The challenge of creating a comprehensive security solution for grey-hat tools requires subtlety, detail and forethought.

Protecting against Cobalt Strike

Preventative and Threat Hunting Methodology

Black Cell Hungary Ltd.

Table of Contents

Protecting Against Cobalt Strike

Executive Summary

Recently many large corporations have been struck by cyber-attacks, many of which were assumed or even proven to be perpetrated by the help of an offensive tool named Cobalt Strike. The protective measures developed by Black Cell Hungary Ltd. and described in this document are effective at preventing, detecting and remediating such attacks. Cobalt Strike has 38 distinct capabilities, each of which requires the implementation of specific logical, physical and administrative safeguards. These safeguards need to be applied though purpose-built security tools and devices.

Our methodology puts emphasis on the prevention of attacks. There are many easily detectable indicators of an impending attack, and through early detection and an appropriate response, attacks can be stopped before any damage has been done. Of course, by the nature of cyber attacks 100% protection can never be guaranteed, as such we have developed detailed methods for detecting, identifying and mitigating successful attacks. Our methodology is also highly effective against myriads of other cyber-attacks, besides those perpetrated with the help of Cobalt strike.

About Cobalt Strike

Cobalt Strike is a commercial, full featured penetration testing tool, that is designed to simulate the post-exploitation activities of malicious actors. Its interactive capabilities cover almost the entire range of known post-exploit activities, all in a single integrated tool. Besides its own capabilities it also facilitates the use of other offensive tools, such as Metasploit and Mimikatz in order to broaden its capabilities. Recently this tools has been used increasingly for unauthorized, malicious uses and opening the door for many cyber-attacks.

Technical Capabilities

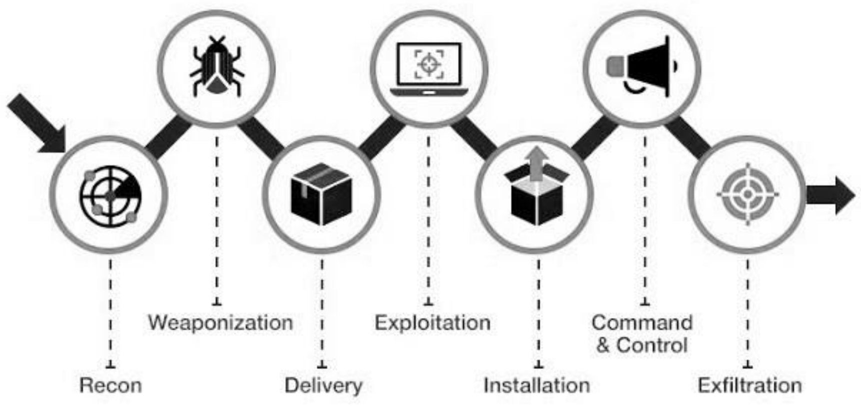
1. Access Token Manipulation
2. BITS Jobs
3. Bypass User Account Control
4. Command-Line Interface
5. Commonly Used Port
6. Component Object Model and Distributed COM
7. Connection Proxy

8. Credential Dumping
9. Custom Command and Control Protocol
10. Data from Local System
11. Execution through API
12. Exploitation for Privilege Escalation
13. Indicator Removal from Tools
14. Input Capture
15. Man in the Browser
16. Multiband Communication
17. Network Service Scanning
18. Network Share Discovery
19. New Service
20. Parent PID Spoofing
21. Pass the Hash
22. PowerShell
23. Process Discovery
24. Process Hollowing
25. Process Injection
26. Remote Desktop Protocol
27. Remote Services
28. Remote System Discovery
29. Scheduled Transfer
30. Screen Capture
31. Scripting
32. Service Execution
33. Standard Application Layer Protocol
34. Timestomp
35. Windows Admin Shares
36. Valid Accounts
37. Windows Management Instrumentation
38. Windows Remote Management

Identifying an Attack

Indicators

The key to identifying the indicators of this attack, is to detect its early stages and to deploy preventative and protective measures. The attack begins (after reconnaissance has completed) by sending a document containing malicious code to an individual, or group of individuals with the hope that someone will open it. When one of the recipients opens the document, malicious code is executed. For persistence purposes a beacon is installed by the malicious code which then begins to communicate with and take orders from a command and control (C2) server. A Cobalt Strike attack can thus be detected in the delivery phase of



the cyber kill chain.

