SurfRight
A SOPHOS Company

HitmanPro.Alert

Exploit Test Tool Manual

Table of contents

1 Introduction to the Exploit Test Tool

To check a pc's security posture or verify the correct working of HitmanPro.Alert, we have developed an Exploit Test Tool that is able to perform specific exploit techniques. The exploit techniques performed by the Exploit Test Tool are not malicious and safe to use.

If HitmanPro.Alert was successfully installed and is active, every individual exploit technique test that can be started with the Exploit Test Tool should lead to an intercept message from HitmanPro.Alert. But you can also use the Exploit Test Tool without HitmanPro.Alert to check the abilities of other security software on the computer, like Anti-Virus, Internet Security, Advanced Endpoint Protection, Host Intrusion Prevention System, Exploit Mitigation tools or Anti-Exploit software.

1.1 Individual exploit techniques

Whereas in a real-world scenario attackers need to combine multiple exploit techniques to successfully exploit a software vulnerability, the Exploit Test Tool is purposely designed to trigger single exploit techniques to test and reveal the individual capabilities of security software. Because attackers can e.g. unpivot[1] a (nowadays common) stack pivot technique before security software inspects the stack, they effectively bypass a stack pivot security check (see 3.5 Unpivot Stack). Since more security vendors are adding stack pivot detection to their software, attackers are expected to update their techniques.

By triggering single exploit techniques end-users will get a better and more reliable understanding regarding their security software's ability to stop individual attack techniques or phases. Being able to block more individual exploit techniques means that security software is not only more robust against todays exploits, but also to tailored and unknown future attacks as well.

1.2 Dynamic searching for ROP gadgets

Some anti-exploit programs have a signature-based approach to stop (some) individual in-the-wild exploit attacks. Like a real attacker, the Exploit Test Tool will dynamically search for ROP gadgets in legitimate software on your computer to facilitate some of the ROP tests. This approach will expose anti-exploit software that claim signature-less technique-based ROP mitigations, as they will fail to detect these tests.

[1] https://bromiumlabs.files.wordpress.com/2014/02/bypassing-emet-4-1.pdf

1.3 How to use

There are 2 versions of the Exploit Test Tool, a 32-bit and a 64-bit version, each with their own distinguishable icon, see Figure 1. Note that the 32-bit version of the tool contains more tests than the 64 bit version. Some of the exploit techniques are not present on 64-bit versions of the Windows operating system.

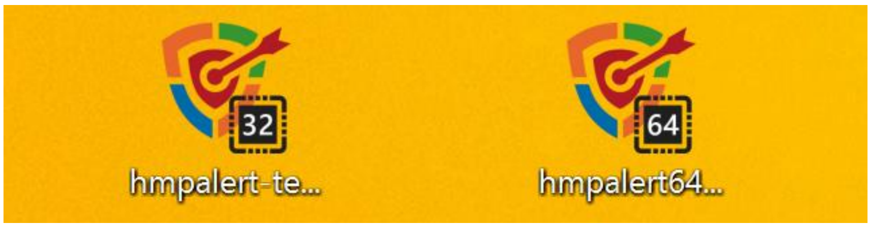
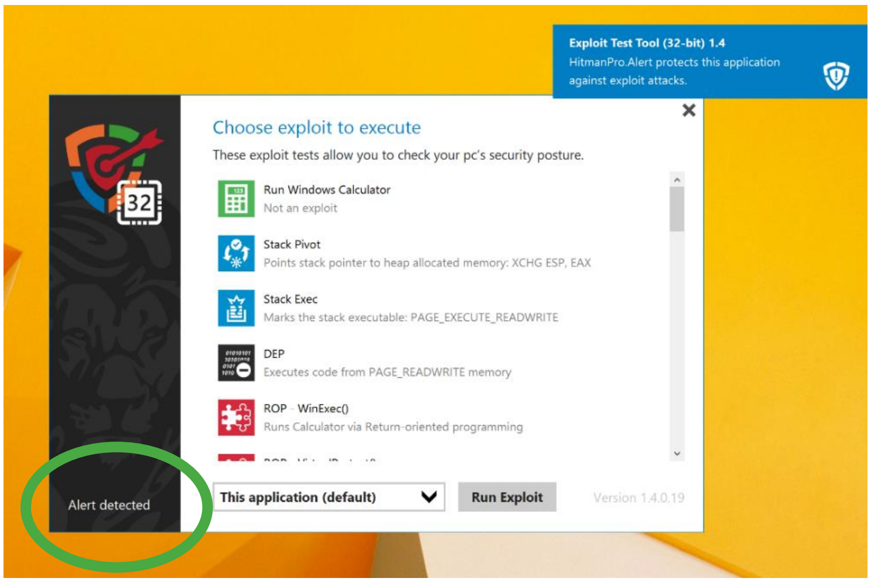


Figure 1: 32-bit and 64-bit version of the HitmanPro.Alert Exploit Test Tool

Once the Exploit Test Tool is started, Figure 2: Test selection panel will appear. When HitmanPro.Alert is installed, you will see a **notification message** in the top right corner of the screen, sliding onto the desktop to indicate that HitmanPro.Alert is actively protecting the Exploit Test Tool. In the left bottom corner of the Exploit Test Tool, the text **Alert detected** appears when the presence of HitmanPro.Alert is detected in the Test Tool's running process.



Exploit Test Tool (32-bit) 1.4
HitmanPro.Alert protects this application against exploit attacks.

Choose exploit to execute

These exploit tests allow you to check your pc's security posture.

