for thread 2864[2856] @ 7C800168
7  1673.7777...   [2856] Exception: AccessViolation(C0000005) [7DCA9A22] 1
```

```
Follow TCP Stream
Stream Content                                                   3
GET /StageOne/iexplore_exe/6_0_2900_5512/mshtml_dll/6_0_2900_5969/00079a22.htm?
OS=5.1.2600.2.00010100.3.0&lcid=1033 HTTP/1.1
User-Agent: MSDW
Host: watson.microsoft.com
Cache-Control: no-cache

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 26 Dec 2013 20:45:54 GMT
Connection: close
Content-Length: 43

Bucket=1923662853
BucketTable=1
Response=1

Entire conversation (480 bytes)
```

```
Calculator
4 7DCA9A22
```

```
Hiew: C:\WINDOWS\system32\mshtml.dll
Count of sections                    4
Symbol table     00000000[00000000]
Size of optional header           00E0
Linker version                    7.10
Image version                     5.01
Entry point                   00083C9D
Size of init data             0004B600
Size of image                 002F4000
Base of code                  00001000
Image base                  2 7DC30000
Section alignment             00001000
Stack                00040000/00001000
Checksum                      002EE686
```

Figure 1:

1) The exploit fails and causes an AccessViolation in Internet Explorer at location 7DCA9A22.

2) We can use a hex editor such as Hiew to see the base of the faulting mshtml.dll module that the exploit corrupted at 7DC30000.

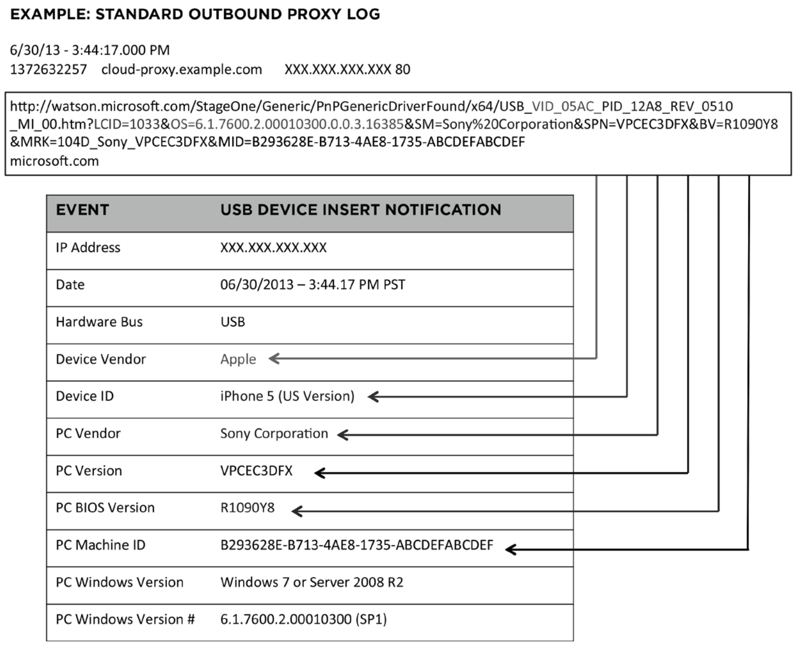
3) The crash information is handed from the Windows OS to WER, which generates a crash dump report including the StageOne metadata with an application crash offset of 00079a22 that is sent to Watson.microsoft.com.

4) By subtracting the Image Base from the Exception Address, you can calculate the application crash offset that is sent to Microsoft in the StageOne report.

If the information shown in Figure 1 can be correlated with other exploits across different operating systems and applications, we can have a way to both detect when attacks occurred, and detect anomalies indicative of new zero-days.

Sample Application Crash Report

Let's now take a closer look at the information contained in a WER StageOne report. These reports can be gathered in a variety of ways, either by examining outbound web proxy logs, creating an IPS rule in an open source IPS, or by simply monitoring a SPAN port. Our example shows a hardware change notification sent when a device was plugged into a computer via USB (see Figure 2).



EXAMPLE: STANDARD OUTBOUND PROXY LOG

```
6/30/13 - 3:44:17.000 PM
1372632257   cloud-proxy.example.com      XXX.XXX.XXX.XXX 80
```

