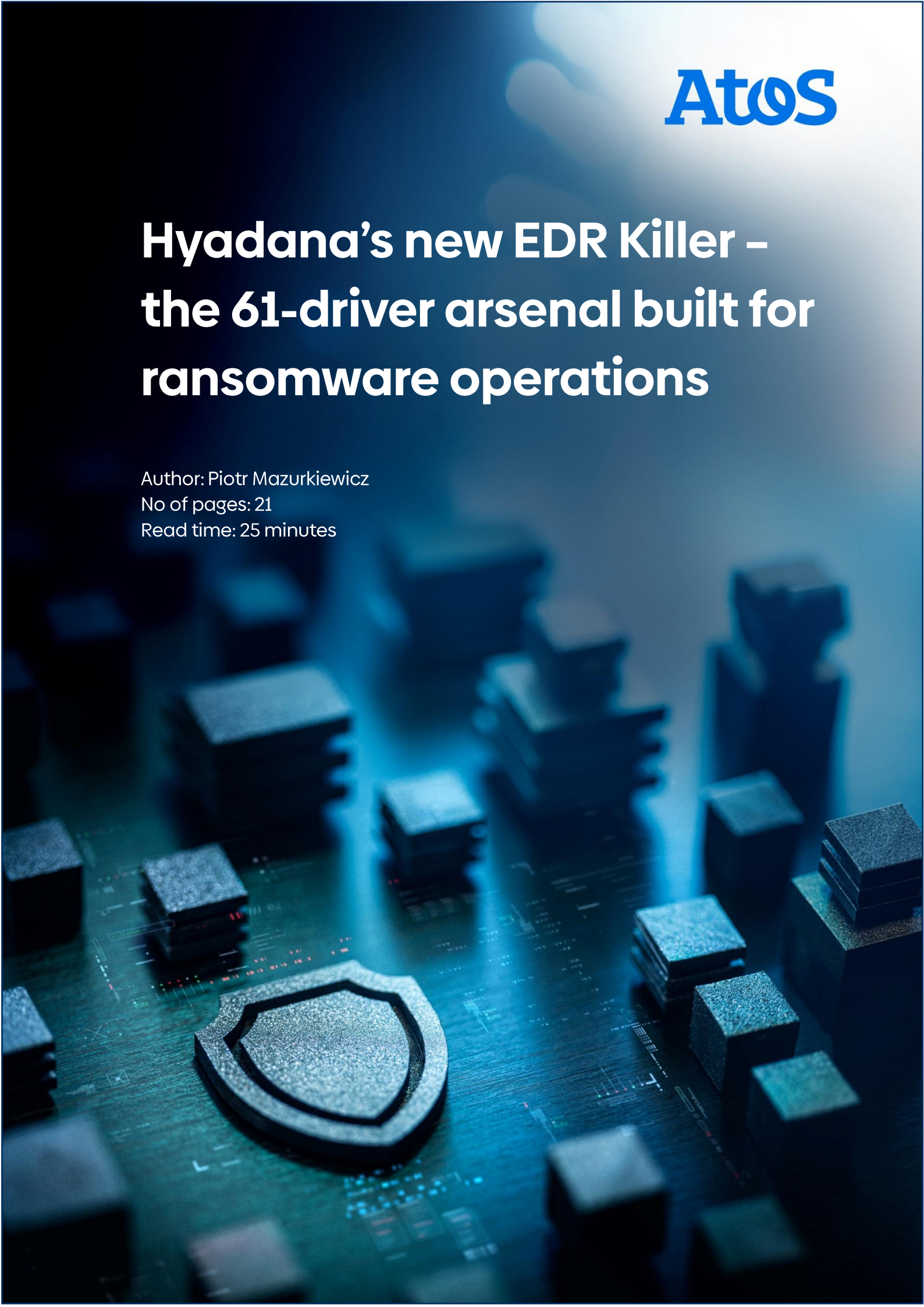
# Hyadana's new EDR Killer – the 61-driver arsenal built for ransomware operations

Author: Piotr Mazurkiewicz
No of pages: 21
Read time: 25 minutes



# Contents


1 Introduction ..... 3
2 From “Monster” and “Beast” to “GodDamn” ..... 4
3 Why vulnerable drivers matter ..... 5
4 “edr.exe” the swiss-knife of BYOVD ..... 7
4.1 A randomizer driver portfolio ..... 7
4.2 Driver-specific process termination ..... 9
4.3 A target list that can change without recompilation ..... 9
4.4 Continuous suppression ..... 9
4.5 What the driver catalog tells us ..... 10
5 The EDR killer is one stage, not the objective ..... 12
6 The ESXi impact chain ..... 13
7 Detection opportunities ..... 14
8 Research summary ..... 15
8.1 Analytical limitations ..... 15
8.2 Conclusion ..... 15
9 Indicators of Compromise ..... 16
9.1 Selected Hashes ..... 16
9.2 List of drivers included ..... 16

