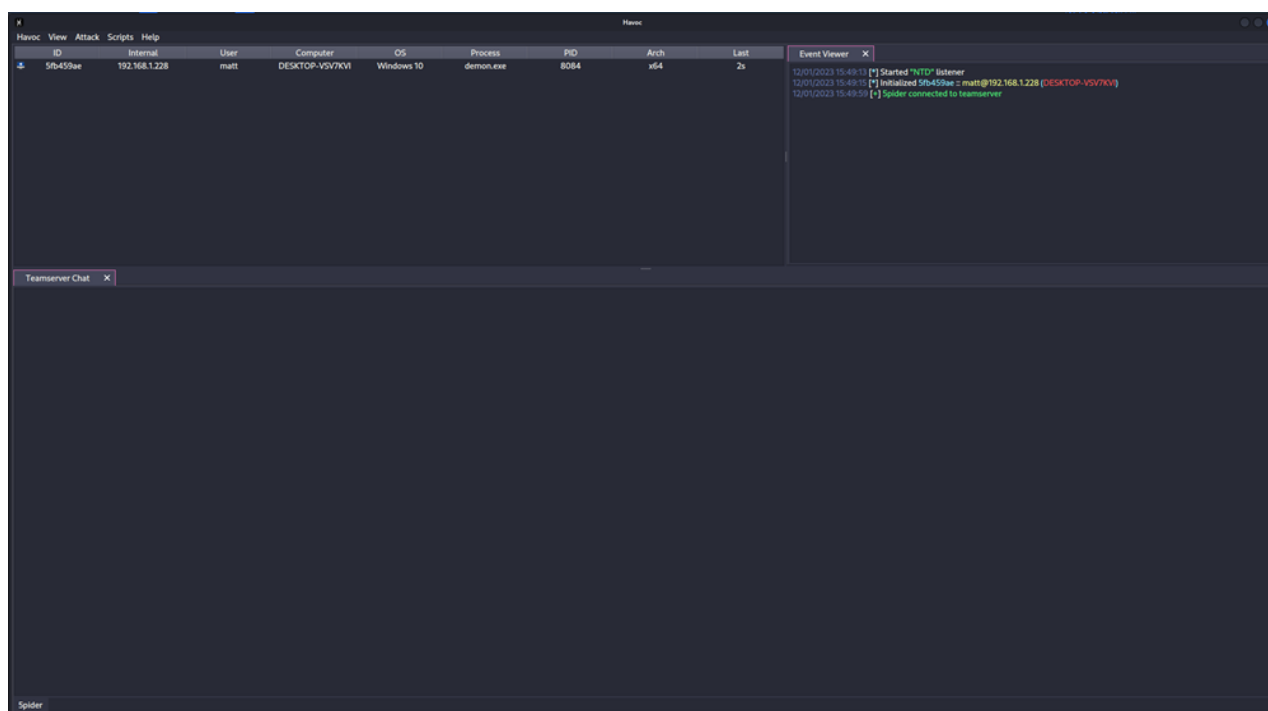


Sliver vs Havoc

 git.culbertreport.com/posts/Sliver-vs-Havoc

January 13, 2023



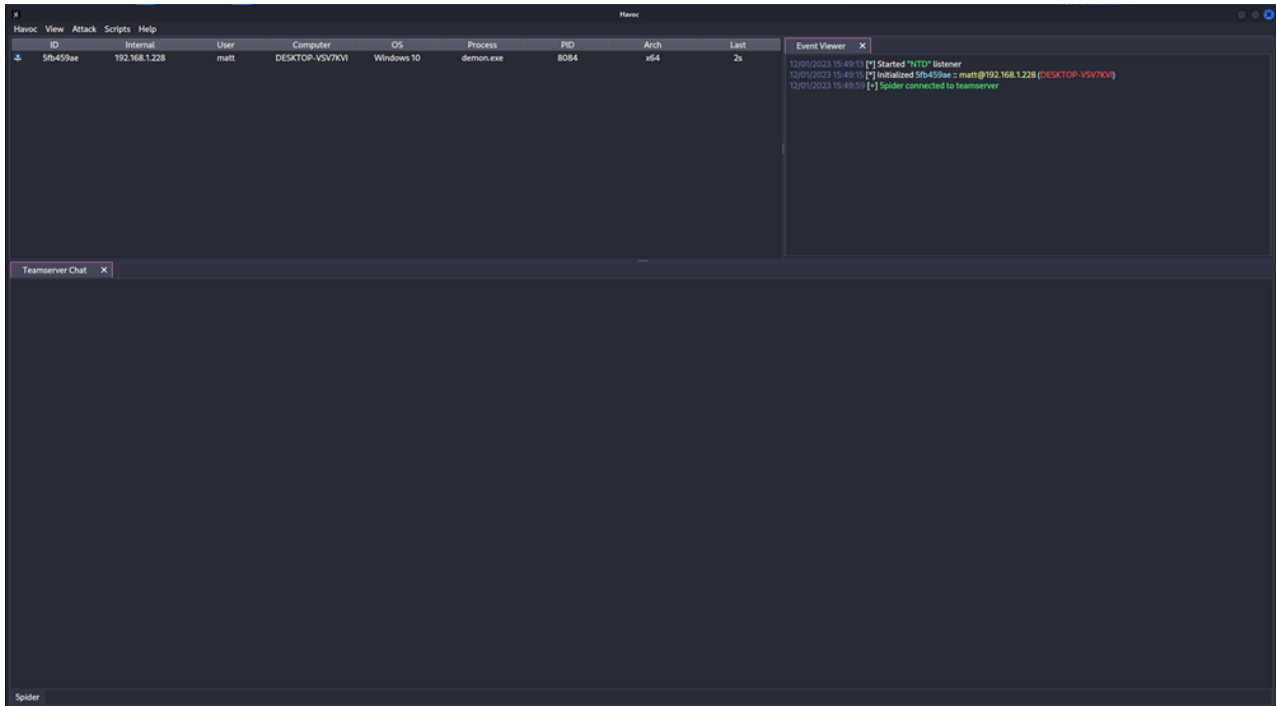
I wanted to objectively measure two well known frameworks against one another and see which fits certain needs best. To this end, each platform has been measured by if you can expand on them, how easy they are to get using, and why you might want one over the other. Before going further, for those unfamiliar, both Sliver and Havoc are command and control frameworks that are free to use on GitHub. Sliver is developed by BishopFox and is a reasonably old project that first started in 2019. It's seen some major upgrades since then, with the most recent being in October 2022. Havoc is a much newer tool developed by three independent contributors that was first started in September 2022. Both teams are responsive to questions and issues and, since they're open source, you are free to expand on them as you need. With that said, the first question that should be answered is how easy is it to spin one of these up?

How easy is it to spin one up and get going vs the other?

Havoc

- Havoc utilizes team profiles at launch to dictate implant functionality
- The documentation is fairly well laid out, but online only.
- You should have prior knowledge with C++ and implementing bypass techniques because out of the box the implants are detected quickly.