```
http://watson.microsoft.com/StageOne/Generic/PnPGenericDriverFound/x64/USB_VID_05AC_PID_12A8_REV_0510
_MI_00.htm?LCID=1033&OS=6.1.7600.2.00010300.0.0.3.16385&SM=Sony%20Corporation&SPN=VPCEC3DFX&BV=R1090Y8
&MRK=104D_Sony_VPCEC3DFX&MID=B293628E-B713-4AE8-1735-ABCDEFABCDEF
microsoft.com
```

EVENT	USB DEVICE INSERT NOTIFICATION
IP Address	XXX.XXX.XXX.XXX
Date	06/30/2013 – 3:44.17 PM PST
Hardware Bus	USB
Device Vendor	Apple
Device ID	iPhone 5 (US Version)
PC Vendor	Sony Corporation
PC Version	VPCEC3DFX
PC BIOS Version	R1090Y8
PC Machine ID	B293628E-B713-4AE8-1735-ABCDEFABCDEF
PC Windows Version	Windows 7 or Server 2008 R2
PC Windows Version #	6.1.7600.2.00010300 (SP1)

Any attacker intercepting such data can create a precise blueprint of the target's hardware and software network, which can be used to create and execute tailored attacks.[4]

Figure 2: After building lookup tables, we can determine that an Apple iPhone 5 was plugged into a Sony Vaio notebook with Machine ID B293628 running Windows 7 SP1 with a BIOS version of R1090Y8.

[4] We presume this is the methodology alluded to in a Der Spiegel report on documents leaked by Edward Snowden: www.spiegel.de/international/world/0,1518,940969,00.html

Using WER Data in Statistical Analysis of Crash Reports

If you've ever wondered at how Microsoft always seems to know when and where attacks first occurred, here's your answer: WER generates a tremendous amount of actionable data. Websense sees about 640 million reports across its infrastructure each year; Microsoft sees billions.

In Oct. 2009, Microsoft published a white paper about how it approaches debugging in a large-scale environment where hundreds of millions of machines submit reports.[5] To solve the problem of building a distributed debugging system in the early 2000s, Microsoft began researching how to accomplish statistics-based debugging, which it deemed "critical to reduce the billions of reports they receive, mine the error report database to prioritize work, spot trends and test hypotheses."[6]

According to the white paper, groups of application crashes reported to Microsoft are initially labeled and assigned to a "bucket," so that each bucket of crash data contains errors related to a single, specific fault.

As WER further classifies and collects data, bucket identifiers can be combined and grouped to glean further insights from the crash data. Understanding and knowing how to use the proprietary WER bucketing algorithm is critical to understanding crash telemetry.

Applying the WER Bucketing Algorithm to Crash Reports

The WER bucketing algorithm works is shrouded in proprietary secrecy, but one thing is clear: its output — which ideally groups crash reports with the same root cause — is a valuable first step toward isolating crash reports that indicate exploit attempts.

The bucketing algorithm can help us understand a sample set of WER reports. Our sample of 100,000 application crashes contained 8804 unique application crash reports that were sent to Microsoft. The bucketing algorithm was able to reduce these 8804 reports to 8558 unique groupings of application crashes with a common cause, and provided administrators with a hyperlink to instructions on how to solve the crash problem 218 times.

In the sample set of crash reports that was run against Microsoft's bucketing algorithm, the algorithm also was capable of linking up to four independent crash reports together to a similar cause, most groupings accounting for variations in reporting between different OS builds. In certain cases, the bucketing algorithm is capable of linking together reports with different application crashes offset (see Table 1). This valuable capability of the bucketing algorithm helps us significantly, because it reduces the number of crash reports we need to analyze and in many cases, provides historical context about whether Microsoft has seen the crash before.

http://watson.microsoft.com/StageOne/iexplore_exe/8_0_7601_17514/4ce79912/StackHash_0a9e/0_0_0_0/00000000/c0000005/00000000.htm
http://watson.microsoft.com/StageOne/iexplore_exe/8_0_7601_17514/4ce79912/StackHash_0a9e/0_0_0_0/00000000/c0000005/0000000b.htm
http://watson.microsoft.com/StageOne/iexplore_exe/8_0_7601_17514/4ce79912/StackHash_0a9e/0_0_0_0/00000000/c0000005/01f80f03.htm
http://watson.microsoft.com/StageOne/iexplore_exe/8_0_7601_17514/4ce79912/StackHash_0a9e/0_0_0_0/00000000/c0000005/03190b60.htm

Table 1. These four Internet Explorer crash reports from our sample set happened at different crash locations, yet were grouped together because they have the same root cause.

[5] "Debugging in the (Very) Large: Ten Years of Implementation and Experience," www.sigops.org/sosp/sosp09/papers/glerum-sosp09.pdf

[6] Ibid.

Other statistics we gathered from our sample set include:

- 17,416 distinct applications discovered.
- 34.5 percent of reports were application crashes. Others were automated reports such as update failures and USB drive insertions.
- Most distinct program versions reported:
  - Internet Explorer: 128 versions
  - Firefox: 60 versions
  - Skype: 60 versions
  - Java: 51 versions
  - Acrobat: 39 versions
- Highest number of crashes reported in corporate environments:
  - Internet Explorer
  - Windows processes (SearchProtocolHost, DllHost, SvcHost)
  - Acrobat Reader
  - Microsoft Word
  - Microsoft Explorer

Adding Security Context to Crash Data

Having used the WER bucketing algorithm to start our search for possible exploit attempts, we now can further reduce the data set by focusing on crashes with security relevance. To make the leap from detecting anomalous activity in crash reports to prioritizing anomalous events in the context of targeted attacks, we need to first define potential risk indicators within application crashes; augment those risk indicators with threat intelligence; and incorporate into a risk framework to prioritize events for analysis.

Below are some of the key risk indicators and concepts that we have incorporated into our proprietary crash anomaly classifier:

- Crashes in widely deployed, highly exploitable applications
  - Windows, IE, Adobe, MS Office, Firefox, Chrome
- Crashes in key applications for businesses at high risk of attack
  - SCADA, Point of Sale, Telecom, Research, etc.
- Temporal anomalies, or groups of crashes isolated to a company or industry that might be the indicator of exploit attempts through targeted email, waterholes, or lateral movement of an attacker through a network.
- Unusual faulting libraries, such as Duqu's exploitation of the Windows True Type Embedded Font library (T2embed.dll), which is rarely accessed during normal browsing.
- Indicators of possible code injection, such as applications crashing outside of executable space. (WER often reports this as "StageOne/Generic/AppHangB1," or in cases where no faulting DLL is reported).
- "Fingerprints" for known exploits that detect application crashes according to application version, DLL version, and crash offset locations of known common vulnerabilities and exposures (CVEs). These can be useful to detect crash activity.

Next, we estimated risk to organizations on the following principles (in order of risk):

1. Knowledge of the threat actor
2. Intent of the attack
3. Sophistication of the attack
4. Stage of the kill chain[7]

When we apply these principles to our goal of using crash telemetry logs to detect exploit attempts that have bypassed defenses, we discover:

» A new crash report might be able to serve as a risk indicator of possible attack.

» More specifically, new crash reports isolated to an industry or geographic region might be able to serve as a risk indicator of possible attack.

» More specifically still, new crash reports across these verticals with other indicators of attack — such as use of known threat actors' attack infrastructure, malware and delivery methods — might increase our confidence that we are looking at an attack instead of a series of benign anomalous events.

Testing a Zero-Day Exploit to Create a "Crash Fingerprint"

We'll use a sample attack to apply our principles of risk to validate our hypothesis about using crash reports to discover new attacks.

One of the hottest zero-days of the past year is CVE-2013-3893, which we observed being used in highly targeted attacks in Sept. 2013 against manufacturers of high tech equipment in Taiwan and major financial institutions in Japan.[8] This zero-day was used by the "DeputyDog" crew and the attacks were linked to the same threat actor that compromised Bit9 in February 2013.[9]

Applying the Microsoft Windows debugger WinDbg to a failed processes from a CVE-2013-3893 exploit attempt against a Windows XP machine, Websense Security Labs created a "crash fingerprint":