Run Windows Calculator
Not an exploit

Stack Pivot
Points stack pointer to heap allocated memory: XCHG ESP, EAX

Stack Exec
Marks the stack executable: PAGE_EXECUTE_READWRITE

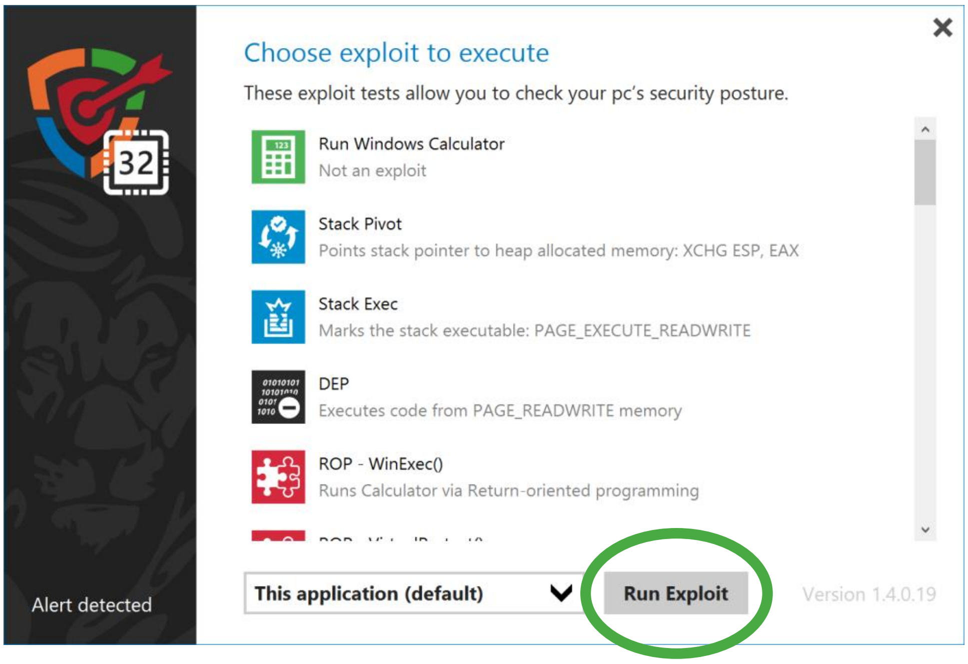
DEP
Executes code from PAGE_READWRITE memory

ROP - WinExec()
Runs Calculator via Return-oriented programming

Figure 2: Test selection panel

Important: The **Alert detected** indicator does not reflect the status of enabled exploit mitigation features. If e.g. all exploit mitigations are disabled in HitmanPro.Alert, the Exploit Test Tool will still indicate **Alert detected**. The indicator only reflects the presence of the HitmanPro.Alert DLL.

To start a specific test, select the test by clicking on it and then press the **Run Exploit** button to execute the exploit.


Choose exploit to execute

These exploit tests allow you to check your pc's security posture.

Run Windows Calculator
Not an exploit

Stack Pivot
Points stack pointer to heap allocated memory: XCHG ESP, EAX

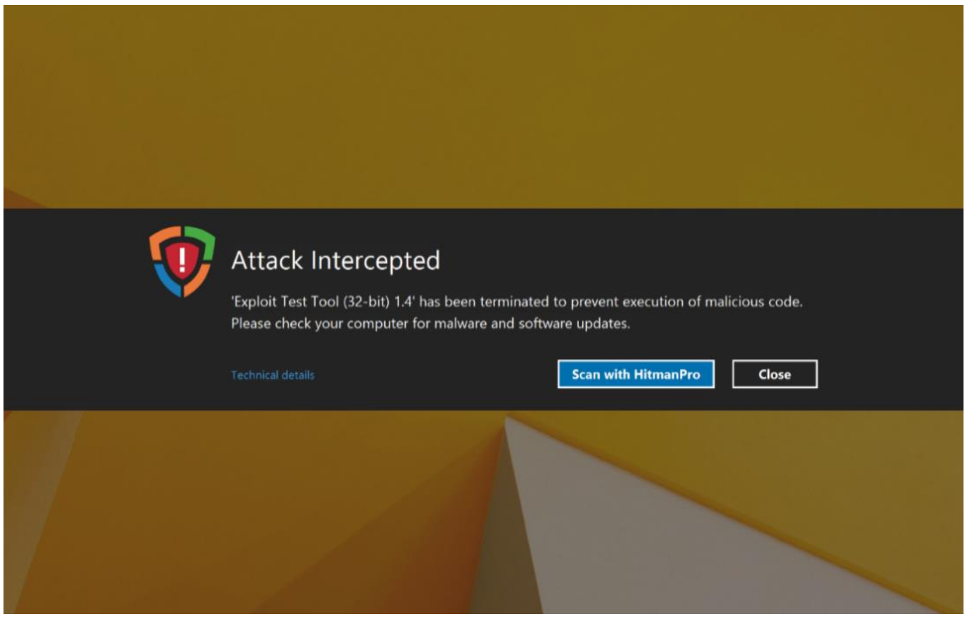
Stack Exec
Marks the stack executable: PAGE_EXECUTE_READWRITE

DEP
Executes code from PAGE_READWRITE memory

ROP - WinExec()
Runs Calculator via Return-oriented programming

Figure 3: Starting an exploit test

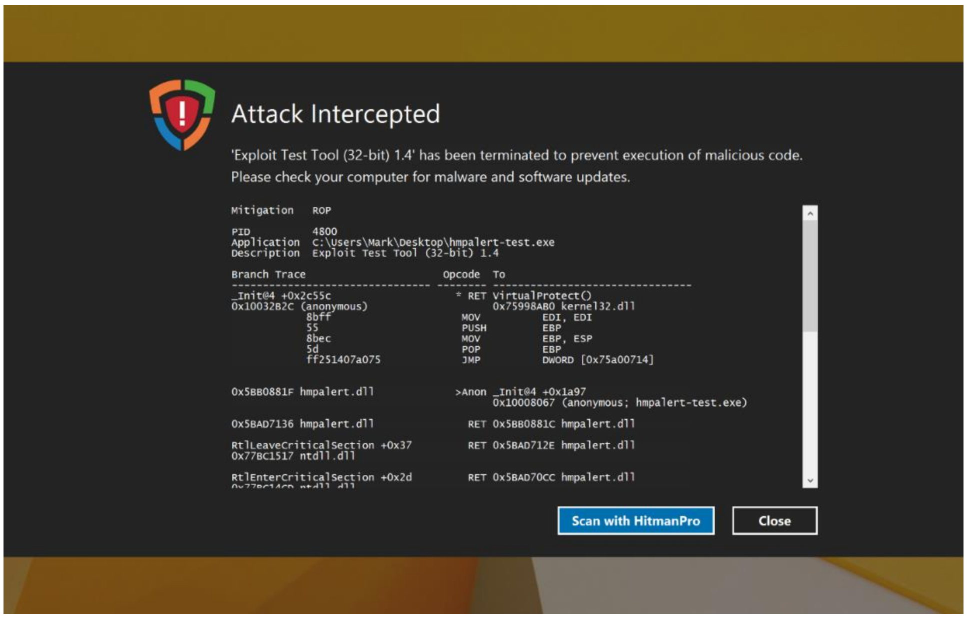
If an exploit technique is successfully blocked by HitmanPro.Alert, you will see an **Attack Intercepted** message as shown in Figure 4.