Havoc is very neat. Much like Cobaltstrike, you can start the teamserver by passing it a json profile of users and their passwords as well as some general functionality things like sleep time and jitter of your implants. An opsec concern right off the bat for teams may be that the passwords in this file not encrypted, so setting proper read permissions of it and the directory is very important. Once the server is started, the dashboard is very intuitively laid out if you've ever used Cobaltstrike.



Starting a listener is easy as well. Hitting **view -> listeners** allows us to add a new one. From here, you just step through the simple option menu and configure a few things like the port, if we'll be using hostnames, and the header to use, and you're set.

Create Listener

Name:

Payload:

Https

Config Options

Hosts

Add

Clear

Host Rotation:

round-robin

Host (Bind):

192.168.1.50

Port:

443

User Agent:

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/41.0.2226.150 Safari/537.31

Headers:

Add

Clear

Uris:

Add

Clear

Host Header:

☐ Enable Proxy connection

Proxy Type:

http

Proxy Host:

Proxy Port:

UserName:

Password:

Save

Close

And the payload options, while slim, have enough to them that we can make the necessary modifications to let them slip by the EDR we anticipate meeting. We can output in four file formats. These are exe, dll, shellcode, or service exe.

Payload

Agent: Demon

Options

Listener: NTL

Arch: Windows Dll

Format: ✓ Windows Shellcode

Config

Sleep: 2

Indirect Syscall: ✓

Sleep Technique: WaitForSingleObjectEx

Injection:

- Alloc: Native/Syscall
- Execute: Native/Syscall
- Spawn64: C:\Windows\System32\notepad.exe

Generate

The default implant of an EXE gets detected very quickly, so we'll be testing the shellcode option here and implant it through the resource section of another EXE we develop ourselves.