```
0:015> g
ModLoad: 76360000 76370000   C:\WINDOWS\system32\winsta.dll
ModLoad: 5b860000 5b8b5000   C:\WINDOWS\system32\NETAPI32.dll
(228.10c): Unknown exception - code 000006ba (first chance)
ModLoad: 76b40000 76b6d000   C:\WINDOWS\system32\WINMM.dll
ModLoad: 72d20000 72d29000   C:\WINDOWS\system32\wdmaud.drv
ModLoad: 76c30000 76c5e000   C:\WINDOWS\system32\WINTRUST.dll
ModLoad: 77a80000 77b15000   C:\WINDOWS\system32\CRYPT32.dll
ModLoad: 77b20000 77b32000   C:\WINDOWS\system32\MSASN1.dll
ModLoad: 76c90000 76cb8000   C:\WINDOWS\system32\IMAGEHLP.dll
ModLoad: 72d20000 72d29000   C:\WINDOWS\system32\wdmaud.drv
ModLoad: 72d10000 72d18000   C:\WINDOWS\system32\msacm32.drv
ModLoad: 77be0000 77bf5000   C:\WINDOWS\system32\MSACM32.dll
ModLoad: 77bd0000 77bd7000   C:\WINDOWS\system32\midimap.dll
ModLoad: 3d7a0000 3d854000   C:\WINDOWS\system32\jscript.dll
ModLoad: 76ee0000 76f1c000   C:\WINDOWS\system32\RASAPI32.dll
ModLoad: 76e90000 76ea2000   C:\WINDOWS\system32\rasman.dll
ModLoad: 5b860000 5b8b5000   C:\WINDOWS\system32\NETAPI32.dll
ModLoad: 76eb0000 76edf000   C:\WINDOWS\system32\TAPI32.dll
ModLoad: 76e80000 76e8e000   C:\WINDOWS\system32\rtutils.dll
ModLoad: 722b0000 722b5000   C:\WINDOWS\system32\sensapi.dll
ModLoad: 71e50000 71e65000   C:\WINDOWS\system32\msapsspc.dll
ModLoad: 78080000 78091000   C:\WINDOWS\system32\MSVCRT40.dll
ModLoad: 767f0000 76819000   C:\WINDOWS\system32\schannel.dll
ModLoad: 75b00000 75b15000   C:\WINDOWS\system32\digest.dll
```

[7] For a description of the kill chain, visit: www.websense.com/assets/white-papers/whitepaper-7-stages-of-advanced-threats-cyber-attack-kill-chain-en.pdf

[8] Websense first spotted the attacker's campaign on July 1, 2013. Learn more at http://community.websense.com/blogs/securitylabs/archive/2013/09/26/zero-day-analysis-cve-2013-3893-attacks-more-widespread-than-previously-reported.aspx

[9] https://blog.bit9.com/2013/02/25/bit9-security-incident-update/

```
ModLoad: 747b0000 747f7000   C:\WINDOWS\system32\msnsspc.dll
ModLoad: 78080000 78091000   C:\WINDOWS\system32\MSVCRT40.dll
ModLoad: 77c70000 77c95000   C:\WINDOWS\system32\msv1_0.dll
ModLoad: 76790000 7679c000   C:\WINDOWS\system32\cryptdll.dll
ModLoad: 76d60000 76d79000   C:\WINDOWS\system32\iphlpapi.dll
ModLoad: 71a50000 71a8f000   C:\WINDOWS\system32\mswsock.dll
ModLoad: 662b0000 66308000   C:\WINDOWS\system32\hnetcfg.dll
ModLoad: 71a90000 71a98000   C:\WINDOWS\System32\wshtcpip.dll
Heap corruption detected at 12320000
Heap corruption detected at 12120018
Heap corruption detected at 11F20000
(228.10c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=11f20000 ebx=00000000 ecx=11420000 edx=11920000 esi=11920000 edi=00150000
eip=7c929f09 esp=015dd090 ebp=015dd14c iopl=0         nv up ei pl nz ac pe cy
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010217
ntdll!towlower+0x517:
7c929f09 3b5004          cmp     edx,dword ptr [eax+4] ds:0023:11f20004=????????
```

Below is the output from the exception analysis command we executed to find all of the relevant information that is included in a WER report:

```
FAULTING_IP:
ntdll!towlower+517
7c929f09 3b5004          cmp     edx,dword ptr [eax+4]

EXCEPTION_RECORD:  ffffffff -- (.exr 0xffffffffffffffff)
ExceptionAddress: 7c929f09 (ntdll!towlower+0x00000517)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000000
   Parameter[1]: 11f20004
Attempt to read from address 11f20004

FAULTING_THREAD:  000005e0
PROCESS_NAME:  iexplore.exe
```

Bingo! We have our fingerprint:

```
WATSON_STAGEONE_URL: hxxp://watson.microsoft.com/StageOne/iexplore_exe/8_0_6001_18702/49b3ad2e/ntdll_
dll/5_1_2600_6055/4d00f27d/c0000005/00029f09.htm?Retriage=1
```

Using a Fingerprint to Discover a New APT Attack

This fingerprint enabled us to search for other (unknown) examples of this crash patterns among a sample set of 16 million Windows Error Reports.

Using the Websense ThreatSeeker® Intelligence Cloud,[10] we detected six crash report anomalies — from five distinct organizations — that may be indicators of failed CVE-2013-3893 exploit attempts.

One organization of the five had generated WERs containing two crash fingerprints and, because the company was a mobile network operator, it stood out immediately as a possible target of an APT attack. Let's take a look at what we found:

Potential Target: Tier-1 Mobile Network Operator

Industry: Telecommunications

Date: 12-16-2013

Crash 12-10-2013 URL: http://watson.microsoft.com/StageOne/iexplore_exe/8_0_6001_18702/ntdll_dll/5_1_2600_6055/00029f09.htm?

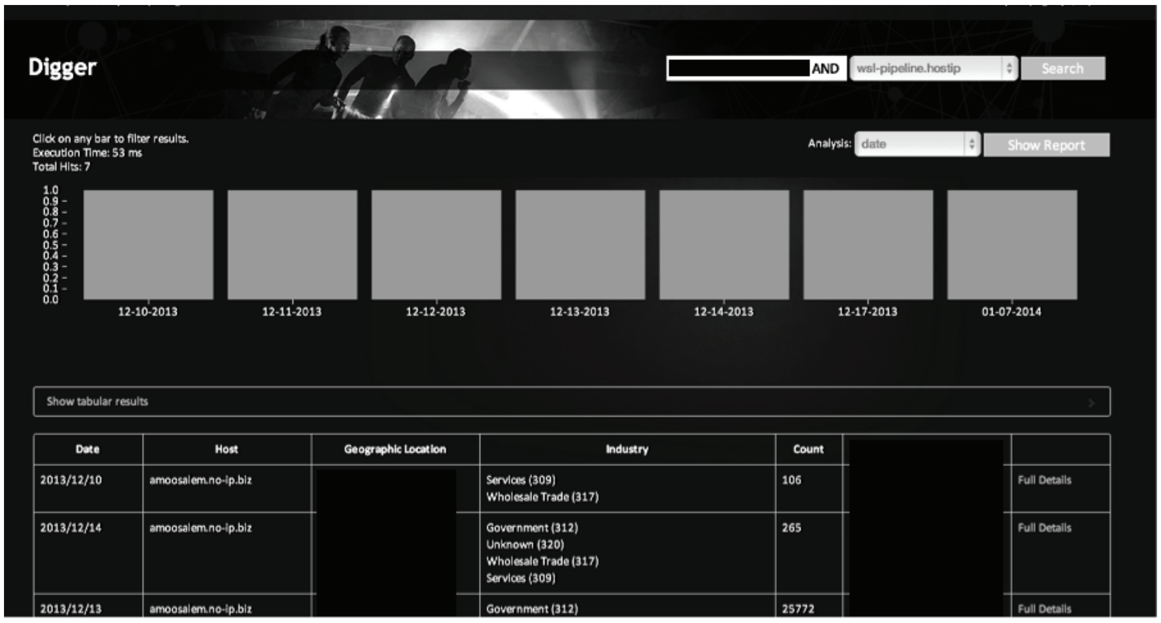
Crash 12-16-2013 URL: http://watson.microsoft.com/StageOne/iexplore_exe/8_0_6001_18702/ntdll_dll/5_1_2600_6055/00029f09.htm?

These results indicate a possible failed exploit attempt of CVE-2013-3893 on the mobile network operator's network on Dec. 10 and 16, 2013.

[10] The world's largest information security intelligence network. Learn more at www.websense.com/content/websense-threatseeker-network.aspx

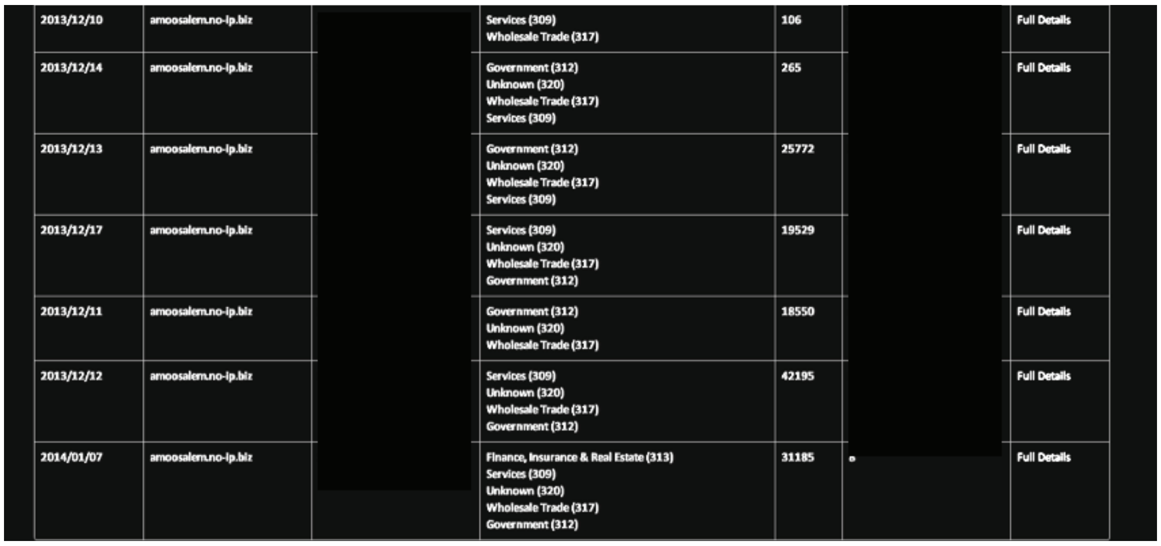
Further investigation validated the timing of these five crash report anomalies with advanced persistent threats (APTs). The use of the zero-day vulnerability, combined with the specific malware used and the prominence of the organizations targeted, revealed the presence of several active unreported APT campaigns.

On the same day of the first exploit attempt, Websense ACE (Advanced Classification Engine) real-time analytics stopped command-and-control (beacon) traffic going to "amoosalem.no-ip.biz" that our analytics attribute to the Houdini H-Worm, a remote access Trojan (RAT) often associated with targeted attacks (see Figures 3 and 4).[11]



Click on any bar to filter results.
Execution Time: 53 ms
Total Hits: 7

Date	Host	Geographic Location	Industry	Count	[redacted]	
2013/12/10	amoosalem.no-ip.biz	[redacted]	Services (309) Wholesale Trade (317)	106	[redacted]	Full Details
2013/12/14	amoosalem.no-ip.biz	[redacted]	Government (312) Unknown (320) Wholesale Trade (317) Services (309)	265	[redacted]	Full Details
2013/12/13	amoosalem.no-ip.biz	[redacted]	Government (312)	25772	[redacted]	Full Details



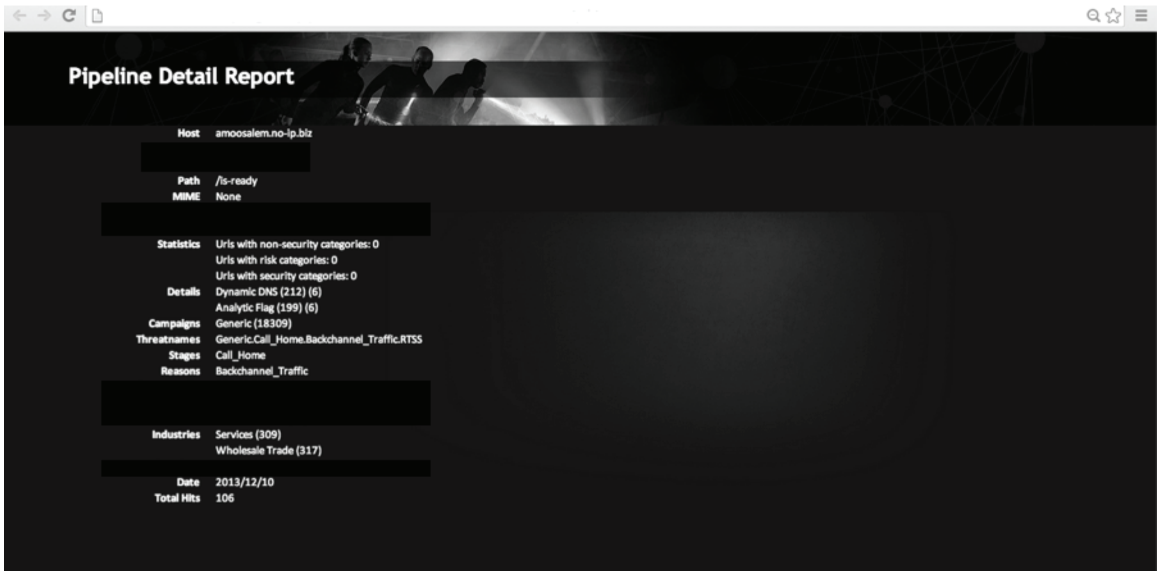
2013/12/10	amoosalem.no-ip.biz	[redacted]	Services (309) Wholesale Trade (317)	106	[redacted]	Full Details
2013/12/14	amoosalem.no-ip.biz	[redacted]	Government (312) Unknown (320) Wholesale Trade (317) Services (309)	265	[redacted]	Full Details
2013/12/13	amoosalem.no-ip.biz	[redacted]	Government (312) Unknown (320) Wholesale Trade (317) Services (309)	25772	[redacted]	Full Details
2013/12/17	amoosalem.no-ip.biz	[redacted]	Services (309) Unknown (320) Wholesale Trade (317) Government (312)	19529	[redacted]	Full Details