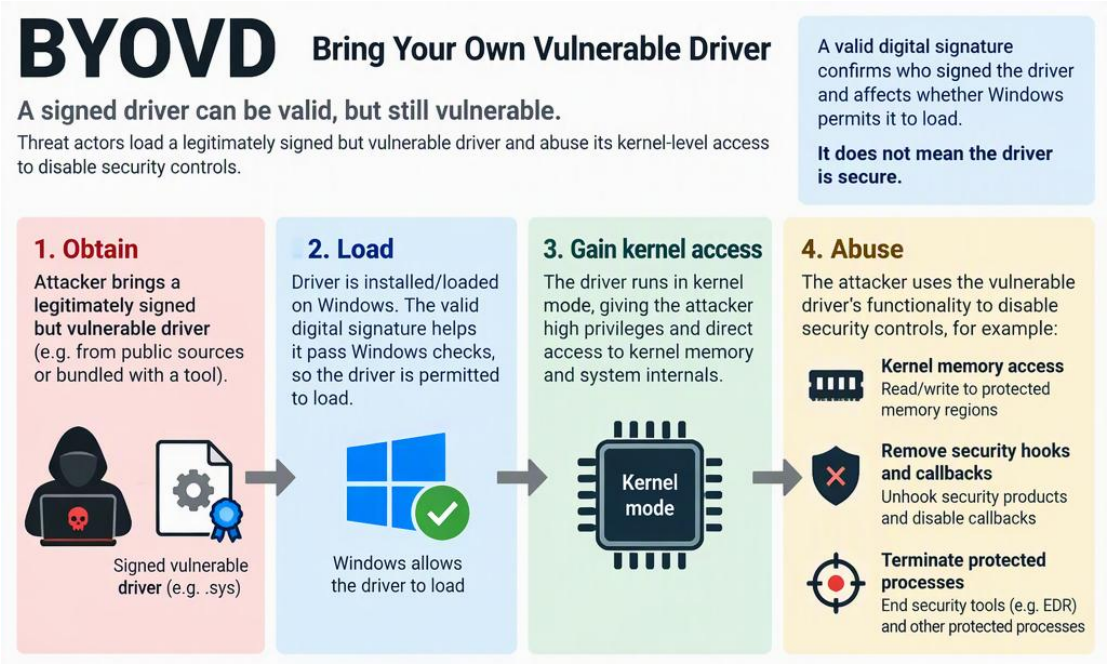
# 1 Introduction

Atos Threat Research Center analyzed a newly recovered EDR-killing tool associated with a GodDamn/Hyadina ransomware operation, revealing a significant expansion of the group's defense-evasion capabilities.

The tool, edr.exe, embeds 61 kernel drivers and randomizes 59 active driver configurations to increase the likelihood of finding a vulnerable driver that can be successfully loaded on a compromised Windows system.

Rather than relying on a single BYOVD technique, the tool uses a broad driver portfolio as a compatibility strategy against different Windows versions, security configurations, blocklists, and enforcement policies.

The built-in configuration targets hundreds of EDRs, antivirus, monitoring, backup, and administrative processes, while command-line options allow operators to replace the default list without recompiling the tool.




The EDR killer forms part of a wider operator-driven ransomware workflow involving credential theft, redundant remote access, infrastructure reconnaissance, security-control disruption, and direct attacks against VMware ESXi environments.

Recovered deployment scripts show operators moving the final impact stage below the guest operating system, stopping workloads and encrypting *-flat.vmdk virtual disks directly on ESXi hosts

The findings show why hash and filename based detections are not enough. Defenders should also monitor vulnerable driver loading, security process termination, privileged management activity, and suspicious activity within virtualization infrastructure.

# 2 From “Monster” and “Beast” to “GodDamn”

According to Symantec[1], which tracks the operation as Hyadina, the group’s ransomware has evolved through three known stages. Monster emerged in 2022 as the first observed version. Beast followed in 2024, expanding the operation’s scope and capabilities. GodDamn represents the latest known iteration and continues the development of the same ransomware operation.

| Period | Hyadina iteration | Notable development |
|---|---|---|
| March 2022 | Monster | Windows-focused ransomware, affiliate model, CIS countries avoidance |
| June 2024 | Beast | Linux and ESXi support, improved encryption and operator control |
| June 2025 | Beast evolution | Broader process-killing and defense-disabling toolset |
| 2026 | GodDamn | Continued defense evasion, PoisonX use, and Beast code lineage |
| **October 2026** | **GodDamn-associated workspace** | **Multi-driver EDR killer and operator-driven ESXi impact chain** |

[1] Symantec article regarding GodDamn/Hyadana: https://www.security.com/threat-intelligence/goddamn-ransomware-beast-rebrand

