

From Armillaria loader to EDR killer

 threatlocker.com/blog/from-armillaria-loader-to-edr-killer

December 10, 2025

Intro

In the criminal sphere of ransomware, Akira is known for its abundance of victims and utilizing its opportunism to achieve its goals. Armillaria continues to adapt to the threat landscape and prove useful for threat actors. The Threat Intelligence team has captured samples of both the Armillaria loader and the EDR killer used by Akira, and this analysis aims to highlight the tactics leveraged to successfully terminate endpoint security vendor products.

Technical analysis

Armillaria loader

ThreatLocker Threat Intelligence observed the delivery of an Armillaria loader variant (“version.dll”) in late November, where the payload content is embedded within the loader file, which is decrypted and executed within the context of a “rundll32.exe” process. Using dynamic analysis, the embedded payload can be identified in an unpacked state by the presence of several new Windows PE headers, and the respective magic bytes “4D 5A” (“MZ”). Dumping the offending memory region to disk provides an intact copy of the second stage.

```
CC CC CC CC CC CC
40 53 48 83 EC 20 65 48
8B 04 25 60 00 00 00 45...
8B 8B E0 00 00 00 4C 39...
C8 41 32 D0 D2 C2 41 C1...
00 72 CE 48 8B 8B E0 00...
00 00 00 49 8B 8B F8 0F...
F8 0F 00 00 48 8B 89 A0...
8A A7 13 14 61 9F B7 C3...
A1 68 07 A5 CF 51 7D E0...
C8 FB 52 0B 5A 6D F6 01...
61 C9 67 EC 46 36 FC 78...
4F AE 01 EE 9A 0E F3 3E...
05 60 8F AF 14 FD 11 3D...
D5 FE AF 26 DE 0F DE 8E...
7F 49 58 09 B7 0D FB CD...
9E EF 94 40 A6 40 50 4A...

; -----
; align 10h
qword_180001030 dq 486520EC83485340h, 450000006025048Bh, 54D538CDB841D233h, 4C0000FF8988B48h
; DATA XREF: DllMain+CFF404o
dq 394C000000E08B8Bh, 333476000000F093h, 0E0838B48C9h, 8A4108148AC2FF41h
dq 0C141C2D2D03241C8h, 118841CA8B4103C8h, 490000F503E88141h, 0F08B3B48C1FFh
dq 0E0888B48CE7200h, 888B485853FF0000h, 0D093FF000000E8h, 60251C8B4C650000h
dq 0FF888B48490000000h, 0A089834800000h, 0C8B48651A740000h, 898B480000006025h
dq 0A0898B4800000FF8h, 1801853FF0000000h, 0C35B20C48348h, 1E9h dup(0)
dq 0C3B79F611413A78Ah, 0E5659EF8FD2D2551h, 663F6944D02C3942h, 0B22FD64E91F94B79h
dq 0E07D51CFA50768A1h, 9B6FA99BF7D1A70Dh, 9CDB1FB57AA1362Fh, 90ED4065DFB5D41Ch
dq 1F66D5A0B52FBC8h, 74229E224E7405Bh, 7F7F523249527769h, 31BF816F71FB5D17h
dq 78FC3646EC67C961h, 0FC3A2A4AD8EA4160h, 2E884566794E27C4h, 52C289957771CB84h
dq 3EF30E9AEE01AE4Fh, 796A5B3E32C44529h, 73431779A83DDF2Fh, 1FD71A7CA8BC6294h
dq 3D11FD14AF8F6005h, 841DABA6310C7ACAh, 0DA38863F87087B2Ah, 80B74AF60B3AAC23h
dq 8EDE0FDE26AFFED5h, 15D0DD2F0BF9C1E9h, 674F08A73DBC83EEh, 11C7B3774C9C702Eh
dq 0CDFB0DB70958497Fh, 5926B2BF11968D26h, 0C945DA257926BEA5h, 0D3CE83C0D9346E8Fh
dq 4A5040A64094EF9Eh, 21C6EE2227DFD2D8h, 489E582CE58B1EF6h, 12BC9DE2AF2E4C21h
```