2013/12/11	amoosalem.no-ip.biz	[redacted]	Government (312) Unknown (320) Wholesale Trade (317)	18550	[redacted]	Full Details
2013/12/12	amoosalem.no-ip.biz	[redacted]	Services (309) Unknown (320) Wholesale Trade (317) Government (312)	42195	[redacted]	Full Details
2014/01/07	amoosalem.no-ip.biz	[redacted]	Finance, Insurance & Real Estate (313) Services (309) Unknown (320) Wholesale Trade (317) Government (312)	31185	[redacted]	Full Details

Figure 3 and 4: A list of command and control servers that Websense ACE analytics detected and provided protection against for the targeted organization starting the same day as the possible failed exploit attempt using CVE-2013-3893.

[11] H-Worm is a Visual Basic Script (VBS)-based RAT which is believed to have been written by an Algerian author who goes by the name "Houdini." Historically, we have seen the H-worm RAT wrapped in a PE-executable dropper with multiple levels of obfuscation to avoid AV detection. Upon successful installation of the RAT, H-Worm beacons back to its command and control servers with a signature HTTP POST /ready HTTP/1.1 (older versions), and HTTP POST/is-ready HTTP/1.1 (newer versions). H-WORM's command messages contain typical RAT functionality, with the ability to allow attackers to execute shell commands, enumerate and download files, and recon running processes and delete files. Historically, we have seen H-Worm used to target primarily government, telecommunications services and manufacturing industries.

We had not previously seen “amoosalem.no-ip.biz” used in attacks with the H-Worm RAT, but it appears to be consistent with other domain names that have been used in the past (we have seen at least one group using Houdini utilize the dynamic no-ip.biz domain).

Taking a closer look at the domain and request structure, we can confirm the traffic as the Houdini H-Worm RAT — the /is-ready HTTP request is a tell-tale sign of the more recent versions of the Houdini malware (see Figure 5).



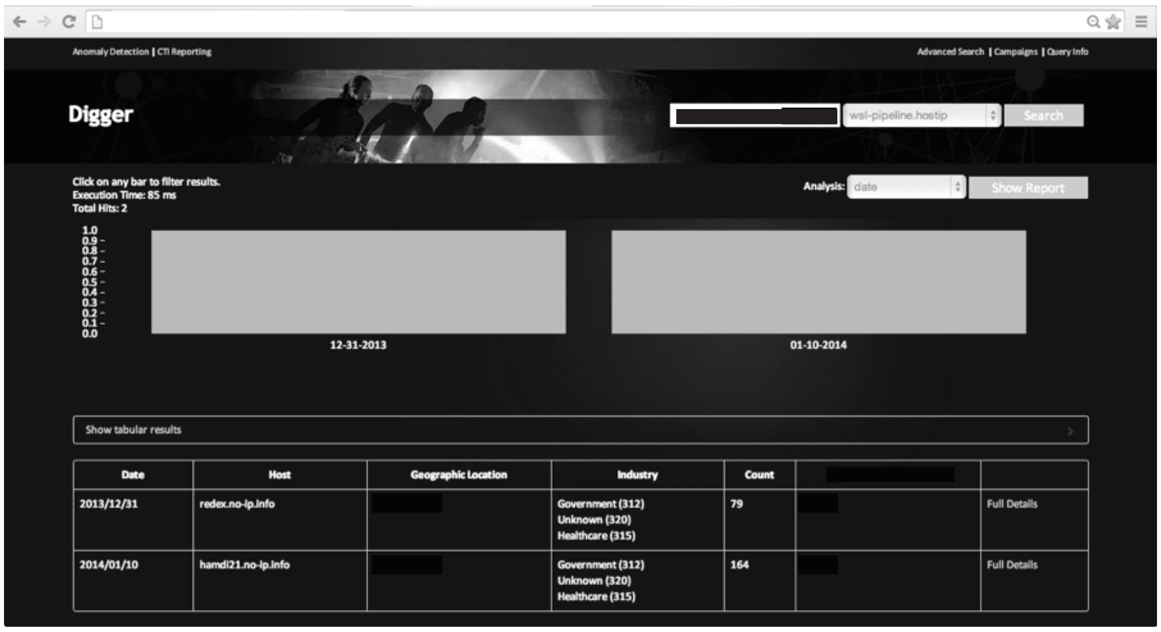
Pipeline Detail Report

Host	amoosalem.no-ip.biz
[redacted]	
Path	/is-ready
MIME	None
[redacted]	
Statistics	Urls with non-security categories: 0 Urls with risk categories: 0 Urls with security categories: 0
Details	Dynamic DNS (212) (6) Analytic Flag (199) (6)
Campaigns	Generic (18309)
Threatnames	Generic.Call_Home.Backchannel_Traffic.RTSS
Stages	Call_Home
Reasons	Backchannel_Traffic
[redacted]	
Industries	Services (309) Wholesale Trade (317)
[redacted]	
Date	2013/12/10
Total Hits	106

Figure 5

Found: A Second Victim

A second potential victim identified through crash telemetry data, a government organization, also had the Houdini H-Worm RAT begin beaconing to a command-and-control server within the same time period (see Figure 6).



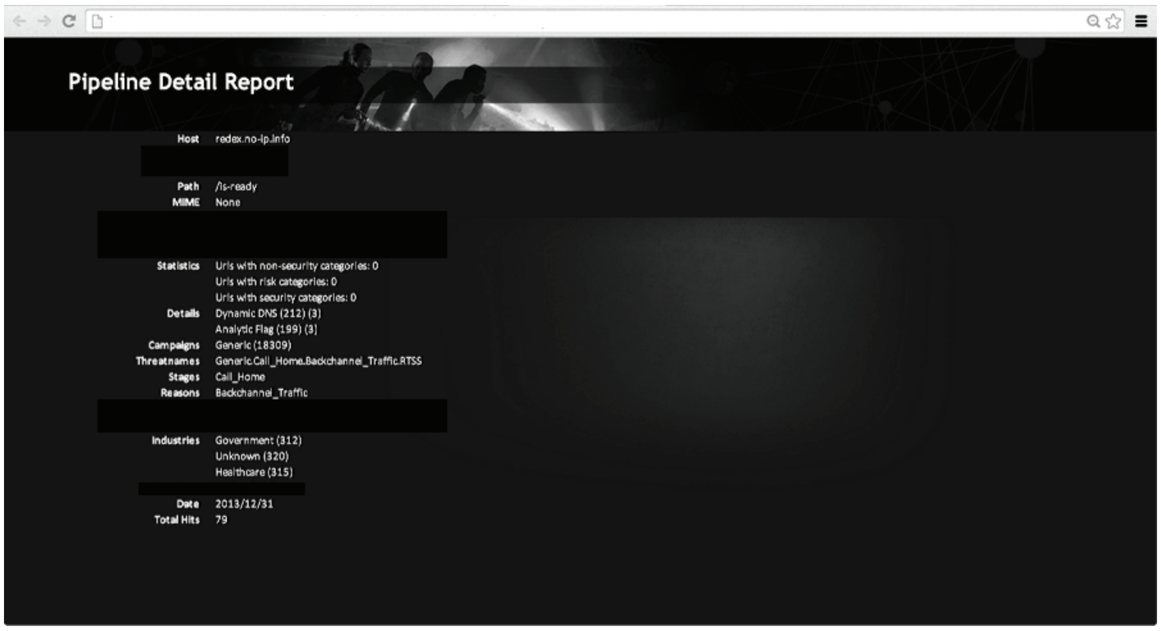
[redacted]

Click on any bar to filter results.
Execution Time: 85 ms
Total Hits: 2

Date	Host	Geographic Location	Industry	Count	[redacted]	
2013/12/31	redex.no-ip.info	[redacted]	Government (312) Unknown (320) Healthcare (315)	79	[redacted]	Full Details
2014/01/10	hamdi21.no-ip.info	[redacted]	Government (312) Unknown (320) Healthcare (315)	164	[redacted]	Full Details

Figure 6