Next, we have to select the memory allocation and execution methods. For those, our options are syscalls or Win32. My understanding is that this means either performing a VirtualAlloc or NtAllocateVirtualMemory in the case of allocation, and for execution it would be NtCreateThreadEx or CreateRemoteThread.

NtAllocateVirtualMemory is typically called through VirtualAlloc since it is an undocumented function. As we briefly touched on in the NTDLL article in September 2022, these undocumented functions can change on a dime but are very useful for bypassing detection methods that focus solely on break points in NTDLL when called.

Their downside is that seeing syscalls like this is quite unusual and alarming, so EDR watching for this could alert very quickly. VirtualAlloc on the other hand is the lowest level call from user space that we can perform and hooks into Ring0 using syscalls to allocate memory within the bounds of what's known as the system granularity boundary. For most Windows systems, this will be 64kb. Allocations also must be a multiple of this boundary, so for example a 3 byte allocation would cause errors. Moving to the execution methods, NtCreateThreadEx is the undocumented, lower level, version of CreateRemoteThread. Like NtAllocateVirtualMemory, it is contingent on syscalls being up to date and suffers from the same pitfalls. CreateRemoteThread is much safer *but* also easier to detect. Finding a balance between the two options is important when generating your implant. Different EDR might be tuned to detect one method more over the other.

After taking all this into account and developing our executable for the shellcode, we can see that the detection levels has dropped from 26 to 16. That's not ideal, but if your target platform is Microsoft, you've made it past stage 1 with only a few changes.

The image shows the VirusShare analysis interface for a file named 'test.exe'. The top section displays a score of 16/72, indicating that 16 security vendors and no sandboxes flagged the file as malicious. The file's SHA-256 hash is d02713884c77f995ccc0550ccc3672c5487fccc905cd54424a1ab37f8339dde2. The file size is 202.67 KB, and it was uploaded on 2023-01-06 at 16:24:48 UTC. The file type is EXE. The interface includes tabs for DETECTION, DETAILS, BEHAVIOR, and COMMUNITY. The DETECTION tab is active, showing a table of security vendors' analysis.

Security vendors' analysis	Detection	Vendor	Detection
Avast	Win64:Trojan-gen	AVG	Win64:Trojan-gen
Cybereason	Malicious.c47eef	Cynet	Malicious (score: 100)
Elastic	Malicious (high Confidence)	ESET-NOD32	A Variant Of Win64/Rozena.UD
Fortinet	W64/Rozena.KP!tr	Google	Detected
Ikarus	Trojan.Win64.Agent	Kaspersky	VHO:Backdoor.Win64.Havoc.gen
Malwarebytes	Trojan.Crypt	Rising	Backdoor.Havoc!8.970A (TFE:5:7FcE8O...
Sangfor Engine Zero	Trojan.Win32.Save.a	Symantec	ML.Attribute.HighConfidence
Trellix (FireEye)	Generic.mg.c85a31986303b5f0	ZoneAlarm by Check Point	VHO:Backdoor.Win64.Havoc.gen
Acronis (Static ML)	Undetected	Ad-Aware	Undetected

While using Havoc, it felt like their target audience was professionals who are experienced with other C2 platforms and want something that they can build off of on their own. It's hard to get leadership buy in on a project that is only maintained by three people and doesn't have a company backing it like BishopFox, so it's hard to say how likely you are to encounter it in an enterprise. For those who really enjoy working in a GUI though, this will definitely scratch that itch.

Sliver

Let's now shift a little and check out Sliver.

- Sliver is completely terminal based meaning if you need a GUI to be able to visualize things, this won't work for you.

- The documentation is expansive and intimidating, which is great, and you can type help for any function.
- Sliver also lets you set expiration dates for beacons, so they stop working after a set time.