Figure 1: Embedded content

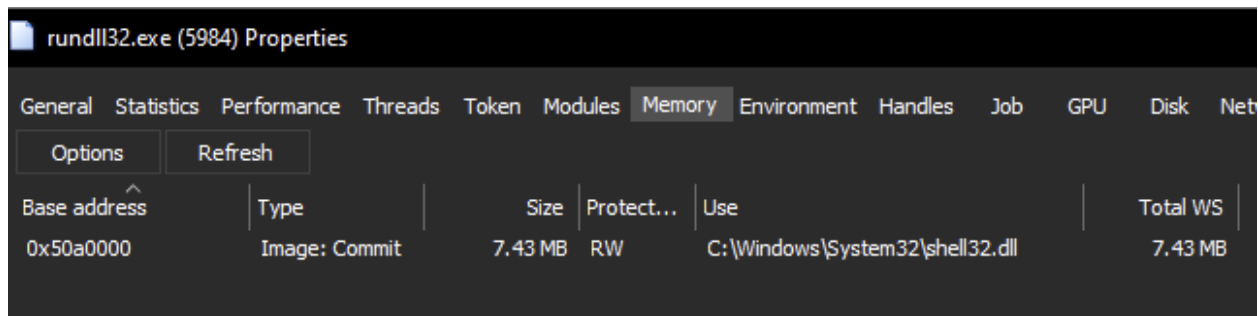


Figure 2: Payload target page

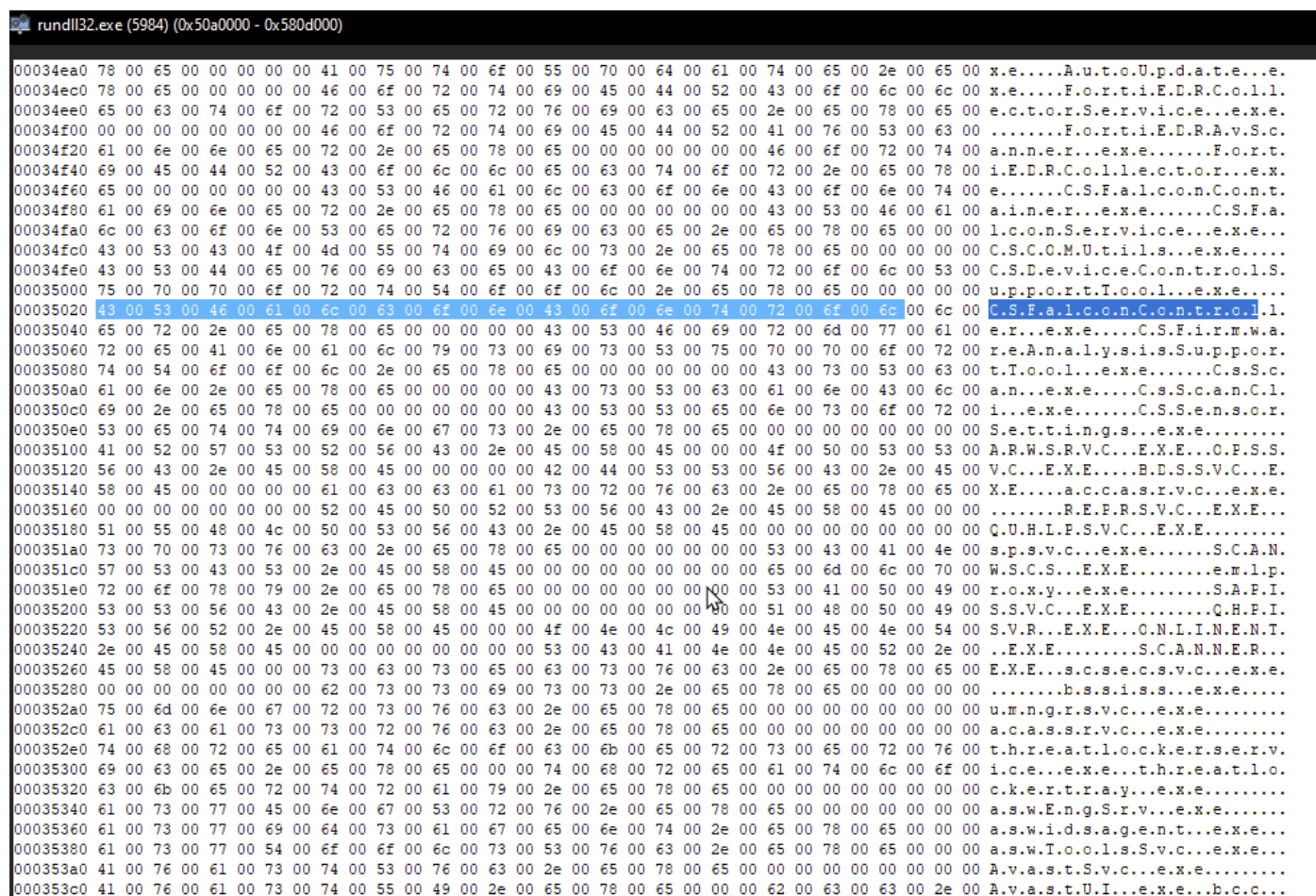


Figure 3: Payload content in memory

Figure 4: Payload in DIE

BYOVD loader & EDR killer

The second stage payload executed within the “rundll32.exe” process by “version.dll” did not match any publicly available samples during initial discovery and has since been shared with the security intelligence community under the filename “owned2.dll.” The primary purpose of this stage is to terminate endpoint security products and their associated services. Termination is achieved through the use of two embedded drivers, “rwdrv.sys”, a known vulnerable driver identified as TechPowerUp’s ThrottleStop driver, and a second unsigned driver “hlpdrv.sys” that is used to terminate target processes and services.

Figure 5: Second stage first sighting

Figure 6: ThrottleStop driver signature

Figure 7: Embedded ThrottleStop driver

Figure 8: Embedded termination driver

Before termination, the second-stage EDR killer payload checks the primary language and locale of the host against a list of protected countries, with the majority of exceptions being members of the CIS. If an excluded locale is detected on the target host, execution halts and the host process is terminated. Simple changes such as changing user keyboard language, or even the entire system language do not change the result of this check, indicating that the value being checked is set by the operating system on OS installation.

Figure 9: Protected locales

Figure 10: Locale comparison

Versions of the ThrottleStop driver 3.0.0 and below expose an IOCTL allowing for arbitrary physical memory read and write through the use of “MmMapIoSpace”, from a user mode application. This vulnerability allows this second stage payload to interact with the created driver device, write to physical memory, bypass driver signature enforcement and load the malicious termination driver “hlpdrv.sys”, without an associated service control manager entry.

Figure 11: ThrottleStop driver written to disk

The malicious termination driver, identified with the filename “hlpdrv.sys” in the user profile temporary directory, exposes IOCTLs. This allows a user mode application to terminate protected or privileged processes through “ZwTerminateProcess” which can be identified with IOCTL code “0x222004”. It is able to modify the DACL (discretionary access control list) and security descriptor for a target object or file, rendering it inaccessible, identified by IOCTL code “0x222000”, and a combination of both, identified by IOCTL code “0x222008”, which is the code leveraged by the second stage payload.

