SECURITY ADVISORY

CYBER STASH

July, 2025

Offensive Tradecraft Intelligence Brief

Stealth Mode: Evading the Defenders

Context

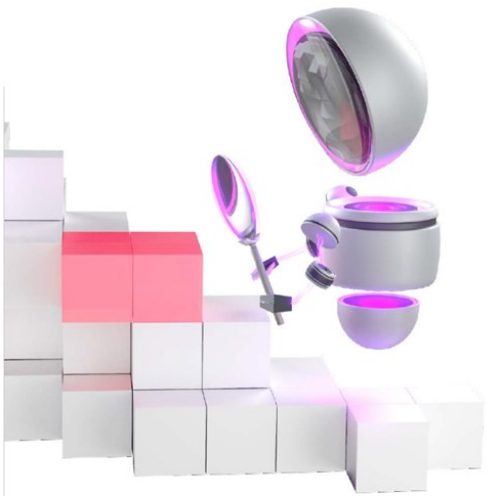
Despite continued advancements in endpoint security, from Next-Gen Antivirus (NGAV) to modern EDR and SIEM platforms, many organisations operate under an illusion of protection. In reality, threat actors are innovating faster than defenders can adapt. The emergence of offensive tools like the newly released Zig Strike toolkit reveals just how easily even advanced, policy-compliant security stacks can be bypassed.

Zig Strike represents a new generation of open-source red teaming frameworks that are designed not merely to test detection, but to exploit the architectural blind spots of endpoint protection platforms, including Microsoft Defender for Endpoint. Written in the memory-safe, high-performance Zig programming language, the toolkit provides attackers with a web interface for crafting highly evasive payloads that bypass modern AV, NGAV, and EDR solutions through a blend of stealthy injection techniques, compile-time obfuscation, anti-sandbox mechanisms, and entropy reduction.

More than a red team utility, Zig Strike is a proof point: even the most hardened environments are susceptible to techniques that operate below the radar of behavioural analytics and machine learning-based detection engines. By leveraging trusted interfaces (e.g. Excel Add-ins), hijacking process threads, fragmenting shellcode across memory, and exploiting legitimate APIs, the toolkit underscores a troubling reality: the modern attacker doesn't need to break in—they can walk in undetected.

This report uses Zig Strike as a lens to examine the broader industry-wide challenge of endpoint evasion, referencing recent CVEs and bypass techniques that illustrate the systemic weaknesses in Defender and EDR architectures. We explore:

- How offensive toolkits are evolving to bypass enterprise-grade security.
- The injection and obfuscation techniques used to evade detection at runtime and at rest.
- Real-world vulnerabilities and architectural design flaws that make these bypasses possible.
- Strategic recommendations for mitigating exposure and strengthening endpoint resilience.



Why This Matters:

In an era where nation-state-level tools are open-sourced and operationalised by commodity threat actors, enterprises can no longer rely solely on vendor-issued controls or default configurations. Understanding the mechanics of evasion is essential to building effective countermeasures, not just detecting known threats, but anticipating unknown techniques.

This is not just about tool evasion. It is about trust in our defensive architecture, and the imperative to reassess that trust with urgency.

How Offensive Toolkits Are Evolving to Bypass Enterprise-Grade Security

The rise of offensive frameworks like Zig Strike, Sliver, Mythic, and Brute Ratel signals a pivotal shift in the threat landscape. Once the domain of nation-state actors, these tools are now open-source, commoditised, and virtually indistinguishable from legitimate software in both form and function. Purpose-built to evade enterprise-grade defences, including Microsoft Defender, CrowdStrike, Cortex XDR, FortiEDR, and Sophos, they bring stealth, flexibility, and surgical precision to even low-skilled adversaries.

Key Evolutionary Traits in Modern Toolkits:

Compile-Time Obfuscation & Entropy Reduction

- Zig Strike uses Zig's comptime functionality to fragment shellcode into wide-encoded strings, burying payloads in .rdata sections to evade static AV scanning.
- Similar obfuscation techniques have been seen in attacks exploiting CVE-2024-20671, where logic flaws in Microsoft Defender allowed crafted binaries to bypass signature inspection by embedding malicious content in non-executable sections.