Cyber Kill Chain © Mitre

The simplest yet most important protective procedure is to inform and educate employees of dangers of malicious documents and other cyber attack vectors, and to provide them with channels to communicate with cyber security professionals that can analyze suspicious documents before they are opened.

Most of Cobalt Strikes capabilities can be detected through sufficient logging and log analysis, although some others require additional specialized software or hardware devices. With log analysis, threats relating to command lines, PowerShell, services, user accounts, remote access and network scanning can be alleviated.

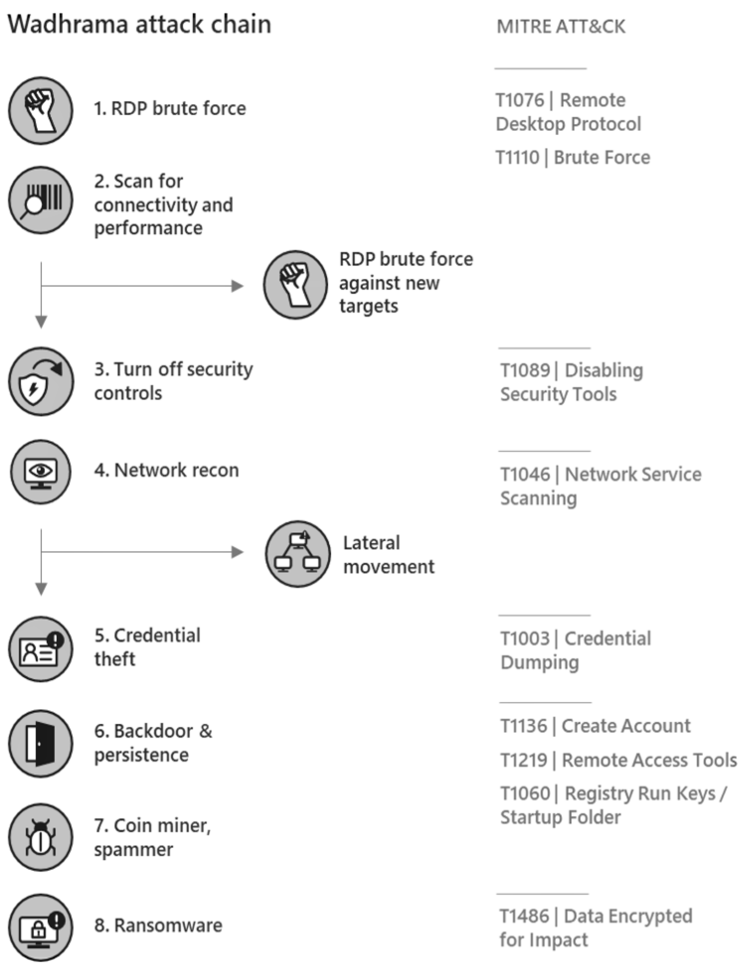
Cobalt Strike’s first phase starts with reconnaissance and the collection of organizational information. The attacker will use various network scanning utilities to detect network infrastructure components and related vulnerabilities. The scans are figuratively “loud” because the high volume of requests to deferent network components draws attention and is easy to detect. A SIEM system and sufficient logging is enough to detect and alert to these discovery activities. That said, these alerts in themselves won’t be able to identify Cobalt Strike attacks specifically, but will rather give IT security professionals sufficient time and information to prepare for a forthcoming attack.

Following the pre-exploit activities, the attacker will attempt to gain initial access and implant a beacon. This usually takes the form of a spear phishing campaign, where malicious documents are sent to various internal email addresses, in hopes of someone opening it. This campaign is one of the most easily detectable parts of a Cobalt Strike attack. When a large number of emails are received from an unknown sender in a short amount of time, then an alert is triggered, and an IT security professional’s investigation will quickly reveal the malicious content of the email. The list of recipients and more specifically the people that opened the attachment can be collected and the necessary remediation steps can be taken. Furthermore, the hash value of the attachments of all emails can be automatically checked against other known malicious files helping to automatically identify phishing campaigns.

The malicious macros employed by Cobalt Strike can also be easily detected. The malicious VBA scripts spawn new processes, that with the sufficient process creation logging, will trigger an alert in the SIEM system, even if the attacker has successfully disabled the antivirus software or clears the relevant log files after the fact.