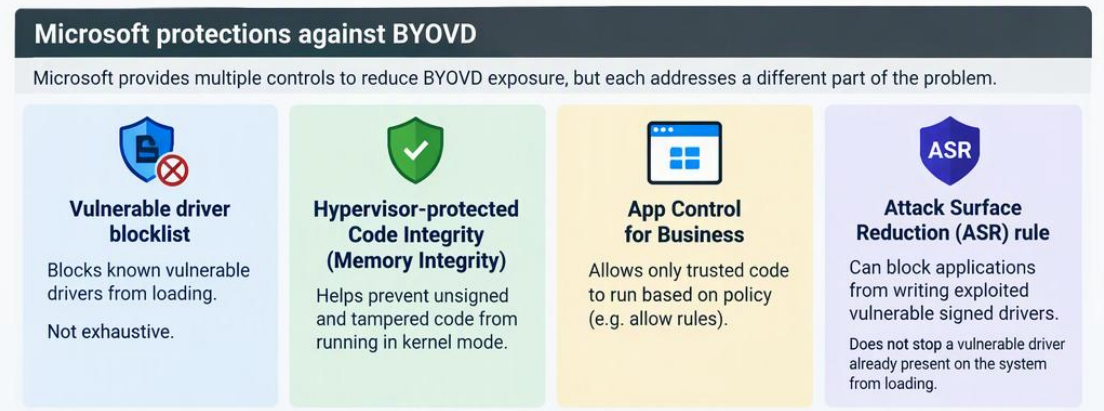
# 3 Why vulnerable drivers matter

Endpoint security products operate within Windows security boundaries. Even with administrative privileges, a user-mode process often cannot terminate protected security processes. Kernel drivers operate with higher privileges and provide access to functions unavailable from user mode.

**Bring Your Own Vulnerable Driver** (BYOVD), attacks exploit this difference. An operator loads a legitimately signed but vulnerable driver and abuses its kernel-level access to disable security controls. Typical actions include kernel memory access, removal of security hooks and callbacks, and termination of protected processes.




**A valid digital signature does not mean a driver is secure.** It confirms the signing relationship and affects whether Windows permits the driver to load. **Threat actors exploit the gap between signed and secure.**




Microsoft provides several protections against BYOVD, including the vulnerable driver blocklist, Hypervisor-protected Code Integrity, App Control for Business, and an Attack Surface Reduction rule designed to block applications from writing exploited vulnerable signed drivers. These controls reduce BYOVD exposure, but each addresses a different part of the problem. Blocklists are not exhaustive, policies differ between systems, and compatibility requirements affect enforcement. An ASR rule focused on driver creation also does not stop a vulnerable driver already present on the system from loading.

The newly recovered tool addresses this uncertainty by carrying dozens of vulnerable drivers instead of relying on a single option.

# 4 “edr.exe” the swiss-knife of BYOVD

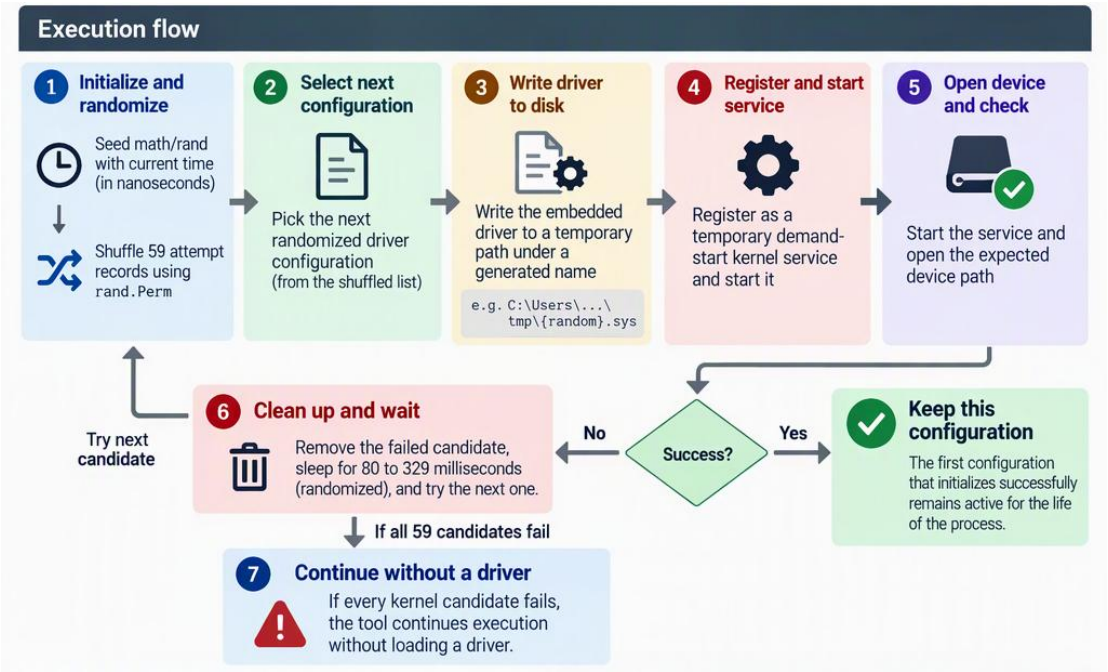
The analyzed sample is identified by SHA-256:
*14507130b07bba7199f96dc3da7aedfd230ec3bc3188378293cfb4c3314320c8*

Static analysis identified edr.exe as an unsigned PE32+ x64 Go console executable. The binary contains no PE resource directory, debug directory, or trustworthy linker timestamp. No application-level beaconing, command polling, file transfer, listener, or remote task protocol was identified. Go networking symbols are present, but they originate from linked runtime and library components rather than confirmed network functionality.