Process Injection Without Suspicious API Usage

- Modern toolkits are moving away from classic injection APIs (`CreateRemoteThread`, `WriteProcessMemory`) that trigger EDR heuristics.
- Zig Strike implements thread hijacking and memory mapping techniques using `SetThreadContext`, `MapViewOfFileNuma2`, and `CreateFileMappingW`, mimicking legitimate process behaviour to bypass detection.
- These evasion paths align with lessons from CVE-2024-5905, where early startup vulnerabilities in Cortex XDR's driver allowed malicious code execution prior to EDR agent initialisation.

Sandbox & Virtualisation Evasion

- Trusted Platform Module (TPM) checks, domain membership validation, and delayed execution are commonly embedded to avoid detection in sandbox environments.
- CVE-2023-36025 was exploited to bypass Microsoft SmartScreen protections using `.URL` files, triggering payloads only in real user environments, an approach now mirrored in red team tools using Excel Add-ins (XLL) or signed Office macros.

Living-Off-the-Land Binary (LOLBins) Abuse

- Adversaries increasingly exploit trusted, signed Windows binaries—known as LOLBins—such as regsvr32, msiexec, wscript, powershell, and rundll32 to bypass security controls and execute malicious payloads without triggering alerts.
- CVE-2021-40444 (MSHTML RCE) is a prime example of rundll32.exe abuse. Malicious Office documents exploited MSHTML to load remote ActiveX/DLL payloads, which were executed via the trusted rundll32.exe using scriptlet protocols (T1218.011), enabling stealthy Cobalt Strike deployment and evasion of EDR and application controls.

Syscall-Level Evasion & Kernel Interface Abuse

- Toolkits are adopting direct/indirect syscalls and API unhooking to bypass user-mode hooks deployed by EDR agents. The "Bring Your Own Vulnerable Driver" (BYOVD) technique, exploited in various campaigns, leverages signed drivers to disable EDR kernel callbacks.
- Examples include CVE-2025-47161, which enabled SYSTEM-level escalation by exploiting flaws in Defender for Endpoint's service permissions—effectively neutering EDR from within.

Cloud EDR Management Interface Weaknesses

- Adversaries increasingly target misconfigured or exposed EDR cloud consoles and APIs, allowing them to disable protections, manipulate telemetry, or modify policies without needing endpoint access.
- Attackers exploited SentinelOne's cloud console by uploading a tampered MSI installer, silently uninstalling the EDR agent and bypassing tamper protection to enable undetected ransomware deployment. Similarly, adversaries with access to a misconfigured CrowdStrike Falcon console used custom MSI packages to remotely disable EDR agents across endpoints without triggering local defenses.

BYOVD (Bring Your Own Vulnerable Driver)

- Attackers abuse digitally signed but vulnerable drivers to gain kernel-level execution, disable EDR hooks, or kill protected processes. These attacks bypass user-mode protections entirely by exploiting the trusted Windows driver loading mechanism.
- CVE-2015-2291 was exploited by BlackByte and AvosLocker ransomware groups using the vulnerable IntelHaxm.sys driver to terminate EDR processes and evade detection before executing payloads.
- CVE-2024-1853 abused by tools like SpyBoy (Terminator) and KillerUltra, used the vulnerable Zemana AntiLogger driver to kill Defender and EDR services from kernel space.

Malicious Delivery Vectors (ISO, OneNote, XLL)

- Threat actors increasingly rely on trusted file formats to evade endpoint controls, phishing defences, and ASR rules.
- ISO images bypass Mark-of-the-Web (MotW) when mounted.
- OneNote (ONE) files hide executables behind images to trick users.
- Excel XLL add-ins enable DLL execution via Excel's trusted plugin system, a method rarely scanned or blocked. Campaigns using IcedID, QakBot, and Remcos routinely employ these vectors to establish initial access.

AMSI and Script Engine Bypass

- Advanced malware bypasses Microsoft's Antimalware Scan Interface (AMSI) to evade detection during script-based attacks.
- Techniques include in-memory patching of AmsiScanBuffer() or loading obfuscated payloads after disabling AMSI.
- This removes runtime inspection of PowerShell, VBScript, and JScript commands, effectively blinding EDRs that rely on AMSI telemetry. Common in toolkits such as Brute Ratel, Covenant, and Cobalt Strike.