The aforementioned basic logging combined with a SIEM system, makes successful Cobalt Strike attacks extraordinarily unlikely, in fact, with a few adjustments can prevent the most common attack vectors, such as the RDP brute force attack detailed in this image. Even if a successful attack takes place, there are still many protection methods that can warn us before significant damage can be done.

The beacon installed during a successful Cobalt Strike attack will have to communicate with a C2 server sooner or later. This communication will often be disguised via various obfuscation techniques.


Wadhrama attack chain
MITRE ATT&CK
1. RDP brute force
T1076 | Remote Desktop Protocol
T1110 | Brute Force
2. Scan for connectivity and performance
RDP brute force against new targets
3. Turn off security controls
T1089 | Disabling Security Tools
4. Network recon
T1046 | Network Service Scanning
Lateral movement
5. Credential theft
T1003 | Credential Dumping
6. Backdoor & persistence
T1136 | Create Account
T1219 | Remote Access Tools
T1060 | Registry Run Keys / Startup Folder
7. Coin miner, spammer
8. Ransomware
T1486 | Data Encrypted for Impact

© Microsoft

This concealed communication can be easily identified with a sufficient Next Generation Firewall (NGF) or Intrusion Detection System (IDS). These devices conduct deep analysis on the packets that form the communication flowing through them, looking for anomalies indicative of malicious activity.

The defensive methods described above make successful, undetected Cobalt Strike attacks near impossible. However, there are still as of yet unmentioned capabilities (with mostly local effects), that cannot be detected through common log collection and analysis techniques. These require advanced Endpoint Detection and Response (EDR) and/or vulnerability management tools.

Sufficient Logging and SIEM Deployment	Intrusion Detection System and/or Next Generation Firewall	Endpoint Detection and/or Vulnerability Management
• Network Service Scanning (ID: T1046) • Network Share Discovery (ID: T1135) • New Service (ID: T1050) • Command-Line Interface (ID: T1059) • Credential Dumping (ID: T1003) • Data from Local System (ID: T1005) • PowerShell (ID: T1086) • Process Discovery (ID: T1057) • Process Injection (ID: T1055) • Remote System Discovery (ID: T1018) • Scripting (ID: T1064) • Service Execution (ID: T1035) • Timestomp (ID: T1099) • Windows Admin Shares (ID: T1077) • Valid Accounts (ID: T1078) • Windows Management Instrumentation (ID: T1047)	• Remote Desktop Protocol (ID: T1076) • Remote Services (ID: T1021) • Scheduled Transfer (ID: T1029) • Standard Application Layer Protocol (ID: T1071) • Commonly Used Port (ID: T1043) • Connection Proxy (ID: T1090) • Custom Command and Control Protocol (ID: T1094) • Multiband Communication (ID: T1026) • Windows Remote Management (ID: T1028)	• Access Token Manipulation (ID: T1134) • BITS Jobs (ID: T1197) • Bypass User Account Control (ID: T1088) • Component Object Model and Distributed COM (ID: T1175) • Execution through API (ID: T1106) • Exploitation for Privilege Escalation (ID: T1068) • Indicator Removal from Tools (ID: T1066) • Input Capture (ID: T1056) • Man in the Browser (ID: T1185) • Parent PID Spoofing (ID: T1502) • Pass the Hash (ID: T1075) • Process Hollowing (ID: T1093) • Screen Capture (ID: T1113)

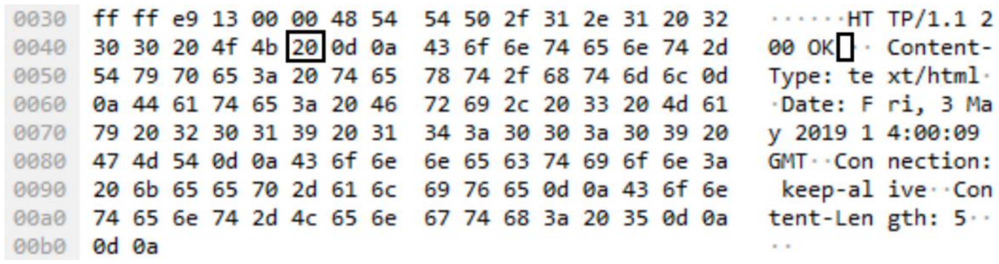
The various protective measures and the Cobalt Strike capabilities (with Mitre Att&ck reference) they cover.