The sample has a focused purpose: initialize a process-killing driver, enumerate running processes, and repeatedly terminate processes selected by the operator.

## 4.1 A randomizer driver portfolio

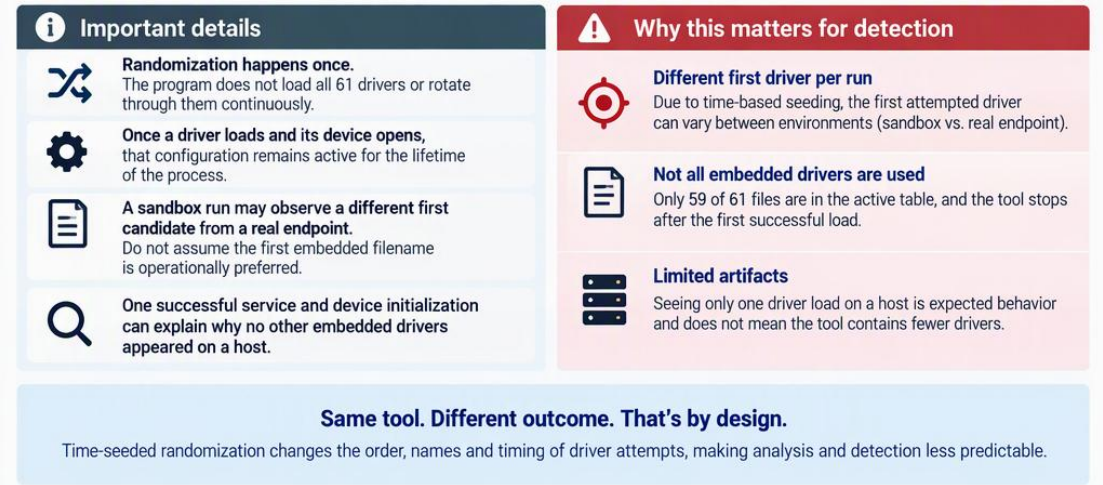
The Go embedded filesystem contains 61 PE files with valid MZ headers. The active configuration defines 59 driver attempts referencing 58 files. Three additional drivers are embedded but inactive in this build, while one active driver appears in two service configurations.




At startup, the tool randomizes all 59 driver configurations using Go’s *math/rand.Perm*. It then writes each selected driver to a temporary path under a generated name, registers it as a demand-start kernel service, starts the service, and attempts to open the expected

device. If initialization fails, the tool removes the candidate, waits between 80 and 329 milliseconds, and tries another. The first driver that initializes successfully remains active for the lifetime of the process. If all candidates fail, execution continues without a kernel driver.

The tool does not load all embedded drivers or continuously rotate between them. This matters during analysis because different executions might select different drivers first. A sandbox might therefore observe a different candidate than a compromised endpoint. A single successful service and device initialization also explains why other embedded drivers might never appear on the system.




The pseudorandom generator uses the current time in nanoseconds as its seed. The same mechanism controls driver order, generated names, and retry delays. This introduces execution variation but does not provide cryptographic unpredictability.

## 4.2 Driver-specific process termination

Each active driver configuration links four values: the embedded driver, device path, IOCTL control code, and request structure defining how the target PID is passed. The active set includes 17 ardrv samples, 18 PoisonX samples, and 24 records from other driver families.

When a matching process is found, edr.exe opens the active device and sends the configured IOCTL request. If the driver-based termination fails, the tool opens the target with PROCESS_TERMINATE and calls TerminateProcess. This fallback provides a secondary termination method when kernel-level termination fails.

A failed process termination does not trigger driver rotation. Once a driver initializes successfully, edr.exe keeps using the same configuration even when it fails against a specific PID. The tool then attempts user-mode termination and retries matching processes during later sweeps.

## 4.3 A target list that can change without recompilation

The built-in target list contains 406 entries, including 405 exact spellings and 393 unique process names after case-insensitive normalization. The list covers EDR, antivirus, monitoring, backup, and administrative software.

Two command-line options allow operators to replace the default targets. The -n option selects a single process, while -l loads process names from a text file. If both options are present, -n takes priority. Without either option, edr.exe uses the embedded list.

The file loader removes whitespace, ignores empty lines and comments, removes case-insensitive duplicates, and adds the .exe extension when missing.

The embedded list therefore represents the default configuration, not a fixed targeting scope. Operators can supply custom process names, reducing the value of detections based only on the embedded target list.

## 4.4 Continuous suppression

After each process sweep, edr.exe sleeps for a randomized 60 to 209 milliseconds. Process enumeration and termination add some execution time, but the short interval still supports hundreds of sweeps per minute on a responsive endpoint.

The tool tracks successfully handled PIDs. Failed targets remain eligible for another termination attempt, while security services that restart under a new PID are detected during later sweeps.

This behavior makes edr.exe more than a one-time security-process killer. It continuously attempts to suppress targeted security processes while the operator performs follow-on activity.