Sliver gives you everything and expects you know what to do with it. If you don't know what to do with it, there's a help dialogue for each option, but aside from that you are left to figure it out. Just check out the options for compiling an implant.

```
Flags:
-a, --arch string      cpu architecture (default: amd64)
-c, --canary string    canary domain(s)
-d, --debug            enable debug features
-G, --disable-sgn      disable shikata ga nai shellcode encoder
-n, --dns string       dns connection strings
-e, --evasion          enable evasion features (e.g. overwrite user space hooks)
-E, --external-builder use an external builder
-f, --format string    Specifies the output formats, valid values are: 'exe', 'shared' (for dynamic libraries), 'service' (see 'psexec' for more info) and 'shellcode' (windows only) (default: exe)
-h, --help            display help
-b, --http string      http(s) connection strings
-X, --key-exchange int  wg key-exchange port (default: 1337)
-w, --limit-datetime string limit execution to before datetime
-x, --limit-domainjoined limit execution to domain joined machines
-F, --limit-fileexists string limit execution to hosts with this file in the filesystem
-z, --limit-hostname string limit execution to specified hostname
-L, --limit-locale string limit execution to hosts that match this locale
-y, --limit-username string limit execution to specified username
-k, --max-errors int    max number of connection errors (default: 1000)
-m, --mtls string       mtls connection strings
-N, --name string       agent name
-p, --named-pipe string named-pipe connection strings
-o, --os string         operating system (default: windows)
-P, --poll-timeout int  long poll request timeout (default: 360)
-j, --reconnect int     attempt to reconnect every n second(s) (default: 60)
-R, --run-at-load      run the implant endpoint from DLLMain/Constructor (shared library only)
-s, --save string       directory/file to the binary to
-l, --skip-symbols     skip symbol obfuscation
-Z, --strategy string  specify a connection strategy (r - random, rd - random domain, s - sequential)
-T, --tcp-comms int     wg C2 comms port (default: 8888)
-i, --tcp-pivot string  tcp-pivot connection strings
-l, --template string  implant code template (default: sliver)
-t, --timeout int       command timeout in seconds (default: 60)
-g, --wg string         wg connection strings
```

If you don't know why you might want to disable the Shikata-Ga-Nai shellcode encoder, you're offered no explanation. There are numerous guides available online for Sliver, which lowers the learning curve significantly and if you would like to get started, these are almost mandatory readings. [Some can be found here](#). The linked reading does a deep dive into Slivers code and finds some interesting shortcuts taken by the developers. For example, generating stagers under the hood is handled through MSFVenom and, while you can specify a DNS name, only hardcoded IPs are passed. This is not publicly documented and a potential drawback when trying to fly under the radar.

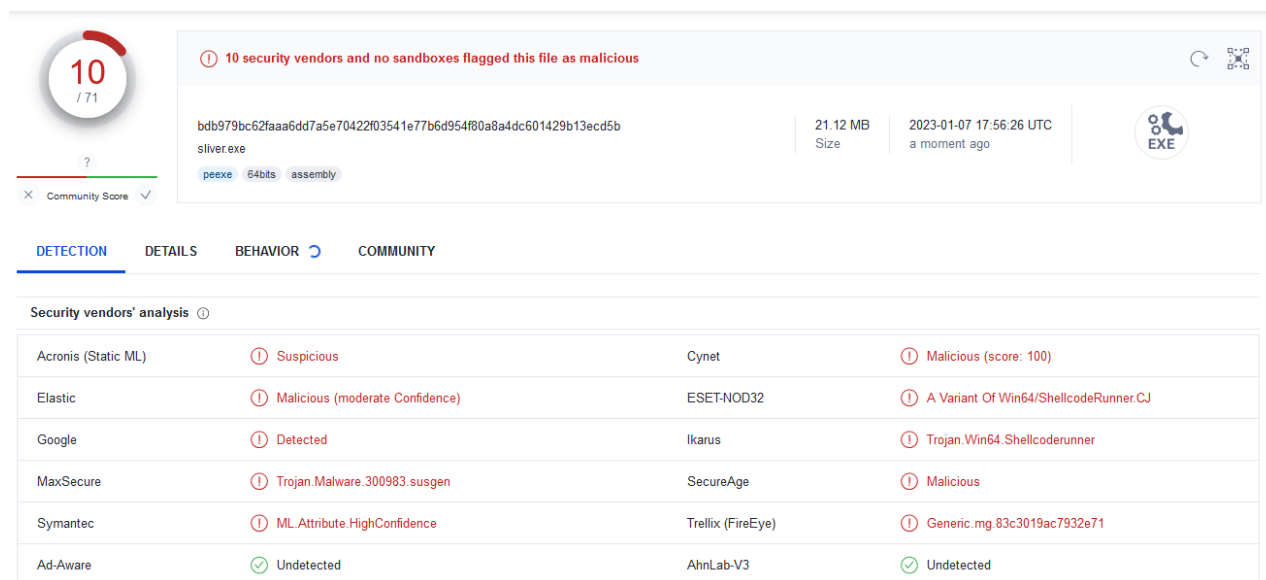
Again, we will be compiling and outputting shellcode for us to further obfuscate and hide. This can be done with the **-f** flag.