Attack Intercepted

'Exploit Test Tool (32-bit) 1.4' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

Figure 4: Attack Intercepted message

If you click on the blue text **Technical details**, detailed information about the reason why HitmanPro.Alert generated the intercept message is revealed. A report as shown in Figure 5 will be presented.


Attack Intercepted

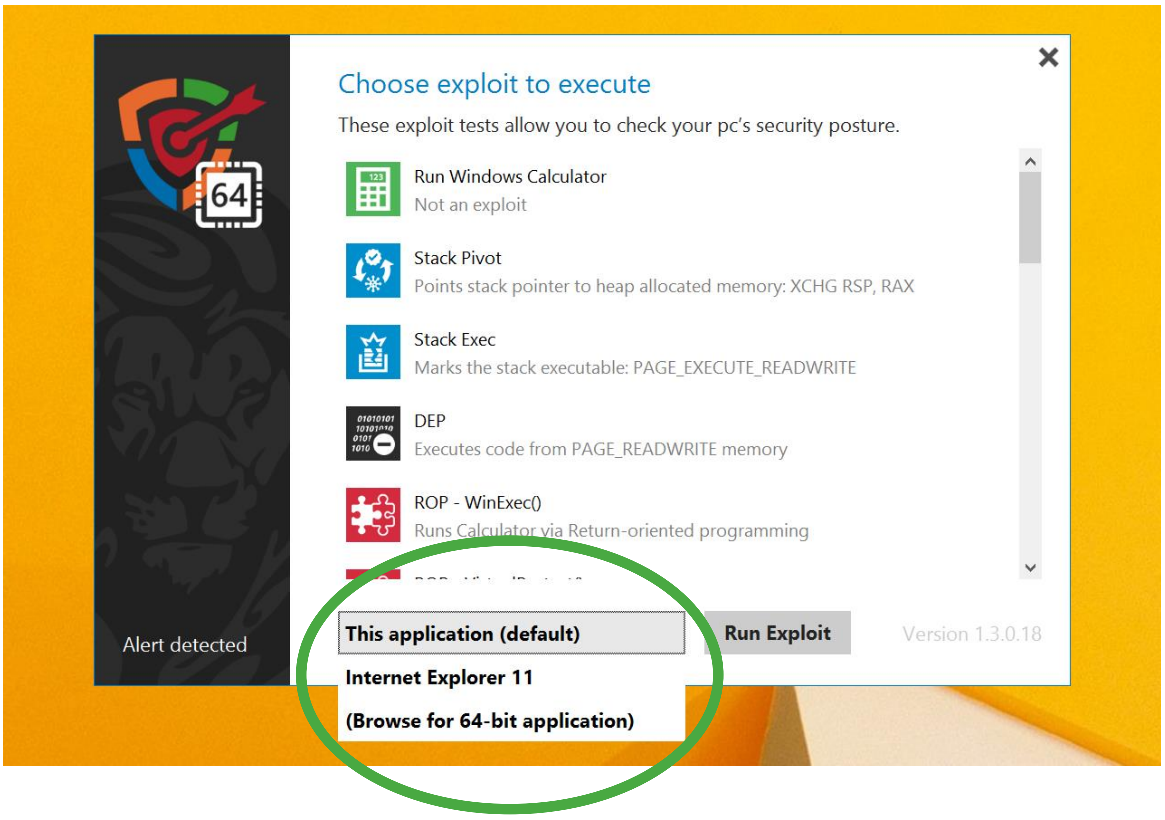
'Exploit Test Tool (32-bit) 1.4' has been terminated to prevent execution of malicious code.
Please check your computer for malware and software updates.

```
Mitigation   ROP

PID          4800
Application  C:\Users\Mark\Desktop\hmpalert-test.exe
Description  Exploit Test Tool (32-bit) 1.4

Branch Trace                      Opcode  To
--------------------------------- -------- --------------------------------
_Init@4 +0x2c55c                     * RET VirtualProtect()
0x10032B2C (anonymous)                     0x75998AB0 kernel32.dll
            8bff                    MOV            EDI, EDI
            55                      PUSH           EBP
            8bec                    MOV            EBP, ESP
            5d                      POP            EBP
            ff251407a075            JMP            DWORD [0x75a00714]


0x5BB0881F hmpalert.dll              >Anon _Init@4 +0x1a97
                                           0x10008067 (anonymous; hmpalert-test.exe)

0x5BAD7136 hmpalert.dll                RET 0x5BB0881C hmpalert.dll

RtlLeaveCriticalSection +0x37          RET 0x5BAD712E hmpalert.dll
0x77BC1517 ntdll.dll

RtlEnterCriticalSection +0x2d          RET 0x5BAD70CC hmpalert.dll
```

Figure 5: Technical details report

1.4 Detonating exploit tests in other applications

By default, the Exploit Test Tool detonates its exploit tests in its own process. As of version 1.2 of the Exploit Test Tool you can instruct it to detonate an exploit test in a different application, e.g. Internet Explorer. This allows you to check the overall security posture of your pc, as triggering an exploit should at least trigger on of the security applications. Click on the "This application" list to select a different application or browse for an application of your choosing.



Choose exploit to execute

These exploit tests allow you to check your pc's security posture.

Run Windows Calculator
Not an exploit

Stack Pivot
Points stack pointer to heap allocated memory: XCHG RSP, RAX

Stack Exec
Marks the stack executable: PAGE_EXECUTE_READWRITE

DEP
Executes code from PAGE_READWRITE memory

ROP - WinExec()
Runs Calculator via Return-oriented programming

When you use the 64-bit version of the Exploit Test Tool on 64-bit Windows, and you'd like to detonate an exploit test in an application of your own choosing, make sure that you select a 64-bit application. I.e. you cannot select e.g. **WINWORD.EXE** from inside the **C:\Program Files (x86)\** folder as that is a 32-bit application. The Exploit Test Tool will warn you about this. If you want to detonate an exploit test in e.g. the 32-bit version of Microsoft Word, use the 32-bit Exploit Test Tool.