Bottom Line

Modern offensive toolkits don't just exploit software vulnerabilities, they exploit the limitations of detection logic, overreliance on trusted architectures, and assumptions that traditional security controls will alert when compromised. From stealthy evasion techniques that operate undetected within a functioning Defender environment, to EDR killers that neutralise protections at the kernel level, attackers are making enterprise-grade security tools appear operational while silently disabling or bypassing them. These developments expose a critical weakness in today's security model: the illusion of protection.

To defend effectively, organisations must go beyond conventional telemetry, rule engines, and alert-based detection. This requires:

- **Independent breach detection capabilities** that function outside the primary tech stack
- **Forensic-based methods** that detect signs of compromise even when logs are missing or manipulated
- **Validation-driven visibility** that tests whether controls are not just deployed, but actively working

Without independent validation, defenders risk trusting a compromised signal — and not knowing they've already been breached.

Threat Detection vs. Breach Detection: A Critical Distinction in Modern Defence

As attackers evolve, so must our understanding of what security solutions are actually capable of detecting — and what they miss. One of the most important but often misunderstood distinctions in enterprise security is between threat detection and breach detection.

What's the Difference?

Category	Threat Detection	Breach Detection
Objective	Detect live threats during execu-	Detect if a system has already been com-
Assumptions	Assumes endpoint controls (EDR, AV, telemetry pipelines) are opera-	Assumes controls can be bypassed or neutralised by the adversary
Methods	Uses meta-data, telemetry, signa-	Uses direct forensic evidence, validation
Technology Stack	Relies on SIEM, EDR/XDR, NGAV, and their logs or alerts	Functions outside the existing stack, unaffected by telemetry or control failure
Timeframe	Typically real-time or near-real-time	Retrospective or post-compromise valida-
Analytical Focus	Tracks suspicious behavior (process, network, identity anoma-	Seeks out indicators of failure or compromise that tools may have missed
Examples	Detection rules, behavioural analytics, alert correlation	Evidence of persistence, memory implants, modified services, abnormal bina-

Why Breach Detection Is Essential

Modern malware is increasingly engineered to:

- Evade detection, not just execution.
- Bypass telemetry, by running outside monitored processes or disabling logging.
- Manipulate operator confidence, by letting tools like Defender or EDR appear "healthy" while inactive or disabled.

In this context, organisations that rely solely on real-time detection are assuming their security stack is infallible — an assumption advanced adversaries actively exploit.

Bottom Line

You can't detect what your tools don't see — and you can't trust what your tools no longer control. A mature security posture must include both:

- Threat detection for early interception, and
- Breach detection to validate whether those defences are still holding.

Without the latter, defenders are fighting blind.

Understanding the Threat Landscape: EDR Evasion vs. EDR Killers

Modern adversaries don't just bypass security — they undermine its very foundations. Whether through silent evasion or active termination, both techniques result in the same outcome: endpoint compromise without effective defence.

Defining Terminology

Term	Definition	Example
EDR Evasion	Techniques that allow malicious activity to avoid detection or response without disabling the security tool.	Snake Keylogger evades Windows Defender using obfuscation and anti-sandbox checks.
EDR Killer	Techniques that actively disable, uninstall, or neutralise the EDR agent or its	Terminator (SpyBoy) and AUKill use BYOVD attacks to kill security

Key Distinctions

- Evasion affects visibility – malware operates silently while the EDR continues to appear functional.
- Killers affect availability – the EDR service is forcibly disabled, leaving the system entirely defenceless.

Operational Impact

Evasion – Silent Intrusion Under the Radar

- The EDR remains "green" in dashboards and telemetry.
- Security teams receive no alerts despite malware activity.
- False assurance leaves organisations blind to compromise.

Killers – Direct Neutralisation of Endpoint Defences

- The EDR agent is disabled or uninstalled, often using kernel-level exploits.
- Attackers gain unrestricted access post-exploitation.
- May trigger alerts — but often after damage is done.