These URLs also seem similar to those used with the H-Worm RAT in the past. Let's take a closer look (see Figure 7):


Pipeline Detail Report

Host	redex.no-ip.info
[redacted]	
Path	/is-ready
MIME	None
[redacted]	
Statistics	Uris with non-security categories: 0
	Uris with risk categories: 0
	Uris with security categories: 0
Details	Dynamic DNS (212) (3)
	Analytic Flag (199) (3)
Campaigns	Generic (18309)
Threatnames	Generic.Call_Home.Backchannel_Traffic.RTSS
Stages	Call_Home
Reasons	Backchannel_Traffic
[redacted]	
Industries	Government (312)
	Unknown (320)
	Healthcare (315)
[redacted]	
Date	2013/12/31
Total Hits	79

Figure 7

To recap:

» We started by creating a fingerprint to detect possible failed exploit attempts using CVE-2013-3893.

» After finding six possible fingerprints across five different organizations using four months of data, we added security context to help substantiate our hypothesis.

» In the case of the mobile network operator, we were able to confirm that the H-Worm RAT, a RAT often used in targeted attacks, began beaconing back to its command and control infrastructure on the same day as the initial exploit attempt. This indicated that a second successful exploit may have occurred that day as well, resulting in installation of the H-Worm RAT by the attacker.

These examples helped validate our theory that crash reports, when used correctly can be a great risk indicator to help companies focus their resources to detect advanced attacks.

Using Crash Reports to Discover a Point-of-Sale (POS) Systems Attack

Data breaches of retail organizations, including Target and Niemen Marcus, have dominated recent news cycles.[12]

At the heart of the Target Corp. breach was a novel piece of malware known as a point-of-sale RAM scraper (POSRAM). "Reedum" and several other POSRAM variants are capable of targeting applications used at POS terminals to search for credit card numbers, credentials and customer billing information.

We'll now use crash reports to examine POSRAM malware using the following methodology and questions:

» What risk indicators can we learn from the POSRAM malware?

» Can we use those risk indicators to identify malicious intent?

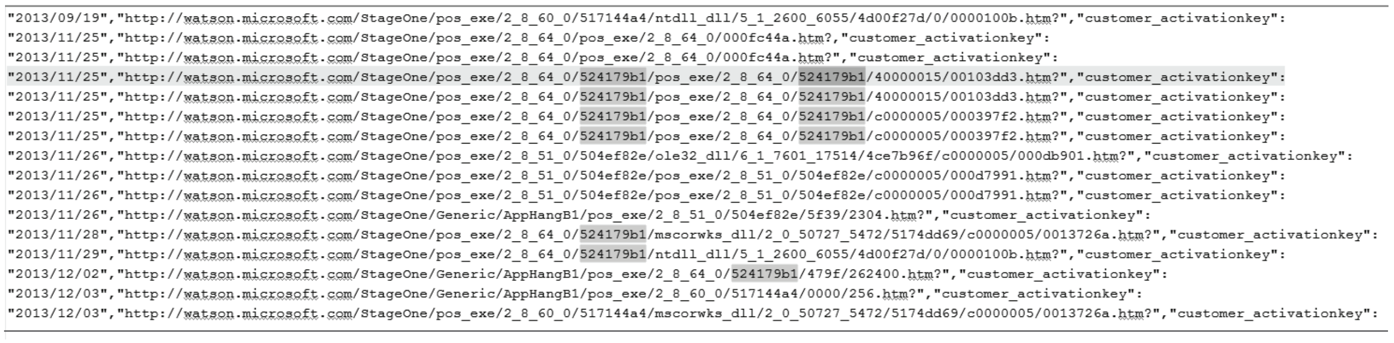
» Can we characterize these malware campaigns and all of the above to detect any persistence or traffic verticals (APT escalation)?

» Can we use a combination of these risk indicators to describe the target uniquely (APT escalation)?

[12] "Six Ways to Prevent Point-of-Sale Malware Attacks," Jan. 30, 2014, community.websense.com/blogs/websense-insights/archive/2014/01/30/six-ways-to-prevent-point-of-sale-malware-attacks.aspx

We'll start by searching for application crash reports from POS applications such as those targeted by the POSRAM malware (see Figure 8). The big question is whether the risk indicators (crashes) that we see on pos.exe, a common POS process, across our telemetry feeds indicate risk or malicious activity, or are simply indicative of bugs in software applications.

As expected, we saw a wide variety of crashes in normal program activity of POS applications in our 16-Million record data set. However, one organization stood out as having anomalous crashes.