If your anti-virus, exploit mitigation or anti-exploit software has a feature to shield custom applications, you can add the **hmpalert-test.exe** and **hmpalert64-test.exe** executables to the list of protected applications. This way you can also test the abilities of this other security software without abusing a third-party application.

2 Background information

This section provides some background information about exploits and the different technologies that are present to counter exploitation of software vulnerabilities.

2.1 Blended attacks

Software exploits, commonly used by cybercriminals and nation-state hackers, are combined with computer viruses resulting in complex so-called ‘blended attacks’, which go beyond the general scope of antivirus software. To optimize success, some attackers exploit multiple vulnerabilities through several different attack vectors to silently deliver (often never before seen) malware.

Web browsers are a particular target because of their widespread distribution and usage. Attackers can also send e-mail attachments that exploit vulnerabilities in the application opening the attachment. Typically, a use-after-free or buffer overflow vulnerability in these applications can result in a call to a sensitive system function, possibly a critical function that the application was never intended to use.

2.2 Return-Oriented Programming (ROP)

Thanks to the massive adoption of Data Execution Prevention (DEP) technology, exploitation of buffer overflows through code injection is difficult; to facilitate DEP most modern processor architectures have a MMU feature called XD (eXecute Disable) or No eXecute (NX). DEP forces an application to terminate when it attempts to execute (foreign) code placed in memory areas marked for data only.

As a result, attackers now leverage existing code in the process image to compromise an application. Using existing code has been generalized in a technique called return-oriented programming (ROP), where short snippets of code – called gadgets – are chained together to introduce a new, not initially intended, control-flow. Put another way, return-oriented programming provides a fully functional "language" that an attacker can use to construct malware (shellcode) by borrowing instructions from legitimate applications running on the computer.

2.3 ASLR

Defenses against ROP exploits are based on randomizing the process address layout so that attackers do not know where application code is mapped in the address space. Address Space Layout Randomization (ASLR) is a technology that achieves this. It maps processes (EXEs and DLLs) in random addresses each time an application is started. Thanks to ASLR, attackers can no longer accurately predict the location of instructions that might be useful in gadgets to construct malware. Therefore, attackers now attempt to control or discover the location of certain memory regions through the use of an address space information disclosure. When you know the location of one known function, the position of all others can be inferred and a ROP attack can be constructed.

2.4 Control-Flow Integrity (CFI)

HitmanPro.Alert version 3 introduces control-flow integrity (CFI) – a technique to prevent flow of control not intended by the original application, without requiring the source code or debug symbols of the protected applications. Without requiring prior knowledge of the attack, code or malware involved, it effectively stops attackers that combine short pieces of benign code, already present in a system, for a malicious purpose (a ROP attack).

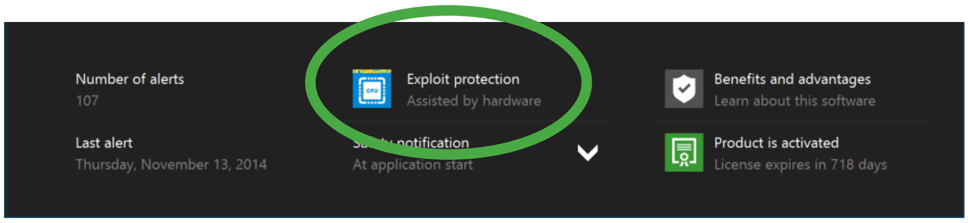
2.5 Hardware-assisted Control-Flow Integrity

Whenever a critical function is triggered, HitmanPro.Alert checks whether it is a benign system call or part of a ROP exploit. It can even leverage special hardware registers inside Intel® processors to assist in the detection of ROP attacks. Besides a performance advantage, employing hardware traced records has a security benefit over software stack-based approaches. Stack-based solutions, like Microsoft EMET, rely on stack data, which is (especially in case of a ROP attack) in control of the attacker, who in turn can affect or control the defender as well.

2.5.1 Supported processors

HitmanPro.Alert will automatically employ Intel MSR hardware registers when it detects an Intel® Core™ i3, i5 or i7 processor (CPU). HitmanPro.Alert supports all Core i3, i5 and i7 processors, which includes those with the codenames Nehalem, Westmere, Sandy Bridge, Ivy Bridge, Haswell, Broadwell and Skylake.

To determine if CFI supports your specific CPU, simply open HitmanPro.Alert. If the CPU badge is displayed, your processor is supported:



Number of alerts
107

Last alert
Thursday, November 13, 2014

Exploit protection
Assisted by hardware

[illegible] notification
At application start

Benefits and advantages
Learn about this software

Product is activated
License expires in 718 days

Figure 6: Your processor is supported for hardware-assisted exploit protection

HitmanPro.Alert will automatically fallback on software-only stack-based Control-Flow Integrity checks if your computer does not have a hardware-assisted CFI supported processor.

Important: In e.g. VMware and VirtualBox the hardware registers are not available, as these registers are not virtualized by VMware and other virtualization software. Therefore, reviewers cannot not use a virtual environment if they want to test the full anti-exploit capabilities of HitmanPro.Alert 3.

3 Details of individual tests

In this section the details of the individual tests of the Exploit Test Tool will be presented.

3.1 Run Windows Calculator (not an exploit)

Exploit Test Tool: 32 64

This test is not an exploit test, but a test to verify if the tool is able to start the Windows Calculator (%systemroot%\system32\calc.exe).

Important: If the Windows Calculator cannot be started, none of the exploit techniques in the Exploit Test Tool can be reliably tested.

Most exploit tests will start the Windows Calculator as proof of a successful exploit of a software vulnerability. Note that if the Windows Calculator can be started via an exploit, an attacker could have started some malicious code instead, using the exploit technique and a software vulnerability.

3.2 Stack Pivot

Exploit Test Tool: 32 64

A common method for an attacker to control program execution involves creating a 'fake stack' using attacker-specified values. When the attacker tricks the victim computer into using a fake stack, the attacker can control the program execution.