The previously described detection methods provide robust, reliable protection against Cobalt Strike attacks, by detecting its pre- and post-exploit capabilities. There are some other detection methods that rely on detecting aspects specific to the framework, that Cobalt Strike provides, rather than the effects of its functions.

- Default certificate hash: Cobalt Strike's default certificate often remains unchanged. Thus, if we detect this certificate's hash on any device or in network communication, then we know that a Cobalt Strike attack has taken place.

- Default controller port: Cobalt Strike's default port for communication is 50050. If we detect this port open or find it in any network communication, then we know a Cobalt Strike attack has taken place.

- HTTP response whitespace: Cobalt Strike servers pre version 3.13 contain an easily detectable anomaly in their HTTP responses.



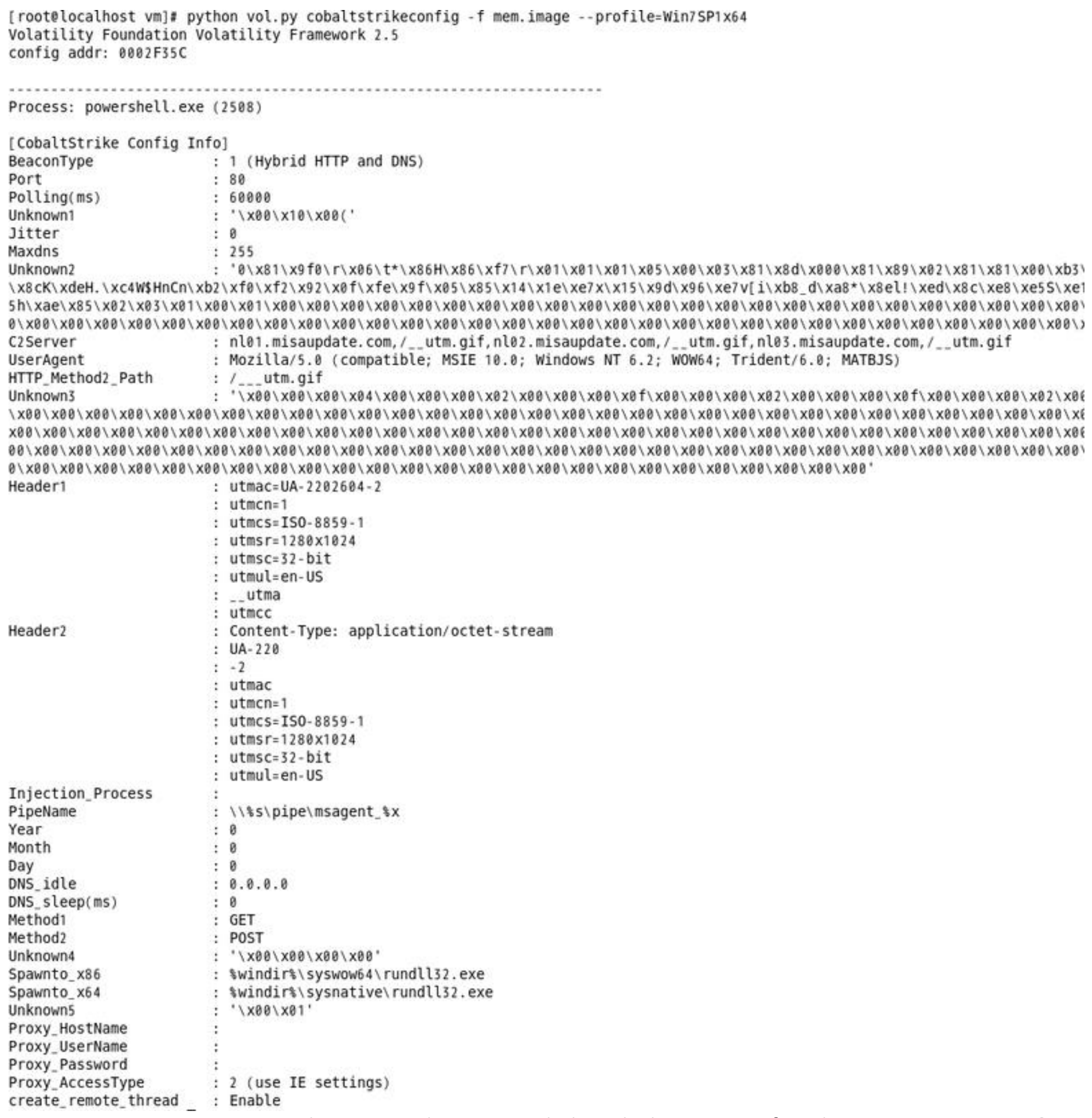
```
0030  ff ff e9 13 00 00 48 54  54 50 2f 31 2e 31 20 32   ······HT TP/1.1 2
0040  30 30 20 4f 4b 20 0d 0a  43 6f 6e 74 65 6e 74 2d   00 OK ·· Content-
0050  54 79 70 65 3a 20 74 65  78 74 2f 68 74 6d 6c 0d   Type: te xt/html·
0060  0a 44 61 74 65 3a 20 46  72 69 2c 20 33 20 4d 61   ·Date: F ri, 3 Ma
0070  79 20 32 30 31 39 20 31  34 3a 30 30 3a 30 39 20   y 2019 1 4:00:09
0080  47 4d 54 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   GMT··Con nection:
0090  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 43 6f 6e    keep-al ive··Con
00a0  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 35 0d 0a   tent-Len gth: 5··
00b0  0d 0a                                              ··
```