Because Sliver does not make it obvious what their execution and injection methods are like Havoc does, we'll have to do some digging. Looking at the `task_windows.go` file, `VirtualAlloc` and `CreateRemoteThread` are both used to allocate and execute the in memory objects. Really interestingly, BishopFox has gone out of their way to implement their own syscalls package, which only has the commands that they will need to use. It's a smart way of limiting bloat and reducing imports. For comparison, 7zip has 69 imports from Kernel32 alone. This is definitely a key piece as well in reducing their detection rate combined with EDR sometimes not being equipped to analyze Go binaries.

Sliver also has its own methods for evasion, though this is not something the authors have focused heavily on. They provide two functions that work in tandem in order to accomplish this. The first is `RefreshPE` which reloads the .text section of a file from disk. The second is what they call `writeGoodBytes`. This function takes a process name and a few other variables and proceeds to reload the clean version of a dll into the current processes memory through the first function mentioned. This is called through `err :=`

`evasion.RefreshPE('c:\windows\system32\ntdll.dll')`. Do note, to do this they allocated RWX memory sections, which will set off EDR. Havoc also allocates memory in the same way, so it's a knock against both.

Now that we have an idea of how things are executed, we can go back to loading our shellcode into the resource section of our executable. Without even encrypting it, we see a strikingly lower detection rate than Havocs shellcode loaded the same way.



The image shows a VirusShare analysis page for a file named `sliver.exe`. At the top, a circular badge displays a "Community Score" of 10 out of 71. A message states: "10 security vendors and no sandboxes flagged this file as malicious". The file's SHA-256 hash is `bdb979bc62faaa6dd7a5e70422f03541e77b6d954f80a8a4dc601429b13ecd5b`, its size is 21.12 MB, and it was uploaded on 2023-01-07 at 17:56:26 UTC. The file type is identified as "EXE". Below this, tabs for "DETECTION", "DETAILS", "BEHAVIOR", and "COMMUNITY" are visible, with "DETECTION" selected. The "Security vendors' analysis" section contains a table with the following data:

Vendor	Detection	Vendor	Detection
Acronis (Static ML)	ⓘ Suspicious	Cynet	ⓘ Malicious (score: 100)
Elastic	ⓘ Malicious (moderate Confidence)	ESET-NOD32	ⓘ A Variant Of Win64/ShellcodeRunner.CJ
Google	ⓘ Detected	Ikarus	ⓘ Trojan.Win64.Shellcoderunner
MaxSecure	ⓘ Trojan.Malware.300983.susgen	SecureAge	ⓘ Malicious
Symantec	ⓘ ML_Attribute.HighConfidence	Trellix (FireEye)	ⓘ Generic.mg.83c3019ac7932e71
Ad-Aware	✅ Undetected	AhnLab-V3	✅ Undetected

A final piece to touch on, BOFs. Sliver allows operators to port over custom beacon object files that were written for Cobaltstrike, as well as downloading prebuilt ones. Let's see how easy they are to drop. Sliver uses something called Armory to manage these in a sort of extension manner. You can list extensions available through `armory` and install them through `armory install`. Then, once you've gotten the BOFs installed, you hop into a session for a beacon and can run any of them just by typing their name and the flags that they require. This is very reminiscent of Metasploit modules and is intuitive to use and figure out. In fact, the whole program is reminiscent of Metasploit down to the help dialogue and how information is presented. This is not surprising, Sliver utilizes MSF internally and has built in functionality that allows operators to do things like inject MSF payloads.

```
File Actions Edit View Help
[server] sliver > help

Commands:
clear      clear the screen
exit      exit the shell
help      use 'help [command]'
monitor   Monitor threat in
wg-config  Generate a new Wi
wg-portfwd List ports forward
wg-socks   List socks server

Generic:
aliases   List current
armory     Automatical
background Background
beacons    Manage beacon
builders   List external
canaries   List previous
cursed     Chrome/elect
dns        Start a DNS
env        List environment
generate   Generate an
hosts      Manage the
http       Start an HTTP

File Actions Edit View Help
msf6 > help

Core Commands

Command      Description
?            Help menu
banner       Display an awesome metasploit banner
cd           Change the current working directory
color        Toggle color
connect      Communicate with a host
debug        Display information useful for debugging
exit         Exit the console
features     Display the list of not yet released features that can be opted in to
get          Gets the value of a context-specific variable
getg         Gets the value of a global variable
grep         Grep the output of another command
help         Help menu
history      Show command history
load         Load a framework plugin
quit         Exit the console
repeat       Repeat a list of commands
route        Route traffic through a session
save         Saves the active datastores
sessions     Dump session listings and display information about sessions
```