## 4.5 What the driver catalog tells us

Atos Threat Research Center extracted and hashed all 61 drivers without loading them. Exact, case-insensitive SHA-256 comparison against a downloaded LOLDrivers JSON snapshot produced the following result:

| Measurement | Result |
|---|---|
| Embedded files | 61 |
| Unique hashes | 60 |
| Files matching LOLDrivers exactly | 59 |
| Unique matching hashes | 58 |
| LOLDrivers parent records represented | 24 |
| Exact nonmatches | 2 |

All 58 unique exact matches were classified as vulnerable-driver samples in that catalog snapshot. The two nonmatching files were HNOs2Ec.sys and MonProcess.sys. Both drivers have appeared in less prominent vulnerable-driver databases since September 2026. VirusTotal also shows low detection rates, with only 6/71 and 1/70 security vendors flagging the respective samples, marked mostly as PuPs.

| Embedded name | SHA-256 |
|---|---|
| HNOs2Ec.sys | ebcb856723bd023c7ecb40fc71b3ff2ed3795fa08313f397569866167655dadf |
| MonProcess.sys | 8a8652604f7789a6259ae05266652580b18729e1f1c05612b9d338eb8379ecee |

The matching set included all 18 PoisonX files, all active ardrv variants, and numerous other drivers. Three matched files, ASTRA64.sys, ktapi.sys, and xhunter1.sys, were embedded but inactive in this build.

Catalog metadata also illustrates why defenders should validate controls rather than infer them. Only three of the 58 unique matched hashes were explicitly recorded as loading despite HVCI; 23 were recorded as not loading and 32 had no value. Those fields report catalog observations, not tests performed in this investigation, and they cannot predict behavior across every Windows build, blocklist revision, certificate state, and local policy.

The portfolio is best understood as compatibility engineering for attackers. Different Windows versions and security configurations may reject different drivers, so the tool searches for whichever privileged interface remains available.

# 5 The EDR killer is one stage, not the objective

The broader dataset came from two separate threat-actor workspaces. One functioned mainly as a payload staging and execution environment. The second contained a larger operational workbench with reconnaissance results, credentials, VMware and storage tooling, network-device configurations, remote-access artifacts, and victim-derived data.

The recovered material documents a multi-stage operation. The operators tested access to exposed infrastructure and remote services, mapped Active Directory, storage, virtualization, backup, and network systems, and collected credentials from memory, domain databases, configurations, shares, and virtual disks. They established additional access through remote administration tools, SSH, web shells, tunnels, and modified infrastructure. The operation then progressed to endpoint-security disruption with edr.exe and impact against ESXi workloads.

The workspace documents a multi-stage operation:

1. Test or obtain access to Internet-facing infrastructure and remote services.
2. Enumerate Active Directory, storage, virtualization, backup, and network systems.
3. Harvest credentials from memory, domain databases, configurations, shares, and virtual disks.
4. Establish redundant access with remote administration, SSH, web shells, tunnels, and modified infrastructure.
5. Impair Windows endpoint defenses with edr.exe.
6. Stop workloads and encrypt ESXi virtual disks.

The strongest persistence and pivoting evidence included FortiGate IPsec configurations, a strongSwan watchdog, and Chisel activity. Credential-access artifacts included NTDS and LSASS data, Kerberos material, application secrets, private keys, and tools used to recover Linux and Windows credentials from guest VMDKs.

The evidence shows an infrastructure-focused extortion operation. The operators prioritized systems that provided control over broader parts of the victim environment.

# 6 The ESXi impact chain

The final impact stage centered on locker_esxi, identified by SHA-256:
*470d681572ffde9b13e680d069be86bbba37120e4855e64efadc7aeb7cdf699d*

The surrounding `blitz_*` and `launch_*` scripts show how the operators deployed the locker across ESXi infrastructure. blitz_push.sh transferred launchers to ESXi hosts through SSH. blitz_local.sh started encryption against identified datastores. launch_all.sh searched for VM directories containing `*-flat.vmdk` files and started one locker process per selected VM. launch_stale.sh retried targets left unfinished or inaccessible during earlier attempts.

The operators targeted `*-flat.vmdk` files, which contain virtual disk data. Encrypting these files at the ESXi layer disrupted guest systems while operating below security controls installed inside the virtual machines.

The workflow also shows direct operator supervision. Separate scripts handled broad deployment, local execution, remote delivery, inactive hosts, and failed targets. Comments and exclusions indicate that operators reviewed progress and adjusted targets during the operation.

The recovered sequence shows a coordinated impact phase. edr.exe suppressed Windows security processes while the operators consolidated credentials and access to management infrastructure. The ESXi tooling then moved impact below the guest operating system and targeted virtual disks directly.

# 7 Detection opportunities

## Windows driver and process behavior

Defenders should prioritize behavior chains over one filename:

- A previously unseen unsigned executable enables SeDebugPrivilege, enumerates processes at subsecond intervals, and repeatedly requests PROCESS_TERMINATE.
- A process writes a kernel driver to a temporary or user-writable path, creates a short-lived demand-start service with a generated name, starts it, and opens a new device.
- The dropped file disappears while the kernel service or driver object remains observable.
- Multiple unrelated security agents stop, fail, or restart in a short period.
- A service restart receives a new PID and is terminated again seconds later.
- Base64 decoding, archive extraction, or chunk reconstruction produces the known edr.exe hash.
- Command lines use -n or -l; referenced target-list files should be preserved immediately.