This specific test allocates a block of 64KB of memory on the heap and copies shellcode to that memory. Then it switches the stack pointer to the end of that memory block and starts the shellcode, which gives the shellcode a comfortably nice large stack for its operations.

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.3 Stack Exec

Exploit Test Tool: 32 64

This test makes a piece of the stack memory executable via a call to VirtualProtect(). Then it copies shellcode to that part of the stack and jumps to the start of that shellcode. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.4 DEP

Exploit Test Tool:

This test allocates a piece of non-executable memory on the heap and copies shellcode to start calc.exe to this memory. Then it jumps to that shellcode. This will trigger a DEP exception which, in case of HitmanPro.Alert, will be intercepted.

3.5 Unpivot Stack

Exploit Test Tool:

Important: This test is only intercepted by HitmanPro.Alert if the main processor (CPU) is an Intel® Core™ i3, i5 or i7 CPU (see chapter 2.5 about Hardware-assisted Control-Flow Integrity, page 10).

This test allocates a piece of non-executable memory on the heap, where shellcode is copied to. Then it dynamically creates a ROP chain on the heap, from gadgets that are searched in mshtml.dll. The intention of this ROP chain is to copy a second-stage ROP chain to the stack of the currently running thread. This second-stage ROP chain is using gadgets to make the shellcode (to start calc.exe) executable and start that shellcode, once it has become executable. Since the second-stage ROP chain is executing from the original stack, the stack will not be pivoted (i.e. un-pivoted again). To call VirtualProtect() from the second-stage ROP chain a 'call-gadget' (# CALL EAX # RET), located in a legitimate DLL, is used so that the call is seemingly originating from a legitimate source.

If the test is successful, calc.exe will be started and the Exploit Test Tool terminates via a call to ExitProcess(). Since the ROP chain is created from dynamically searched gadgets in mshtml.dll it can occur that you will see a message like 'Could not find enough gadgets: stage N'. In that case your version of mshtml.dll does not contain the required ROP-gadgets. This is caused by the fact that mshtml.dll is updated/changed regularly and stored on your computer via a Windows Update.

3.6 ROP – WinExec()

Exploit Test Tool:

This test creates a return-oriented programming (ROP) chain that calls WinExec() in order to start calc.exe. After that, the ROP chain jumps to ExitProcess(), so the Exploit Test Tool will end.

3.7 ROP – VirtualProtect()

Exploit Test Tool: 32 64

This test allocates a piece of non-executable memory on the heap, where shellcode is copied to. Then it creates a ROP chain that performs the following steps:

- Call VirtualProtect() to make the shellcode executable
- Jump into the shellcode, which is now executable

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.8 ROP – CALL preceded VirtualProtect()

Exploit Test Tool: 32

Important: This test is only intercepted by HitmanPro.Alert if the main processor (CPU) is an Intel® Core™ i3, i5 or i7 CPU (see chapter 2.5 about Hardware-assisted Control-Flow Integrity, page 10).

This test allocates a piece of non-executable memory on the heap, where shellcode is copied to. Then it searches for a call-gadget in legitimate DLLs. The address of this call-gadget is used to create a ROP chain that performs the following steps:

- Call VirtualProtect() to make the shellcode executable
- Return to the call-gadget
- Jump into the shellcode, which is now executable

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.9 ROP – NtProtectVirtualMemory()

Exploit Test Tool: 32 64

This test allocates a piece of non-executable memory on the heap, where shellcode is copied to. Then it creates a ROP chain that performs the following steps:

- Call NtProtectVirtualMemory() to make the shellcode executable
- Jump into the shellcode, which is now executable

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.10 ROP – Wow64 bypass

Exploit Test Tool: 32

Important: Due to the nature of this exploit test, a 64-bit version of Windows is required to run this exploit technique from the 32-bit Exploit Test Tool.

While most exploit attacks target vulnerabilities in pure 32-bit applications, the fact is that a significant portion of 32-bit software is nowadays running on 64-bit operating systems. A technique disclosed by Duo Security[1] bypasses payload/shellcode execution and ROP-related mitigations provided by security software by using the Wow64[2] compatibility layer provided in 64-bit versions of Windows. Even though security software can complicate exploitation techniques in true 32-bit and 64-bit applications, for some anti-exploit tools the mitigations are less effective under the Wow64 subsystem.

This specific test attempts to bypass the 32-bit ntdll.dll by directly calling the Wow64 marshalling layer performing NtProtectVirtualMemory() in 64-bit ntdll.dll. When successful, the Windows Calculator will appear.

3.11 ROP – Exploit Wow64

Exploit Test Tool: 32

Important: Due to the nature of this exploit test, a 64-bit version of Windows is required to run this exploit technique from the 32-bit Exploit Test Tool.

Like the test described at 3.10 ROP – Wow64 bypass, this technique attempts to bypass security checks of security software by directly accessing the Wow64 compatibility layer.

This specific test performs the exact technique described by Duo Security in their document[3] "WoW64 and So Can You – Bypassing EMET With a Single Instruction". The 32-bit Exploit Test Tool will employ ROP gadgets to transition from 32-bit mode directly into 64-bit NtProtectVirtualMemory().

[1] https://www.duosecurity.com/blog/wow64-and-so-can-you

[2] https://en.wikipedia.org/wiki/WoW64

[3] https://www.duosecurity.com/static/pdf/WoW64-Bypassing-EMET.pdf

3.12 ROP – system() in msvcrt

Exploit Test Tool:

This test creates a return-oriented programming (ROP) chain that calls system() in msvcrt.dll in order to start calc.exe. When calc.exe has been started, the Exploit Test Tool will crash – this is expected.

A similar ROP technique was e.g. used in a zero-day attack on CVE-2013-1347[1].

Note: This exploit test is currently only available when msvcrt.dll is older than version 7.0.9600.17415.

3.13 ROP – VirtualProtect() via CALL gadget

Exploit Test Tool:

Important: This test is only intercepted by HitmanPro.Alert if the main processor (CPU) is an Intel® Core™ i3, i5 or i7 CPU (see chapter 2.5 about Hardware-assisted Control-Flow Integrity, page 10).

This test is based on the whitepaper **Bypassing EMET 4.1**, by Jared DeMott from Bromium Labs[2].