© Recorded Future

As seen in the above picture, there is a unusual whitespace after the status code, that is not present in legitimate communication.

- TLS negotiation fingerprint: TLS is a protocol which encrypts network traffic, to enable secure communication. To create the secure communication channel, a synchronization process needs to take place, that happens unencrypted. It is possible to create a “fingerprint” of this process, that can be used to identify similar network communication. With a sufficient IDS solution, we can analyze network traffic for fingerprints that allude to Cobalt Strike communication.

- In memory detection: If there is suspicion of a Cobalt Strike attack, then we can use an image created from the endpoint’s working memory, to detect whether a Cobalt Strike beacon is running. This process cannot necessarily be made automatic, but can provide valuable information to validate and remediate attacks.



```
[root@localhost vm]# python vol.py cobaltstrikeconfig -f mem.image --profile=Win7SP1x64
Volatility Foundation Volatility Framework 2.5
config addr: 0002F35C

--------------------------------------------------------------------
Process: powershell.exe (2508)

[CobaltStrike Config Info]
BeaconType                : 1 (Hybrid HTTP and DNS)
Port                      : 80
Polling(ms)               : 60000
Unknown1                  : '\x00\x10\x00('
Jitter                    : 0
Maxdns                    : 255
Unknown2                  : '0\x81\x9f0\r\x06\t*\x86H\x86\xf7\r\x01\x01\x01\x05\x00\x03\x81\x8d\x000\x81\x89\x02\x81\x81\x00\xb3\
\x8cK\xdeH.\xc4W$HnCn\xb2\xf0\xf2\x92\x0f\xfe\x9f\x05\x85\x14\x1e\xe7x\x15\x9d\x96\xe7v[i\xb8_d\xa8*\x8el!\xed\x8c\xe8\xe5S\xe1
5h\xae\x85\x02\x03\x01\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
C2Server                  : nl01.misaupdate.com,/__utm.gif,nl02.misaupdate.com,/__utm.gif,nl03.misaupdate.com,/__utm.gif
UserAgent                 : Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; MATBJS)
HTTP_Method2_Path         : /___utm.gif
Unknown3                  : '\x00\x00\x00\x04\x00\x00\x00\x02\x00\x00\x00\x0f\x00\x00\x00\x02\x00\x00\x00\x0f\x00\x00\x00\x02\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
Header1                   : utmac=UA-2202604-2
                          : utmcn=1
                          : utmcs=ISO-8859-1
                          : utmsr=1280x1024
                          : utmsc=32-bit
                          : utmul=en-US
                          : __utma
                          : utmcc
Header2                   : Content-Type: application/octet-stream
                          : UA-220
                          : -2
                          : utmac
                          : utmcn=1
                          : utmcs=ISO-8859-1
                          : utmsr=1280x1024
                          : utmsc=32-bit
                          : utmul=en-US
Injection_Process         :
PipeName                  : \\%s\pipe\msagent_%x
Year                      : 0
Month                     : 0
Day                       : 0
DNS_idle                  : 0.0.0.0
DNS_sleep(ms)             : 0
Method1                   : GET
Method2                   : POST
Unknown4                  : '\x00\x00\x00\x00'
Spawnto_x86               : %windir%\syswow64\rundll32.exe
Spawnto_x64               : %windir%\sysnative\rundll32.exe
Unknown5                  : '\x00\x01'
Proxy_HostName            :
Proxy_UserName            :
Proxy_Password            :
Proxy_AccessType          : 2 (use IE settings)
create_remote_thread _    : Enable
```