Telemetry should include process creation with full command lines, driver and service installation, Code Integrity events, file creation in temporary paths, device access where available, privilege adjustment, security-service state changes, and process termination failures.

## Virtualization and management plane

Hunt for:

- unexpected SSH enablement or root logins on ESXi;
- sshpass, SCP, or shell scripts distributing files into /tmp;
- unknown binaries opening many *-flat.vmdk files;
- VM shutdowns followed by high-volume datastore writes;
- many concurrent locker-like processes, especially with --ultrafast;
- new ESXi users, authorized keys, startup scripts, or altered execution controls;
- anomalous VMware Guest Operations requests or direct access to VMX process memory;
- broad reads of VMDKs for /etc/shadow, SSH keys, NTDS.dit, SAM, or SECURITY.

Endpoint, identity, firewall, backup, vCenter, and hypervisor telemetry must be correlated. An intrusion that reaches the virtualization layer cannot be scoped from one Windows endpoint.

# 8 Research summary

## 8.1 Analytical limitations

TRC conducted the research statically and offline. No recovered executable, driver, locker, script, archive, or web shell was executed, and no identified infrastructure was contacted.

The evidence provides stronger insight into tooling design and operator intent than confirmed execution. The presence of a staged file does not confirm its execution. A process listed as a target does not confirm the related product was installed or terminated. A vulnerable-driver match does not confirm successful driver loading. Endpoint telemetry is required to determine which driver, payload, or technique was used on a specific system.

Attribution also has clear limits. The recovered infrastructure, filenames, tooling, and impact workflow are consistent with a GodDamn ransomware operation. Public research associates GodDamn with Hyadina, but the available evidence does not identify the operators, separate affiliates from developers, or establish that all recovered material belonged to a single intrusion set.

## 8.2 Conclusion

Hyadina's development shows steady adaptation across several generations of ransomware. Monster emerged as a Windows-focused operation. Beast expanded the target scope to Linux and VMware ESXi, improved encryption performance, and introduced greater control over deployment. Later Beast operations also placed more emphasis on disabling security products before ransomware execution. GodDamn continued this progression, retaining links to the earlier codebase while expanding defense-evasion techniques and supporting tooling.

The analyzed edr.exe sample represents a further step in this development. It carries 61 drivers, randomizes 59 active configurations, supports custom process targets, continuously attempts termination, and falls back to user-mode process killing.

The broader operation shows the purpose behind this capability. The operators collected privileged credentials, maintained multiple access paths, suppressed security controls, and targeted virtual disks from the ESXi layer.

For defenders, blocking individual hashes is insufficient. Detection should cover vulnerable driver loading, security-process termination, privileged management access, and activity across endpoint, identity, backup, network, and virtualization infrastructure.

# 9 Indicators of Compromise

## 9.1 Selected Hashes

| File | SHA-256 | Short description |
|---|---|---|
| edr.exe | 14507130b07bba7199f96dc3da7aedfd230ec3bc3188378293cfb4c3314320c8 | EDR killer with included pack of vulnerable drivers |
| locker_esxi | 470d681572ffde9b13e680d069be86bbba37120e4855e64efadc7aeb7cdf699d | Script that finds and locks all VMDKs |
| cmd_shell.jsp | 72bd1876399767551f2164f707fadbaa41a9772a29307665ad3c119d23e78860 | Reverse shell |

## 9.2 List of drivers included

Below is a list of all vulnerable drivers embedded in edr.exe. All of them are already marked as vulnerable in popular trackers for vulnerable drivers.

| Embedded name | SHA-256 | Use in this edr.exe build |
|---|---|---|
| ardrv.sys | 07c5209bf83065fe760f4fee4ed2308b0c523671f68ca73a3854c2c8c28c0541 | Active |
| ardrv_1.sys | 0b4c5a2d24810748de8ce6fb3d6a3e8acd0305272d6425fa054632410d23ec80 | Active |
| ardrv_2.sys | 1c5129b4d63f2d5252b5b02a0a7d655dac3b6eac23dd4ed92b4dd55f243387ff | Active |
| ardrv_3.sys | 4d07b6423f4d5d80d2daec8faed7ecc40855339bee95c26289179c1fd083078e | Active |
| ardrv_4.sys | 6b208ce68ba3f6c788bd89ba677ebf1c2407a55f9f43a42ea7207635634baf66 | Active |
| ardrv_5.sys | 07c5209bf83065fe760f4fee4ed2308b0c523671f68ca73a3854c2c8c28c0541 | Active |
| ardrv_6.sys | 7b3f8fa5844d829142002d47c2df8573c7d67defe4171414c6a015a38afe55f5 | Active |
| ardrv_7.sys | 9f326ede322930a7d5e149fa8720802b0623c1972dc54e41a2a73608fb6810dd | Active |