First, it allocates a piece of non-executable memory on the heap, where shellcode is copied to. Then it locates a call to VirtualProtect() in legitimate code. The address of this call is used in the ROP chain. Then it creates a ROP chain that performs the following steps:

- Jump to the legitimate code where a call to VirtualProtect() is located
- Jump into the shellcode, which is now executable

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.14 ROP – WinExec() via anti-detour

Exploit Test Tool:

This test determines if WinExec() contains anti-detour instructions. If it does, a piece of shellcode is created that executes the original instructions of WinExec() first, and then jumps over the anti-detour instructions into WinExec(). This shellcode is started via a return-oriented programming (ROP) chain, effectively calling WinExec() in order to start calc.exe. After that, the ROP chain jumps to ExitProcess(), so the Exploit Test Tool will end.

[1] http://www.fireeye.com/blog/technical/cyber-exploits/2013/05/ie-zero-day-is-used-in-dol-watering-hole-attack.html

[2] http://labs.bromium.com/2014/02/24/bypassing-emet-4-1/

3.15 IAT Filtering

Exploit Test Tool:

The IAT Filtering mitigation is designed to block an attack when the attacker uses an address that was snooped from an Import Address Table (IAT).

This test allocates a piece of memory on the heap and copies shellcode to start calc.exe to this memory. Then it fetches the address of the function VirtualProtect() from the Import Address Table (IAT) of a DLL that is present in the process. This function pointer is used to call VirtualProtect() to make the shellcode on the heap executable. Then the test jumps to the shellcode. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.16 NULL Page

Exploit Test Tool:

This test allocates the first page of the virtual memory of the process and copies shellcode into that page. Then it jumps to the shellcode, thus simulating the (faulty) situation where a pointer is used to call a function and that pointer is NULL because of some unexpected condition. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.17 SEHOP

Exploit Test Tool:

HitmanPro.Alert includes Structured Exception Handler Overwrite Protection (SEHOP), which currently protects against the most common technique for exploiting stack overflows in Windows.

The SEHOP test in the Test Tool creates a buffer overflow, which causes the overwriting of the Windows Structured Exception Handler (SEH) record that is located on the stack and also puts our shellcode on the stack. It overwrites the SEH record with a pointer to a known 'POP-POP-RET' sequence and a short jump instruction to the shellcode that is now on the stack. Then it generates an exception which causes the exception handler to traverse the SEH chain and in doing so executing our shellcode on the stack. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

Note: To let this test successfully start calc.exe, both the mitigations **Stack Exec** and **SEHOP** must be disabled.

3.18 Heap Spray 1 – Single byte NOP-sled

Exploit Test Tool:

Heap spraying[1] is a technique used in exploits to facilitate arbitrary code execution. The Dynamic Heap Spray protection in HitmanPro.Alert is a generic solution aimed to detect heap spray behavior. In addition, Dynamic Heap Spray also pre-allocates common heap spray addresses that are often used by inattentive attackers or Metasploit[2].

This specific Heap Spray test in our Exploit Test Tool allocates 256MB of memory on the heap. Like a motivated attacker, the Exploit Test Tool allocates this region at an address not pre-allocated by anti-exploit software. The Exploit Test Tool stamps this 256MB of memory from begin to end with a 64KB template. This template consists for the largest part of a single-byte NOP-sled[3], followed by a small shellcode. Once the complete 256MB of memory is filled, the region is made executable via a call to VirtualProtect() and the test jumps into a NOP-sled in that memory range which leads to the start of the shellcode. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

Note: To let this test start calc.exe, **Control-Flow Integrity** and **Dynamic Heap Spray** must be disabled.

3.19 Heap Spray 2 – Polymorphic NOP-sled

Exploit Test Tool:

This test allocates 256MB of memory on the heap. Like a motivated attacker, this region is allocated at an address not pre-allocated by anti-exploit software. The Exploit Test Tool stamps this 256MB of memory from begin to end with a 64KB template. This template consists for the largest part of a multi-byte NOP-sled (polymorphic NOP-sled), followed by a small shellcode. Once the complete 256MB of memory is filled, the region is made executable via a call to VirtualProtect() and the test jumps into a NOP-sled in that memory range which leads to the start of the shellcode.

When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

Note: In order to let this test successfully start calc.exe, both the mitigations **Control-Flow Integrity** and **Dynamic Heap Spray** must be disabled.

[1] http://en.wikipedia.org/wiki/Heap_spraying

[2] http://www.rapid7.com/products/metasploit/index.jsp

[3] http://en.wikipedia.org/wiki/NOP_sled

3.20 Heap Spray 3 – ActionScript

Exploit Test Tool: 32 64

This test allocates 16MB of memory on the heap. Like a motivated attacker, this region is allocated at an address not pre-allocated by anti-exploit software. The Exploit Test Tool stamps this memory from begin to end with a 1024 byte template. This template consists of an Adobe ActionScript vector of UInts (used by Adobe Flash), simulating the actions performed by e.g. CVE-2014-0322[1] and CVE-2014-1776[2]. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

3.21 Heap Spray 4 – JavaScript

Exploit Test Tool: 32 64

This test allocates 16MB of memory on the heap. Like a motivated attacker, this region is allocated at an address not pre-allocated by anti-exploit software. The Exploit Test Tool stamps this memory from begin to end with a 64KB template. This template consists of a JavaScript ArrayBuffer object, simulating the action as is performed by e.g. CVE-2014-1512[3]. When calc.exe has been started, the Exploit Test Tool terminates via a call to ExitProcess().

[1] http://www.fireeye.com/blog/technical/cyber-exploits/2014/02/operation-snowman-deputydog-actor-compromises-us-veterans-of-foreign-wars-website.html

[2] http://www.fireeye.com/blog/uncategorized/2014/04/new-zero-day-exploit-targeting-internet-explorer-versions-9-through-11-identified-in-targeted-attacks.html

[3] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-1512

3.22 Anti-VM – VMware

Exploit Test Tool:

When malware with Anti-VM techniques (so-called 'vm-aware' or 'sandbox-aware' malware) detects a virtual machine, it will refuse to run because it assumes it is running inside a malware analyst's automated sandbox or honeypot – the malware will simply self-terminate to conceal its hostile intentions. Like malware, the Anti-VM test of the Exploit Test Tool will probe for a guest-to-host communication port used by virtualization software.