Sliver had a lot of touches to it that really gave it the feel of an enterprise ready software. There's a reputable company backing it, it has touches like beacon kill dates built in, and there's a wide range of support for expanding the software while staying within their ecosystem such as with BOFs. It's no surprise that threat actors have caught onto this framework and are integrating it with more and more campaigns. Microsoft has even noted that it is being used in tandem with Cobaltstrike in some attacks.

How can you expand on them?

Havoc

Custom agents are a little tricky to figure out but there are examples provided. It will take some trial and error on your end to determine the exact way to call arguments and add functionality, but CodeXTF2 was kind enough to include their own demo agent written in Python as a demonstration for ease of understanding. Aside from custom agents, if you wish to add your own functionality to the teamserver, the codebase is vast and somewhat poorly commented which makes customization not the easiest.

Sliver

Sliver allows you to use beacon object files from Cobaltstrike to extend the post exploitation capabilities of the framework. While this isn't necessarily a custom agent, this allows customization of agents to add further capabilities. To the unfamiliar, BOF are compiled C programs that are position independent code. Meaning that like Donut, no matter where it is placed in memory, it can execute. BOFs are injected directly into the beacon process typically, avoiding IoC's associated with alternative approaches like execute-assembly which spawns a new empty process to run these assemblies in. Cobaltstrike never intended for BOFs to be executing long-running commands, that's what execute-assembly, is for. Instead, these are for quick functions that return data shortly after launch. Sliver including these as a method for expanding functionality is a

very neat approach. Read more about BOFs in the sources section. Regarding their codebase it's only marginally better commented with comments above primary functions describing what they do, but very little otherwise. This is disappointing to see, I wish commenting code was a more common practice as it helps new people get up to speed with each functions purpose much quicker.

Why might you want one over the other?

Havoc

- The GUI is very intuitive and well thought out.
- Visualizing compromised machines and SSH tunnels was intelligently setup.
- Less can be more. Giving people less to work with means they go deeper on working with what they have. Breeds innovation.

Havoc has the more user-friendly GUI of the two and makes it easy to start and deploy listeners and implants. If you're wanting to get an introduction to C2s and customization, their GUI makes it much easier for newer operators to get accustomed to the environment. The profiles also work more intuitively than Slivers method of saving configurations for listeners and implants by having you edit a JSON file. The roundrobin technique of hosts and URLs is also very clever, allowing much more randomness to be added, which it makes the defenders job that much harder.

Sliver

- Multi platform framework.
- Deeper resources to put into development.
- BOFs implementation is very smooth.
- More listener options like wireguard and mTLS.

If you want the end game experience of free C2's, this hands down goes to Sliver with their implementation of BOFs. Sliver also has agents that can be deployed to not only Windows machines, but also Mac and Linux. On top of this, their default detection rate with shellcode was also much lower. These can all be attributed to the number of resources that BishopFox has to throw at things compared to Havocs development team.

Sources

(BOF) <https://www.trustedsec.com/blog/a-developers-introduction-to-beacon-object-files/>

(BOF) <https://www.cobaltstrike.com/blog/writing-beacon-object-files-flexible-stealthy-and-compatible/>

<https://0x00sec.org/t/process-injection-remote-thread-injection-or-createremotethread/24399>

<https://github.com/BishopFox/sliver>

<https://github.com/HavocFramework/Havoc>

<https://mez0.cc/posts/detecting-syscalls-with-fennec/>