```
"2013/09/19","http://watson.microsoft.com/StageOne/pos_exe/2_8_60_0/517144a4/ntdll_dll/5_1_2600_6055/4d00f27d/0/0000100b.htm?","customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/pos_exe/2_8_64_0/000fc44a.htm?,"customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/pos_exe/2_8_64_0/000fc44a.htm?","customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/pos_exe/2_8_64_0/524179b1/40000015/00103dd3.htm?","customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/pos_exe/2_8_64_0/524179b1/40000015/00103dd3.htm?","customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/pos_exe/2_8_64_0/524179b1/c0000005/000397f2.htm?","customer_activationkey":
"2013/11/25","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/pos_exe/2_8_64_0/524179b1/c0000005/000397f2.htm?","customer_activationkey":
"2013/11/26","http://watson.microsoft.com/StageOne/pos_exe/2_8_51_0/504ef82e/ole32_dll/6_1_7601_17514/4ce7b96f/c0000005/000db901.htm?","customer_activationkey":
"2013/11/26","http://watson.microsoft.com/StageOne/pos_exe/2_8_51_0/504ef82e/pos_exe/2_8_51_0/504ef82e/c0000005/000d7991.htm?","customer_activationkey":
"2013/11/26","http://watson.microsoft.com/StageOne/pos_exe/2_8_51_0/504ef82e/pos_exe/2_8_51_0/504ef82e/c0000005/000d7991.htm?","customer_activationkey":
"2013/11/26","http://watson.microsoft.com/StageOne/Generic/AppHangB1/pos_exe/2_8_51_0/504ef82e/5f39/2304.htm?","customer_activationkey":
"2013/11/28","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/mscorwks_dll/2_0_50727_5472/5174dd69/c0000005/0013726a.htm?","customer_activationkey":
"2013/11/29","http://watson.microsoft.com/StageOne/pos_exe/2_8_64_0/524179b1/ntdll_dll/5_1_2600_6055/4d00f27d/0/0000100b.htm?","customer_activationkey":
"2013/12/02","http://watson.microsoft.com/StageOne/Generic/AppHangB1/pos_exe/2_8_64_0/524179b1/479f/262400.htm?","customer_activationkey":
"2013/12/03","http://watson.microsoft.com/StageOne/Generic/AppHangB1/pos_exe/2_8_60_0/517144a4/0000/256.htm?","customer_activationkey":
"2013/12/03","http://watson.microsoft.com/StageOne/pos_exe/2_8_60_0/517144a4/mscorwks_dll/2_0_50727_5472/5174dd69/c0000005/0013726a.htm?","customer_activationkey":
```

Figure 8: A cluster of anomalous crash reports appear to be occurring from a large wholesale clothing retailer located in Eastern United States.

There are two dominant versions of crash records that we can see in Figure 8. The first type, StageOne failures, are crashes that happen within the pos.exe application at various places, and more likely indicate actual program crashes than attacks.

The second (and more interesting) crash records that we see are reported to WER with the format AppHangB1/<exe>/<version>/:

hxxp://watson.microsoft.com/StageOne/Generic/AppHangB1/pos_exe/2_8_64_0/524179b1/479f/262400.htm?

- » 262400 – Virtual address on system
- » Most likely a run-time error existing in a module outside the application developers program

These crashes are happening outside of executable code modules, indicating a possible injection of code that could occur in a return orientated programming (ROP) or heap-spray exploit technique.

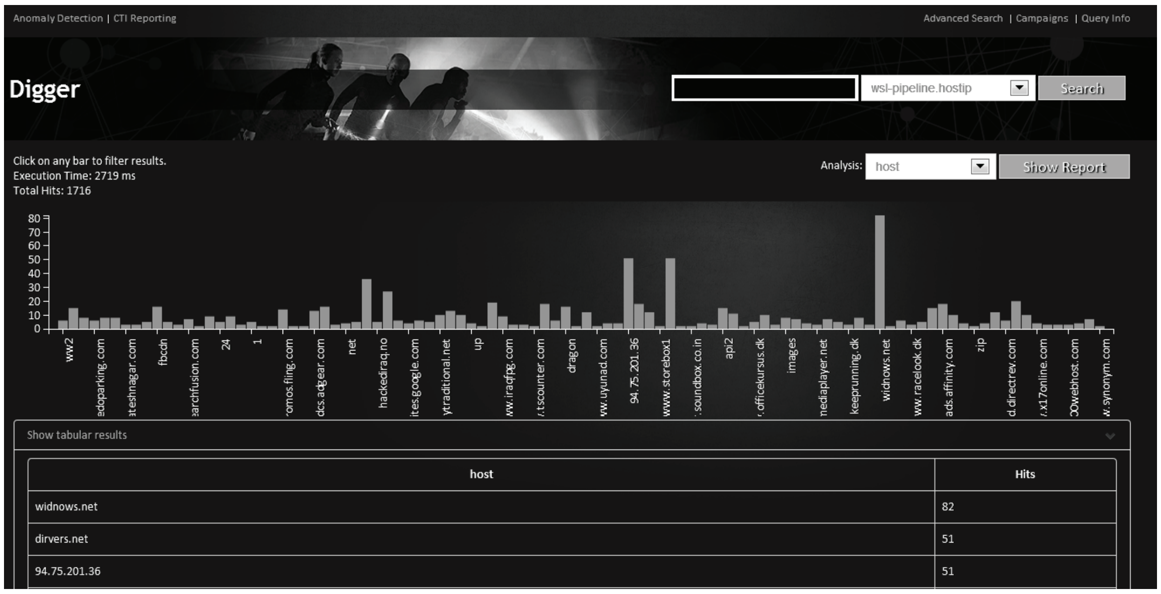
Zeus Makes an Appearance

Now we will examine how we can expand on this risk indicator by correlating it with additional telemetry.

Below is a simple count for number of crashes that stood out in Figure 8:

- » Six crashes on 2013/11/25
- » Four crashes on 2013/11/26
- » Two crashes on 2013/12/03

We now have a potentially targeted organization and time range that we are interested in. We searched the Websense® ThreatSeeker® Intellignece Cloud, which sees billions of daily security-related events, for any suspicious traffic or attacks that may have occurred on the targeted organizations' network during that time period (see Figure 9).



host	Hits
widnows.net	82
dirvers.net	51
94.75.201.36	51

Figure 9

In Figure 9, note that at the top of the list are three suspect hosts that have been visited consistently by machines at the organization: widnows.net, dirvers.net and 94.75.201.36. Websense real-time analytics detected traffic to these sites as Zeus command-and-control (C&C), and a quick look at the URL structure requests confirms what are obviously machine-generated URL request structures associated in the past with Zeus C&C.[13]