On a normal non-virtual machine, probing this port will trigger an alert from HitmanPro.Alert. The process that probed the port – in this case the Exploit Test Tool – will be terminated by HitmanPro.Alert and an entry about this event is written to the Windows Event Log. If HitmanPro.Alert is not installed (or Active Vaccination is disabled) the Windows Calculator will appear. On a guest machine inside VMware, HitmanPro.Alert will not block the probe and the Exploit Test Tool will exit.

Note: The default setting of HitmanPro.Alert's Vaccination feature is set to Passive vaccination. To block this specific test, you must set Vaccination to **Active vaccination**: Settings > Advanced interface > Risk reduction > Vaccination > Active vaccination

3.23 Anti-VM – Virtual PC

Exploit Test Tool:

Similar to the Anti-VM – VMware test.

On a guest machine inside Microsoft Virtual PC[1], HitmanPro.Alert will not block the probe and the Exploit Test Tool will exit.

Note: The default setting of HitmanPro.Alert's Vaccination feature is set to Passive vaccination. To block this specific test, you must set Vaccination to **Active vaccination**: Settings > Advanced interface > Risk reduction > Vaccination > Active vaccination

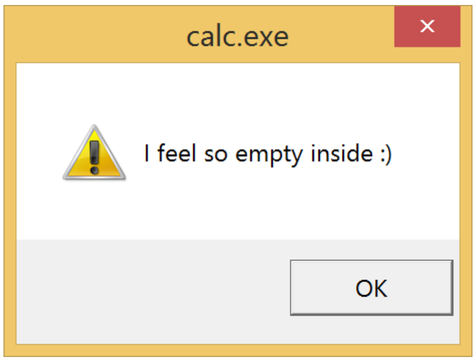
[1] http://en.wikipedia.org/wiki/Windows_Virtual_PC

3.24 Hollow Process

Exploit Test Tool: 🎃 32

Process hollowing (or process replacement) is a technique in which a legitimate process is loaded on the system solely to act as a container for hostile code. Process hollowing is often used by attackers who seek to hide the presence of their malicious process, e.g. a Remote Access Trojan (RAT).

This specific test works on **32-bit operating systems only** and starts the Windows Calculator calc.exe from your Windows installation. It then replaces the in-memory contents of that process with a small program that displays the following message box.


calc.exe

I feel so empty inside :)

Figure 7: Calc.exe process abused to run arbitrary code

Note: **Control-Flow Integrity (CFI)** (Exploit mitigations) and **Process Protection** (Risk reductions) must be disabled in HitmanPro.Alert to let this test successfully start and manipulate the binary contents of the calc.exe process.

3.25 Load Library

Exploit Test Tool: 32 64

This test tries to download a library file (DLL file) over a UNC network path, a common technique used by attackers.

3.26 URLMon

Exploit Test Tool: ? 32 64

This test allocates a piece of executable memory on the heap, where shellcode is placed. This shellcode is then started.

The shellcode then calls the system function URLDownloadToFileA to retrieve the remote file http://test.hitmanpro.com/dummy.dll

When the download was successful it will start calc.exe. In an attack scenario the download is a malicious payload to infect the computer.

3.27 URLMon 2 – Rundll32

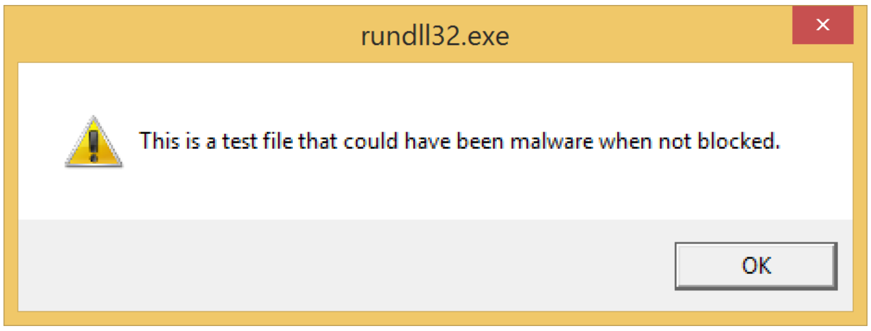
Exploit Test Tool: 32

This test creates a ROP chain that performs the following steps:

- Call the system function URLDownloadToFileA to retrieve the remote file: http://test.hitmanpro.com/dummy.dll

- Load the downloaded dummy.dll using **Rundll32**

If the start of the ROP chain is successful you will see a message box as shown below. This message is generated by the downloaded DLL that could successfully execute.


rundll32.exe

This is a test file that could have been malware when not blocked.

In an attack scenario the download is a malicious payload to infect the computer.

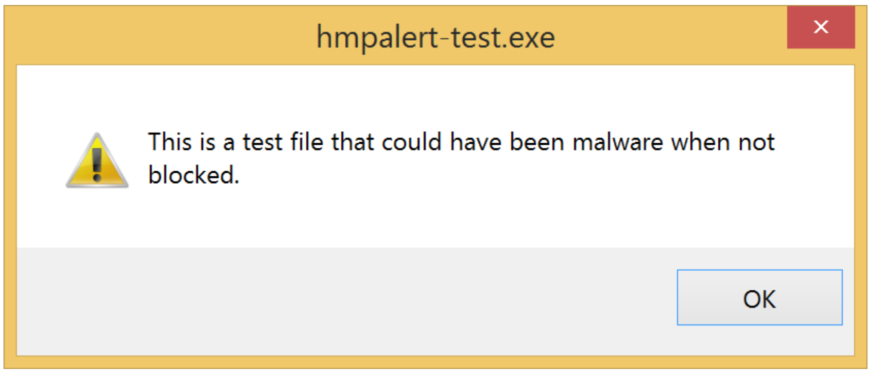
3.28 URLMon 3 – LoadLibraryA

Exploit Test Tool:

This test creates a ROP chain that performs the following steps:

- Call the system function URLDownloadToFileA to retrieve the remote file: http://test.hitmanpro.com/dummy.dll
- Load the downloaded dummy.dll using the system function **LoadLibraryA**

If the start of the ROP chain is successful you will see a message box as shown below. This message is generated by the downloaded DLL that could successfully execute.


hmpalert-test.exe

This is a test file that could have been malware when not blocked.