Why This Matters

In an enterprise context:

- Evasion undermines detection accuracy, allowing lateral movement and exfiltration.
- Killers eliminate visibility and break containment, enabling malware or ransomware to execute freely.

Both are control failures — and both must be tested for and mitigated.

Hidden Realities That Most Enterprises Miss

EDR Bypass Is Now a Feature, Not a Flaw

What many defenders fail to realise is that modern red team and malware frameworks are purpose-built to evade detection:

- Pre-built AV/EDR evasion profiles for Defender, CrowdStrike, Cortex XDR, and others.
- Use of malleable C2 profiles, obfuscated syscalls, stack duplication, and in-memory shellcode loading.
- Delivery via Excel Add-ins (XLL) and signed but vulnerable drivers (BYOVD) to bypass controls.

These tools are not fringe — they are used in production by threat actors and red teams alike.

EDR is often not "defeated" — it is quietly walked past.

EDR Test Results Are Misleading

Most EDR evaluations (e.g., MITRE ATT&CK, AV-Comparatives) do not include:

- BYOVD attacks that disable agents.
- Shellcode-based loader evasion.
- ISO, OneNote, or XLL-based delivery vectors.
- Sleep obfuscation, dynamic API resolution, or AMSI patching.

As a result, security teams may be misled into a false sense of readiness. What is being tested is often detection of known signatures — not real-world bypass resilience.

Why This Matters

In an enterprise context:

- **Evasion** undermines detection accuracy, allowing lateral movement and exfiltration.
- **Killers** eliminate visibility and break containment, enabling malware or ransomware to execute freely.

Both are control failures, and both must be tested for and mitigated.

Real-World Examples of EDR and Defender Bypass Techniques

Despite the continuous evolution of endpoint security solutions, threat actors are actively exploiting architectural weaknesses, implementation flaws, and trust assumptions in leading EDR platforms. The following real-world examples illustrate how adversaries have successfully bypassed or disabled endpoint protections — not through stealth alone, but by directly undermining the security mechanisms themselves.

These cases include CVE-tracked vulnerabilities, kernel-level driver abuse, signature validation bypasses, and agent tampering techniques — each one representing a critical failure point in modern defence strategies. Together, they demonstrate the pressing need for organisations to validate their endpoint protection controls, rather than simply trust that they're working.

CVE-2023-24934 — Defender Signature Update Hijack

Local attacker hijacks Defender's signature update mechanism.

⇒ **Mechanism:** Alters or injects .vdm/.sig signature files to bypass detection.

⇒ **Security Impact:** Turns Defender into a blind spot, allowing malware to be classified as benign.

2. CVE-2024-21412 — SmartScreen Shortcut Bypass (DarkGate)

Malicious .url files exploit SmartScreen validation gaps.

⇒ **Mechanism:** Chained shortcut files execute MSI payloads via silent bypass using remote shares.

⇒ **Security Impact:** Malware installs without SmartScreen alerts, enabling stealthy loader execution.

3. CVE-2023-36025 — Defender SmartScreen Bypass (Phemedrone Stealer)

Abuses .url file handling to launch payloads without prompts.

⇒ **Mechanism:** Downloads malicious .cpl and wer.dll, sideloaded via control.exe and WerFaultSecure.exe.

⇒ **Security Impact:** Full endpoint compromise with no security warnings and persistence via scheduled tasks.

4. Direct Syscalls and XOR-Encrypted Shellcode — Hackmosphere 2025

Custom payload bypasses Defender using stealthy memory execution.

⇒ **Mechanism:** Avoids API hooks by invoking syscalls directly; decrypts shellcode in memory using XOR.

⇒ **Security Impact:** Evades static and behavioral detection, delivers in-memory backdoors undetected.

5. CVE-2024-5905 — Cortex XDR Agent Disablement

Enables unauthorised disablement of the Cortex XDR agent on Windows systems.

⇒ **Mechanism:** Exploits weaknesses in service authentication to terminate or suspend XDR processes.

⇒ **Security Impact:** Endpoint monitoring is fully deactivated, creating a blind spot for threat detection and response.

6. CVE-2024-5912 — Cortex XDR Signature Bypass

Bypasses executable signature validation in Cortex XDR.