| | | |
|---|---|---|
| ardrv_8.sys | 34ad901b4fea05da301ebb619fa48cfac058720e41db71b3fca1c47763bae066 | Active |
| ardrv_9.sys | 4574cf50dbe2fbce849881e8875297753ce2e240c312e01b3b461d80df063bbe | Active |
| ardrv_10.sys | 75454a8583ea2a5b6c7711e9bf32c61b8a7e902f5c429fcbe36673ccc076e910 | Active |
| ardrv_11.sys | 551254a5e59df92e6049867162633e3e476cdf5999d05c3e23a57c98be03f475 | Active |
| ardrv_12.sys | 7504887e1e195ad585cffa5b6a5034161a7cc49f351123d60def79302bdb8326 | Active |
| ardrv_13.sys | 33233988d8c735fdf1162e043ed1e6fff62e7d6102c5bd300561d71fd32526f3 | Active |
| ardrv_14.sys | 59080957de2519f5013be8ccf0b9012dc97ad692e69578cb3f60148c528bdbb1 | Active |
| ardrv_15.sys | d82108e6578f4866d7114d6847282c04f0b146809bc66ded3300fe993b42d20f | Active |
| ardrv_16.sys | e3cb9f52a4fde53fc886dc35d41bb36e8b993c74826389d22440b92d94480a6d | Active |
| ASTRA64.sys | 4a8b6b462c4271af4a32cf8705fa64913bfcdaefb6cf02d1e722c611d428cb16 | Embedded but inactive |
| BdApiUtil64.sys | 47ec51b5f0ede1e70bd66f3f0152f9eb536d534565dbb7fcc3a05f542dbe4428 | Active |
| CcProtect.sys | 5f0cfe8357bb52b45068ddbac053e32bc38e6cb5e086746f5402657b0a5cfb1c | Active |
| EnPortv.sys | d42f1b420747b82533e33107c710c45c29ff20aa5da3d1c8498b7bed7f9ebc81 | Active |
| GameDriverX64.sys | 9ddae47ff968343a8c32a5344060257fdc08e2a7bdb9a227c8b3a584ee3c9f1e | Active |
| GoFly64.sys | 2fdfdd13a0c548bb68c9d5aa8599a9265d4659da3e237fe7a42ac6ac06b9a06a | Active |
| HNOs2Ec.sys | ebcb856723bd023c7ecb40fc71b3ff2ed3795fa08313f397569866167655dadf | Active |
| HWAudioOs2Ec_1 | 5abe477517f51d81061d2e69a9adebdcda80d36667d0afabe | Active |

| | | |
|---|---|---|
| .sys | 103fda4802d33db | |
| HWAudioOs2Ec_2.sys | 90d2e9e994ed8e964845a26dce741ad43b29ff54cf5faa67271d62d4e24acbc8 | Active |
| K7RKScan_1516.sys | b16e217cdca19e00c1b68bdfb28ead53b20adeabd6edcd91542f9fbf48942877 | Active |
| K7RKScan_2310.sys | 5c6ce55a85f5d4640bd1485a72d0812bc4f5188ee966c5fe334248a7175d9040 | Active |
| ksapi64.sys | 1cd219f58b249a2e4f86553bdd649c73785093e22c87170798dae90f193240af | Active |
| ksapi64_del.sys | 26ed45461e62d733f33671bfd0724399d866ee7606f3f112c90896ce8355392e | Active |
| ktapi.sys | 7ee17efef04bb7c9de90d5210263ed6993f867e5a11f86e65e3bb1362c7de237 | Embedded but inactive |
| MonProcess.sys | 8a8652604f7789a6259ae05266652580b18729e1f1c05612b9d338eb8379ecee | Active |
| MonProcessEX.sys | 72d0b5615b996cbb01b1ca139e627079094f734da48a0435ffd8480a25d0a258 | Active |
| NSecKrnl.sys | 206f27ae820783b7755bca89f83a0fe096dbb510018dd65b63fc80bd20c03261 | Active |
| PCTcore64.sys | c76dd87891e3e30db2aed057e7b04c19ca264f434c21f68a7a2d9b17a97aff39 | Active |
| PoisonX.sys | a5035cbd6c31616288aa66d98e5a25441ee38651fb5f330676319f921bb816a4 | Active |
| PoisonX2.sys | b1ec23cf04e9ea0672f18486601e798c882b0b981772f6bca5c144d2f2af9889 | Active |
| PoisonX3.sys | 4948e89b532804590490aaae41f4b582a89592c931e557b6ddfff0b8d6ee8cf5 | Active |
| PoisonX4.sys | 83db6a37d9ec9923ca2aa677b4f4d8b67c8b2468046d21136a57ffe92eba6cac | Active |
| PoisonX5.sys | 63eaa6be7c2f220af0f5fdf3b232f18908777262ecc0919796a92438aec74b89 | Active |
| PoisonX6.sys | 40a46b17a8ccb3104b079716d764ceeda1f8431cb6582308f59af5802fe82fda | Active |