In an attack scenario the download is a malicious payload to infect the computer.

3.29 Lockdown 1

Exploit Test Tool:

When an application contains a logic flaw vulnerability, or when attackers successfully bypass sandbox, memory and other code mitigations, they are able introduce new executables, run malicious code. Another example are specially crafted macros[1] in Microsoft Word documents causing unexpected behavior like hoisting in malware. Or the CVE-2014-4114/CVE-2014-6352[2] vulnerability which allows remote code execution through a specially crafted document with a malicious OLE object.

This test creates an executable by making a copy of calc.exe and tries to execute this copy, a task not normal for e.g. Office and Media applications.

3.30 Lockdown 2

Exploit Test Tool:

This test creates an executable by making a copy of calc.exe, renames it and tries to execute this renamed copy.

3.31 Webcam test (not an exploit)

Exploit Test Tool:

This test is not an exploit test, but a way to verify if the Webcam Notifier functionality of HitmanPro.Alert is operating correctly. When this test is started, a window will appear which shows the captured video from your webcam. If the Webcam Notifier function is enabled in HitmanPro.Alert, an alert will be triggered which gives you the option to Allow or Deny access to the webcam.

[1] http://www.theregister.co.uk/2014/07/08/macro_viruses_return_from_the_dead/

[2] http://www.theregister.co.uk/2014/10/22/powerpoint_attacks_exploit_ms_0day/

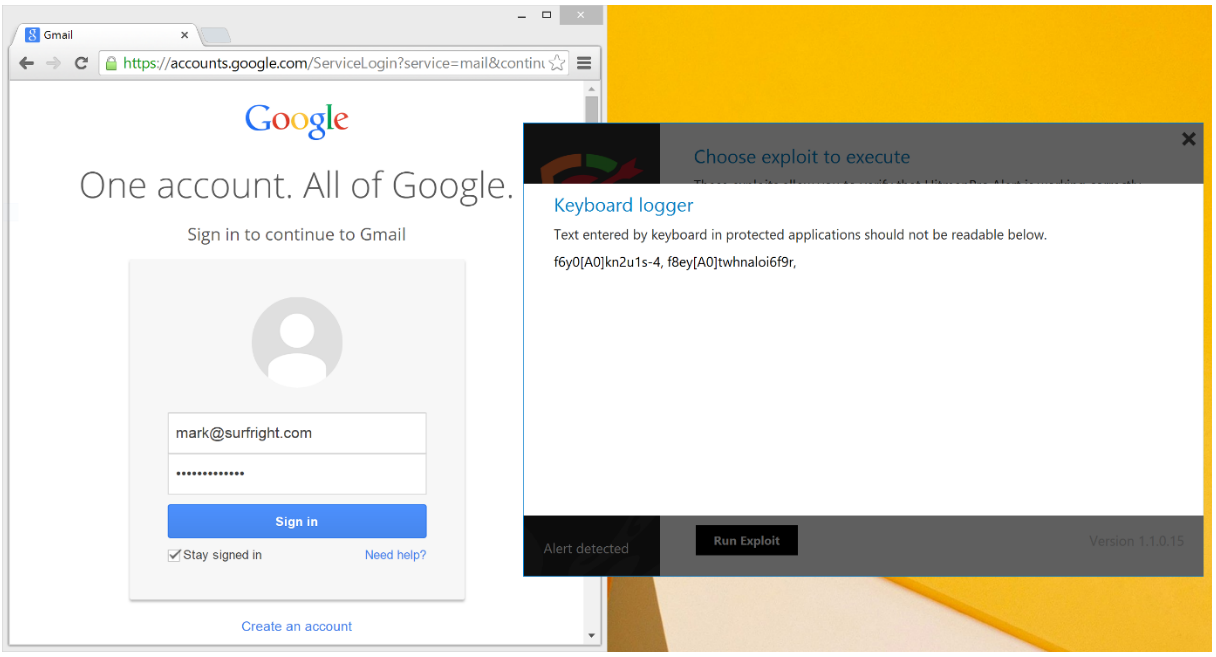
3.32 Keyboard logger (not an exploit)

Exploit Test Tool: 32 64

This test is not an exploit test, but a way to verify if the Keystroke Encryption functionality of HitmanPro.Alert is operating correctly. Note that the Keystroke Encryption functionality in HitmanPro.Alert is applied to web browsers only.

If this test is started, you will see a box overlaying the Exploit Test Tool window. In this text box you can see the keystrokes an attacker would see when he tries to intercept your keystrokes. To see what the effect is, you can open a web browser and type some text, for example on a web page where login credentials are requested.

When the **Keystroke Encryption** function in HitmanPro.Alert is enabled, you will see random characters in the text box of the Test Tool. See Figure 8 for an example.


Gmail
https://accounts.google.com/ServiceLogin?service=mail&continu
Google
One account. All of Google.
Sign in to continue to Gmail
mark@surfright.com
Sign in
Stay signed in
Need help?
Create an account
Choose exploit to execute
Alert detected
Run Exploit
Version 1.1.0.15

Keyboard logger

Text entered by keyboard in protected applications should not be readable below.

f6y0[A0]kn2u1s-4, f8ey[A0]twhnaloi6f9r,

Figure 8: Keystroke Encryption enabled

Revision History

Version	Authors	Remarks	Date
1.6	EE, ML	Added ROP – Wow64 bypass test Added ROP – Exploit Wow64 test	2015-11-16
1.5	EE, ML	Added clarification of testing individual exploit techniques and the use of dynamic ROP gadgets Added Unpivot Stack Updated Heap Spray tests	2015-3-25
1.4	EE, ML	Added ROP – CALL preceded VirtualProtect(), DEP, IAT Filtering and Lockdown tests	2014-12-4
1.3	ML	Added Anti-VM (Virtual PC) test	2014-10-10
1.2	ML	Added Anti-VM test	2014-10-2
1.1	EE, ML	Updated paragraph 2.5 with Nehalem and Westmere Added ROP – WinExec() via anti-detour Added ROP – system() in msvcrt	2014-9-1
1.0	EE, ML	Initial version	2014-7-7