⇒ **Mechanism:** Crafted binaries with forged or missing signatures are incorrectly classified as trusted.

⇒ **Security Impact:** Permits the execution of unverified code, allowing threat actors to operate undetected in protected environments.

7. CVE-2023-3665 — Trellix ENS AMSI Disablement

Low-privilege user disables AMSI in ENS via environment variables.

⇒ **Mechanism:** Alters process initialization to prevent AMSI hook loading.

⇒ **Security Impact:** Disables critical runtime defenses across AV/EDR platforms.

8. EDR Sensor Unhooking via API Abuse

Uses legitimate Windows APIs to remove user-mode monitoring hooks.

⇒ **Mechanism:** Calls functions like SetWindowsHookEx or NtProtectVirtualMemory to tamper with EDR telemetry DLLs.

⇒ **Security Impact:** EDR sensors are disabled or blinded, allowing unrestricted malware activity.

9. Fileless LOLBIN Abuse (e.g. rundll32, mshta)

Executes malicious code via trusted Windows binaries.

⇒ **Mechanism:** Uses built-in tools to execute scripts, payloads, or remote content entirely in memory.

⇒ **Security Impact:** Evades AV/EDR detection and leaves no forensic evidence on disk.

10. BYOVD — Bring Your Own Vulnerable Driver

Abuses signed, vulnerable drivers to gain kernel-level access.

⇒ **Mechanism:** Installs a known-vulnerable driver (e.g., aswArPot.sys) to disable EDR or dump memory.

⇒ **Security Impact:** Grants full kernel access, disables AV/EDR from the inside, and facilitates stealth persistence.

Strategic Recommendations

Defending the Defender: Evasion Countermeasures

Modern adversaries are not simply exploiting vulnerabilities — they are exploiting overconfidence in traditional detection approaches. To counter the rise of evasive and EDR-killing malware, organisations must adopt a multi-layered, adversary-aware strategy that goes beyond prevention and real-time alerts.

1. Validate the Effectiveness of Endpoint Controls — Don't Assume

- Periodically test whether EDR, Defender, and AMSI components are still actively protecting the host.
- Use breach validation tooling or adversary simulation (e.g. Atomic Red Team, SCYTHE) to confirm agent integrity and detection efficacy.
- Integrate continuous control validation into your SOC workflow — "is it running?" is not the same as "is it working?"

2. Complement Threat Detection with Independent Breach Detection

- Assume controls will fail — detect evidence of compromise, not just suspicious behaviour.
- Deploy tools or services that inspect memory, processes, binaries, registry, and persistence mechanisms independently of logs or alerts.
- Apply forensic-grade validation at scheduled intervals or after high-risk events (e.g. privileged access, suspicious persistence, zero-day exposure).

3. Harden the EDR/AV Agents Themselves

- Ensure endpoint protection components (EDR, AMSI, Defender) cannot be disabled or uninstalled by standard users.
- Monitor tamper protection settings, privilege assignments, and service ACLs (e.g. `sc sdshow` for service permissions).
- Implement application whitelisting and WDAC/AppControl with strong code integrity policies to restrict loader abuse.

4. Monitor for EDR Health Degradation

- Use telemetry to flag changes in agent state (e.g. no new telemetry, service stopped, version rollback).
- Set alerts for agent uninstall events, driver deregistration, or changes in cloud policy baselines.
- Ensure EDR dashboards reflect actual device enforcement posture, not just check-in status.

5. Isolate and Instrument High-Value Systems

- Treat domain controllers, privileged user machines, and jump boxes as Tier 0 assets.
- Deploy additional telemetry and breach detection tooling on these systems to catch what EDR may miss.
- Consider active deception (e.g. canary services, decoy credentials) to detect lateral movement or evasive malware on critical assets.

6. Reassess Exposure from EDR Killers and BYOVD

- Maintain a list of known vulnerable drivers and block them via group policy or third-party tools (e.g. Microsoft's recommended driver block list).
- Detect and block driver load events not signed by your OEM or your own code -signing certificate.
- Consider kernel-level protections such as HVCI (Hypervisor-protected Code Integrity) to block untrusted driver loads.