| | | |
|---|---|---|
| PoisonX7.sys | 6fbaad2f00afaa94723fa7d5bd46e7ea4babb7ce478a8e7229ce7bd4b85e0f51 | Active |
| PoisonX8.sys | 1ca2f1c00d54b14f0f50d54f11cb654eddcc0228967f9942a92f544c75d5bb37 | Active |
| PoisonX9.sys | db5d284b9a9c02f76030ba89fd85c7c8f830f8fe4195cdc1f9cddf15f127125d | Active |
| PoisonX10.sys | d58df93524ead1a1f939438ef63b5a9e42aacec7463ed29878293382996640ce | Active |
| PoisonX11.sys | cc6b4e34a4c49ce2770615aa38b00ce479e236d34307f950cf5d8ad46b458627 | Active |
| PoisonX12.sys | c070b4a5614cb234de0faaef4e131b0cfe127682dc4cc89856e1ebfe775f4725 | Active |
| PoisonX13.sys | 50870b82104a309c107ad6ed50c023324fd73b6908a52f6070f47b8d247323c9 | Active |
| PoisonX14.sys | 6452ca681dd818a36ec538fe2d83a795f98e14970855964520294726dedd742b | Active |
| PoisonX15.sys | 846b2f282925d7ddccd41f46f0048bc1f424fe44ccb110ed45ece8637fbece5d | Active |
| PoisonX16.sys | 103b1bbe9d257e3ae6b83befdc53cad0e5432ade535b1f85dd6fddd4356d4898 | Active |
| PoisonX17.sys | 6f24ed64cba4ed0901592770a3aead821317a85efa886bd51f1afc6cb1166990 | Active |
| PoisonX18.sys | 1d9ae1467e604469798d272755afcf845b5efcd588863ad9e2aa3e3cf112985a | Active |
| STProcessMonitor_v114.sys | 70bcec00c215fe52779700f74e9bd669ff836f594df92381cbfb7ee0568e7a8b | Active |
| STProcessMonitor_v2618.sys | 5b4f59236a9b950bcd5191b35d19125f60cfb9e1a1e1aa2e4f914b6745dde9df | Active |
| SysMon.sys | 1c1a4ca2cbac9fe5954763a20aeb82da9b10d028824f42fff071503dcbe15856 | Active |
| unknown.sys | 97bd65e98cdc4e93d49edd4ea905d43a61244df0fd3323e6649330de3b1be091 | Active |
| viragt64.sys | 58a74dceb2022cd8a358b92acd1b48a5e01c524c3b0195d70 | Active |

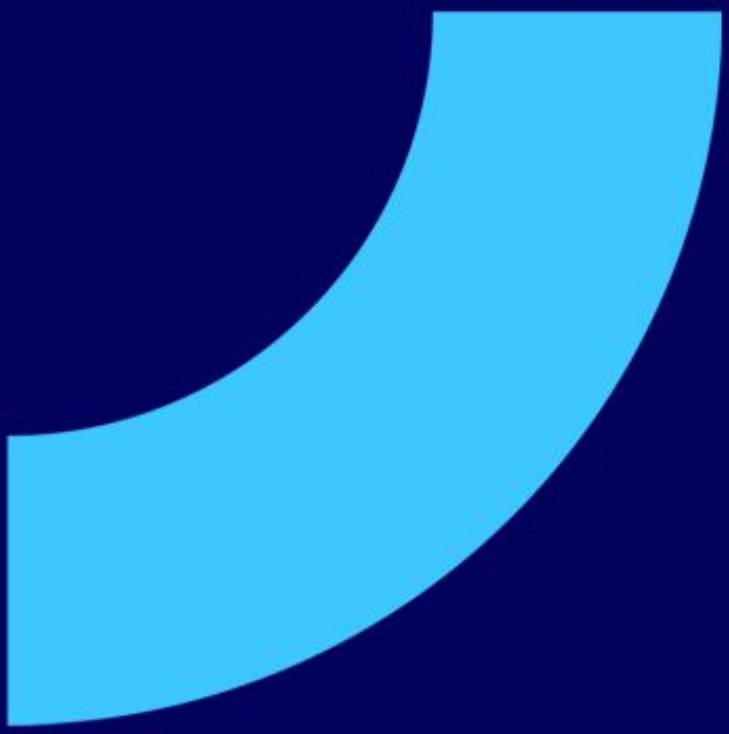
| | | |
|---|---|---|
| | 33e4bd55eff4495 | |
| wsftprm.sys | ff5dbdcf6d7ae5d97b6f3ef412df0b977ba4a844c45b30ca78c0eeb2653d69a8 | Active |
| xhunter1.sys | e727d0753d2cd0b2f6eeba4cea53aa10b3ff3ed2afeb78f545fcf6d840f85c3e | Embedded but inactive |
| xkpsm.sys | 28c5bb735f9fc38e0ebd366978898f0cfcd455e4ff8bf1a321768b09e01ee84d | Active |

## About Atos Group

Atos Group is a global leader in digital transformation with c. 56,000 employees and annual revenue of c. €7.2 billion (at the go-forward perimeter), operating in 54 countries under two brands - Atos for services and Eviden for products and systems. European number one in cybersecurity and a leader in cloud, Atos Group is committed to a secure and decarbonized future and provides tailored AI-powered, end-to-end solutions for all industries. Atos Group is listed on Euronext Paris.

Find out more about us

atos.net
atos.net/career



Atos Group is a registered trademark of Atos Group. © Atos Group. Confidential Information owned by Atos Group, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval of Atos Group.

Atos