Sample output indicating a Cobalt Strike beacon was found in a memory image. © JPCERT

YARA

YARA is a tool is used to identify and categorize malware indicators and samples. With the help of this tool we can also identify Cobalt Strike related indicators. Below you can find as a demonstration a few YARA rules with the aforementioned purpose:

A rule that identifies hosts:

```
rule CobaltStrike_C2_Host_Indicator {
        meta:
                description = "Detects CobaltStrike C2 host artifacts"
                author = "yara@s3c.za.net"
                date = "2019-08-16"
        strings:
                $c2_indicator_fp = "#Host: %s"
                $c2_indicator = "#Host:"
        condition:
                $c2_indicator and not $c2_indicator_fp
                and not uint32(0) == 0x0a786564
                and not uint32(0) == 0x0a796564
}
```

A rule that identifies configuration files:

```
rule CobaltStrike_C2_Encoded_Config_Indicator {
        meta:
                description = "Detects CobaltStrike C2 encoded profile configuration"
                author = "yara@s3c.za.net"
                date = "2019-08-16"
        strings:
                $c2_enc_config = {69 68 69 68 69 6B ?? ?? 69 6B 69 68 69 6B ?? ?? 69 6A 69 6B 69 6D ?? ?? ?? ?? 69 6D 69 6B 69 6D
?? ?? ?? ?? 69 6C 69 68 69 6B ?? ?? 69 6F 69 68 69 6B ?? ?? 69 6E 69 6A 68 69}
        condition:
                $c2_enc_config
}
```

A group of rules that identifies beacons:

```
 privete rule cobaltstrike_beacon_raw
{
        strings:
                $s1 = "%d is an x64 process (can't inject x86 content)" fullword
                $s2 = "Failed to impersonate logged on user %d (%u)" fullword
                $s3 = "powershell -nop -exec bypass -EncodedCommand \"%s\"" fullword
                $s4 = "IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:%u/'); %s" fullword
                $s5 = "could not run command (w/ token) because of its length of %d bytes!" fullword
                $s6 = "could not write to process memory: %d" fullword
                $s7 =
"%s.4%08x%08x%08x%08x%08x.%08x%08x%08x%08x%08x%08x%08x.%08x%08x%08x%08x%08x%08x%08x.%08x%08x%08x%08x%08x%08
x%08x.%x%x.%s" fullword
                $s8 = "Could not connect to pipe (%s): %d" fullword

                $b1 = "beacon.dll"     fullword
                $b2 = "beacon.x86.dll" fullword
                $b3 = "beacon.x64.dll" fullword

        condition:
                uint16(0) == 0x5a4d and
                filesize < 1000KB and
                (
                        any of ($b*) or
                        5 of ($s*)
                )
}

 private rule cobaltstrike_beacon_exe
{
        condition:
                cobaltstrike_template_exe and
                filesize > 100KB and filesize < 500KB and
                pe.sections[pe.section_index(".data")].raw_data_size > 200000 and
                math.entropy(pe.sections[pe.section_index(".data")].raw_data_offset + 1024, 150000 ) >= 7
}

private rule cobaltstrike_beacon_b64
{
        strings:
                $s1a = "JWQgaXMgYW4geDY0IHByb2Nlc3MgKGNhbid0IGluam"
                $s1b = "ZCBpcyBhbiB4NjQgcHJvY2VzcyAoY2FuJ3QgaW5qZW"
                $s1c = "IGlzIGFuIHg2NCBwcm9jZXNzIChjYW4ndCBpbmplY3"

                $s2a = "RmFpbGVkIHRvIGltcGVyc29uYXRlIGxvZ2dlZCBvbi"
                $s2b = "YWlsZWQgdG8gaW1wZXJzb25hdGUgbG9nZ2VkIG9uIH"
                $s2c = "aWxlZCB0byBpbXBlcnNvbmF0ZSBsb2dnZWQgb24gdX"

                $s3a = "cG93ZXJzaGVsbCAtbm9wIC1leGVjIGJ5cGFzcyAtRW"
                $s3b = "b3dlcnNoZWxsIC1ub3AgLWV4ZWMgYnlwYXNzIC1Fbm"
                $s3c = "d2Vyc2hlbGwgLW5vcCAtZXhlYyBieXBhc3MgLUVuY2"

                $s4a = "SUVYIChOZXctT2JqZWN0IE5ldC5XZWJjbGllbnQpLk"
                $s4b = "RVggKE5ldy1PYmplY3QgTmV0LldlYmNsaWVudCkuRG"
                $s4c = "WCAoTmV3LU9iamVjdCBOZXQuV2ViY2xpZW50KS5Eb3"

        condition:
                filesize < 1000KB and
                5 of ($s*)
}
```

```
rule hacktool_windows_cobaltstrike_beacon
{
        meta:
                description = "Detection of the Beacon payload from Cobalt Strike"
                reference = "https://www.cobaltstrike.com/help-beacon"
                author = "@javutin, @joseselvi"
        condition:
                cobaltstrike_beacon_b64 or
                cobaltstrike_beacon_raw or
                cobaltstrike_beacon_exe
}
```