7. Train Defenders on Evasion Tradecraft

- Equip your team to understand how evasion works — not just what it looks like post-fact.
- Red team playbooks should include modern tooling like Zig Strike, Brute Ratel, or Havoc with evasive profiles enabled.
- Create internal threat modelling scenarios where EDR is assumed to be by-passed or disabled.

8. Adopt the Attacker Mindset

Understand How You Would Evade Your Own Controls

Security teams must go beyond passive configuration and adopt an offensive mind-set. Ask yourself: *If I were an attacker with system-level access, how would I disable or evade my organisation's EDR?*

This mindset encourages deeper knowledge of your own environment — its de-pendencies, weaknesses, and blind spots. For instance:

- A local admin can silently apply a Windows Defender Firewall policy to block outbound telemetry to EDR cloud consoles.
- Signature updates, alert delivery, and containment actions will fail — and un-less explicitly monitored, the agent will appear healthy and connected.
- Similarly, an attacker could use host-based DNS poisoning or proxy manipula-tion to disrupt EDR operations without triggering alerts.

By thinking this way, defenders can:

- Identify assumptions that attackers exploit (e.g. trusting EDR status indica-tors).
- Create validation controls that detect tampering and silent degradation.
- Design response playbooks that account for telemetry loss or endpoint isola-tion.

Final Thought

Rethinking Trust in Endpoint Security

The illusion of control is the most dangerous vulnerability in cybersecurity. As this report has demonstrated, modern adversaries no longer need zero-days to succeed — they simply need to understand the gaps defenders don't test. From silent evasion to full-scale EDR termination, offensive toolkits are now engineered to exploit *trust* as the weakest link.

EDR and Next-Gen AV solutions, despite their critical role in enterprise defence, are still just software. Like any software, they are:

- Vulnerable to misconfigurations and flaws
- Susceptible to exploitation
- Prone to failure under targeted pressure

But here's the difference: unlike your CRM or your email client, your EDR solution is your last line of defence. And unlike every other application, it must not only protect the system — it must protect itself. No other control will defend it.

It's no longer enough to deploy best-in-class tools and assume protection. Security leaders must adopt a mindset of continuous validation, adversary emulation, and breach assumption. Only by actively challenging the integrity of your endpoint controls — through independent validation, threat-informed defense, and evidence-based breach detection — can you regain strategic ground.

True resilience doesn't come from believing your tools work. It comes from proving they can't be silently bypassed.

The Path Forward

Security leaders must move beyond the checkbox mentality of "agent deployed = risk managed." It is no longer sufficient to *believe* your tooling is working — you must continuously prove it cannot be bypassed or disabled without detection.

That means:

- Validating EDR resilience under hostile conditions, not lab conditions
- Running breach detection that doesn't rely on logs, agents, or assumptions
- Thinking like the attacker — not just to stop them, but to stop thinking you're safe when you're not

The most dangerous breach isn't the one that triggers alerts — it's the one where everything looks normal, but nothing is working.

References

- https://www.forbes.com/sites/daveywinder/2025/03/31/hackers-bypass-windows-defender-security-what-you-need-to-know/
- https://s.itho.me/ccms_slides/2025/4/22/b74f7f14-1855-4a86-9233-0f43ab3b82c8.pdf
- https://www.vaadata.com/blog/antivirus-and-edr-bypass-techniques/
- https://kpmg.com/nl/en/home/insights/2025/01/bypassing-microsoft-defender-for-endpoints-common-language-runtime-detection.html
- https://www.ibm.com/think/x-force/bypassing-windows-defender-application-control-loki-c2
- https://www.hackmosphere.fr/en/bypass-windows-defender-antivirus-in-2025-evasion-techniques-using-direct-syscalls-and-xor-encryption-part-2/?_gl=1%2A1fkv9d4%2A_up%2AMQ..%2A_ga%2AMjEyMjk3NzM0Mi4xNzUzMjM5MjU0%2A_ga_B4E47PLTTC%2AczE3NTMyMzkyNTMkbzEkZzAkdDE3NTMyMzkyNTMkajYwJGwwJGgw

www.cyberstash.com

Act now to protect your business!

Contact CyberStash today to learn about eclipse.xdr and our round-the-clock managed detection and response service.