A first thought for most security or incident response teams after seeing a Zeus infection is to re-image the affected systems on the network and move on. However, when we add security context to the risk indicators that we have seen it becomes much more interesting. If this is just a standard Zeus mass-malware infection, we would expect to see a wide variety of industries targeted and compromised over a period of time (See Figure 10).

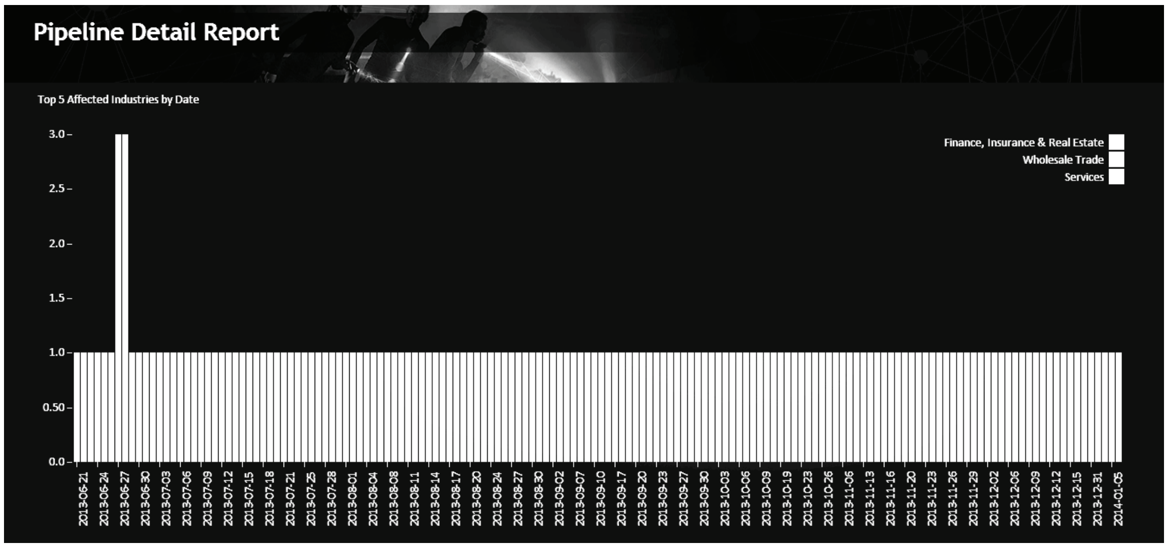


Figure 10

A quick glance at Figure 10 reveals that this is most definitely not a mass malware infection, but rather one that is targeting businesses specifically in the wholesale trade sector — very much different than a typical Zeus infection that is evenly distributed across industries such as financial, government and healthcare. This creates yet another risk indicator that we may be looking at a targeted campaign focused on POS applications.

[13] Examples of Zeus C&C: /sadcxvbv/vdfbffddf.php, /sadcxvbv/fgfvsada.php

We saw spikes in activity in C&C traffic around the dates of our application crashes in late November, moving from a typical amount of approximately 20 requests per day to approximately 300. This indicated two likely possibilities:

» Data exfiltration attempts (stealing customer credit card numbers, billing info, etc.).

» Malware may be causing the POS terminals to crash, forcing restarting of the terminals and increased attempts from the malware to reach C&C servers that were blocked by Websense.

The Relevance of Zeus

Taking a step back, we can ascertain three things from our analysis of anomalous behaviors derived from crash reports:

» We know that some software functions related to the RAM scraping capability (searching RAM for credit card numbers) were originally discovered as part of the same source code tree as the Zeus malware.

» We can see that this retailer has what appears to be a variant of Zeus malware, with increased attempts to contact command-and-control servers in the same time period as the application crashes were observed.

» The three command-and-control servers that we have observed do not appear to be part of a typical Zeus-based mass-malware infection, but targeted specifically at the wholesale/retailer industry.

We therefore believe that these results indicate that malware based on the leaked Zeus and RAM-scraping code is actively targeting POS terminals to steal customer credit card data.

Conclusion

The arguments and examples in this white paper support our belief that even advanced attacks leave a digital “trail” of evidence during their attacks, and that next-generation security systems must be capable of constantly searching for anomalous activity that could be an indication of an ongoing attack.

Further, we have validated that the use of risk indicators as a starting point to detect advanced attacks, which can then be augmented with threat intelligence and security context, can allow us to draw correlations and discover unknown attacks in ways that are not possible using today’s signature- and expert knowledge-based systems. Below are our four conclusions:

1. Windows error reports leak data.
2. For the first time ever, these reports have been used to detect advanced attacks in the wild. The methodology has created a new means of identifying previously unknown threats – attacks that have made it past organizations defenses – in a manner never before accomplished.
3. Using zero-day “Crash Fingerprints” we discovered a new APT attack on a global telecommunication company and a government entity.
4. Using error report telemetry, we also have identified a previously unreported campaign against point-of-sale (POS) systems.

Next Steps

A key to understanding risk indicators is the ability to augment them with a global intelligence perspective. As organizations and security solutions develop new and innovative ways to share threat intelligence, it will become increasingly possible to detect previously unknown and targeted attacks which pose the highest risk to businesses.

To that end, Websense will release additional information about how to decode WER application crash telemetry reports, incorporate these risk indicators into SIEM tools, and share data between industries through groups such as CERTS and industry advisory committees. **http://www.websense.com/DrWatsonGITHUB.**

websense®

Learn more at **www.websense.com** | **+1 800-723-1166** | **info@websense.com**

TRITON STOPS MORE THREATS.
WE CAN PROVE IT.

websense
TRITON®