A rule that identifies post-exploitation modules:

```
rule hacktool_windows_cobaltstrike_postexploitation
{
        meta:
                description = "Detection of strings in the post-exploitation modules of Cobalt Strike"
                reference = "https://www.cobaltstrike.com/support"
                author = "@javutin, @mimeframe"
        strings:
                $s1 = "\\devcenter\\aggressor\\external\\"

        condition:
                filesize > 10KB and filesize < 1000KB and
                all of ($s*)
}
```

A rule that identifies payloads:

```
rule cobaltstrike_template_exe
{
        meta:
                description = "Template to provide executable detection Cobalt Strike payloads"
                reference = "https://www.cobaltstrike.com"
                author = "@javutin, @joseselvi"
        strings:
                $compiler = "mingw-w64 runtime failure" nocase

                $f1 = "VirtualQuery"   fullword
                $f2 = "VirtualProtect" fullword
                $f3 = "vfprintf"       fullword
                $f4 = "Sleep"          fullword
                $f5 = "GetTickCount"   fullword

                $c1 = { // Compare case insensitive with "msvcrt", char by char
                                0f b6 50 01 80 fa 53 74 05 80 fa 73 75 42 0f b6
                                50 02 80 fa 56 74 05 80 fa 76 75 34 0f b6 50 03
                                80 fa 43 74 05 80 fa 63 75 26 0f b6 50 04 80 fa
                                52 74 05 80 fa 72 75 18 0f b6 50 05 80 fa 54 74
                }
        condition:
                uint16(0) == 0x5a4d and
                filesize < 1000KB and
                $compiler and
                all of ($f*) and
                all of ($c*)
}
```

A rule that identifies mimikatz:

```
rule hacktool_windows_mimikatz_copywrite
{
        meta:
                description = "Mimikatz credential dump tool: Author copywrite"
                reference = "https://github.com/gentilkiwi/mimikatz"
                author = "@fusionrace"
                md5_1 = "0c87c0ca04f0ab626b5137409dded15ac66c058be6df09e22a636cc2bcb021b8"
                md5_2 = "0c91f4ca25aedf306d68edaea63b84efec0385321eacf25419a3050f2394ee3b"
                md5_3 = "0fee62bae204cf89d954d2cbf82a76b771744b981aef4c651caab43436b5a143"
                md5_4 = "004c07dcd04b4e81f73aacd99c7351337f894e4dac6c91dcfaadb4a1510a967c"
                md5_5 = "09c542ff784bf98b2c4899900d4e699c5b2e2619a4c5eff68f6add14c74444ca"
                md5_6 = "09054be3cc568f57321be32e769ae3ccaf21653e5d1e3db85b5af4421c200669"
        strings:
                $s1 = "Kiwi en C" fullword ascii wide
                $s2 = "Benjamin DELPY `gentilkiwi`" fullword ascii wide
                $s3 = "http://blog.gentilkiwi.com/mimikatz" fullword ascii wide
                $s4 = "Build with love for POC only" fullword ascii wide
                $s5 = "gentilkiwi (Benjamin DELPY)" fullword wide
                $s6 = "KiwiSSP" fullword wide
                $s7 = "Kiwi Security Support Provider" fullword wide
                $s8 = "kiwi flavor !" fullword wide
        condition:
                any of them
}
```

Suggested solutions

Above we have defined many methods of detecting Cobalt Strike attacks, all of which require the implementation of various security tools and devices. Below you can find the recommended solutions for the methodology developed by Black Cell Hungary Ltd.

Log collection and analysis system: *Splunk*

Splunk is a platform for searching, monitoring and analyzing real time data. The data is stored in a searchable repository from which graphs, reports, alerts, dashboards and visualizations can be generated. The previously described detection rules will be implemented in Splunk and the necessary logs can be easily collected with the low overhead, secure Splunk agent.

IDS (Intrusion Detection System): *Suricata*

Suricata is an open source network threat management software that provides IDS and IPS capabilities. It is notably good at deep packet analysis and detection of malicious signatures, which makes it essential for the detection of Cobalt Strike network indicators.

NGFW (Next-Generation Firewall): *Palo Alto*

Palo Alto Next-Generation firewalls combine the functionality of traditional firewalls with the functionality of other network security appliances, such as application firewalls, packet analysis devices and IDS/IPS systems. These firewalls have industry leading technologies such as TLS / SSL encrypted traffic inspection, QoS, advanced virus and threat protection, application specific rule application, and user identification. A Next-Generation Firewall helps prevent Cobalt Strike attacks through its traditional firewall functions, packet analysis and threat protection, and can help identify indicators of successful attacks.

EDR (Endpoint Detection and Response) & Vulnerability management: *MDATP*

Microsoft Defender Advanced Threat Protection is a platform designed to help organizations prevent, detect, investigate and react to advanced threats. It uses endpoint behavior detectors, cloud-based security analysis, threat intelligence, and vulnerability management to protect against even the most advanced threats. MDATP is used protect against, detect, and alleviate Cobalt Strike capabilities that threaten endpoints.

Glossary

Beacon: A piece of malicious software on a compromised system, which communicates with the attacker-controlled server and waits for new commands which are to be be executed on the system.

Brute force: In this context, a brute force attack means that the adversary tries to log in continuously with a high volume of passwords, hoping that one of the passwords will be correct.

Post-exploit: The phase after host exploitation, with the aim to determine the value of the host and maintain persistence on the host for future use. The value of the host is based on sensitivity of the data stored on the host and the effectiveness of the host in future network attacks. The aim of the methods described in this phase is to help the tester/attacker to identify sensitive data, configuration, communication and the relation with other devices on the network in order to maintain persistence for future access.

Cyber Kill Chain: The Cyber Kill Chain determines the steps of a cyber-attack. 1st step is reconnaissance, which aim is to gather corporate email addresses and gather information about the company. 2nd step is weaponization, the writing of the exploit based on the obtained information. 3rd step is the delivery of the prepared tool, which is followed by the 4th, exploitation step. The 5th step is the installation of the malware, and the 6th step is the setup of the communication with the control server. 7th and final step is the completion of commands issued by the command and control server.

EDR: A technology, that protects endpoints from threats. Gathers and analyzes data from the endpoints in order to detect potential problems and threats.

Hash: Hash algorithms are used to make "fingerprints", or in other words creating output data from input data with the following conditions:

- Always give the same output from a given input.
- The output data clearly refers to the input data, but the output data cannot be used to reverse engineer the input data.
- Even the slightest change in the input data results in a completely different output.

IDS: Intrusion detection system, which identifies suspicious, unusual or malicious activity on the network. Logs, categorizes, and classifies processes.

Log: A component of a logfile, details an event that occurred.

Metasploit: The Metasploit Project is a computer security project that provides information about security vulnerabilities and aids in penetration testing and IDS signature development. Its best-known sub-project is the open source Metasploit Framework, a tool for developing and executing exploit code against a remote target machine.

Mimikatz: Mimikatz is a post-exploit tool, which can gather passwords, hashes, PIN codes and Kerberos tickets from the computer's memory. Furthermore, it makes enables pass-the-hash, pass-the-ticket or golden ticket attack.

Payload: The part of the code program that actually executes the malicious action.

Phishing: An attempt to obtain sensitive information by disguising oneself as a trustworthy entity in an electronic communication.

Process creation log: A log detailing the creation a computer process.

QoS: The measurement or improvement of a network's performance, especially the perceived performance from the perspective of the end user.

RDP: A protocol developed by Microsoft, which provides a user with a graphical interface to connect to and control another computer remotely over a network connection.

Threat intelligence: Information gathered from different open sources about threats and threat actors in order to prevent cybersecurity incidents.

Vulnerability management: The process of identifying, classifying, prioritizing, remediating and mitigating software vulnerabilities.