Figure 12: IOCTL code sent to driver

Figure 13: IOCTL code handlers in malicious driver

Figure 14: Process termination

Figure 15: DACL and SD modification

Figure 16: SD before and after

Processes and services to be terminated are stored in hardcoded lists and are passed to the malicious driver with IOCTL code “0x222008” to render their associated files on disk inaccessible, and terminate any active processes. Interestingly enough, the target service list contains entries for

endpoint security vendor drivers, which cannot be unloaded using “ZwTerminateProcess”, reasonably indicating the group responsible for the development of this payload either utilized the target lists from another source or EDR termination tool.

Figure 17: Target processes

Figure 18: Target services

Usage of the Armillaria Loader by Akira operators serves as a testament that as the threat landscape evolves, new tactics emerge, and old ones may resurface. Previously identified Armillaria Loader samples led to the execution of Akira ransomware, and as such allowed ThreatLocker Threat Intelligence to correlate this attack to Akira affiliates, whom were using tools functionally identical to the samples analyzed above to prepare target hosts for encryption.

[Read more about vulnerable drivers and LOTL attacks](#)

Recommendations for ThreatLocker customers

ThreatLocker Detect EDR and Cyber Hero MDR

ThreatLocker Detect EDR can alert your organization to possible ransomware operators' tactics and procedures, including installing ransomware tools, attempting to disable security services, deleting shadow copies, and performing data exfiltration. If you have [Cyber Hero MDR](#), you'll hear from our analysts promptly.

[Learn more about ThreatLocker EDR](#)

Application Control

[Application Allowlisting](#) can block applications that are not explicitly permitted by ThreatLocker or learned during Learning Mode, such as unauthorized Remote Monitoring and Management applications. Additional explicit deny policies can be created to prevent the usage of high-risk applications or software, such as vulnerable drivers, Rclone, MSBuild, or PSEXec. For applications that are high-risk, but are required for business processes, permit policies with [Ringfencing™](#) can be utilized to restrict what resources applications can interact with, such as certain files & directories, internet access, the registry, or executing other applications.

Filename:	SHA-256:
Version.dll (Armillaria Loader)	95A6F6E79C1842CEA3603DF3209FDDC12AEB4FC77D1C58A852F877B1EAA9C4C
Owned2.dll (EDR Killer)	97DFBB7D087F8E297431D80DFC34C255A407F25FEE120931904225E484B9962A

Filename: **SHA-256:**

Rwdrv.sys (ThrottleStop Driver)	16F83F056177C4EC24C7E99D01CA9D9D6713BD0497EEEDB777A3FFFEFA99C97F0
---------------------------------------	---

Hlpdrv.sys (Malicious Driver)	BD1F381E5A3DB22E88776B7873D4D2835E9A1EC620571D2B1DA0C58F81C84A56
-------------------------------------	--

ABOUT THE AUTHOR

Pandeli Zi, threat analyst at ThreatLocker, brings a lifelong curiosity for how systems work, an instinct that started with taking things apart just to see why they functioned the way they did. That curiosity led him into technology, where early exposure to cybersecurity concepts, data security, and digital risk shaped into a bachelor's degree from UCF. Before joining ThreatLocker, Pandeli built technical depth through a range of IT and asset management roles at organizations including Orlando Health, Florida's Turnpike Enterprise, and Morgan & Morgan. At ThreatLocker, he focuses on daily threat tracking, malware research, and post-incident analysis, translating intelligence into practical insights that strengthen customer defenses.

ABOUT THE AUTHOR

William Pires, threat analyst at ThreatLocker, brings a data-first mindset shaped by hands-on work in MDR and customer-facing threat response. He holds a bachelor's degree in accounting and spent two years as a credit analyst before moving into cybersecurity, a path that sharpened his rigor, accuracy, and judgment under pressure. He advanced from approval requests to MDR, then to threat analysis, where he pieces together attack paths from confirmed incidents and researches active threats, group profiles, and emerging tactics. At ThreatLocker, William focuses on translating research into clear detections and practical guidance for customers. His recent work spans software communication analysis, incident triage, and providing cutting edge threat intelligence to enterprise environments and the cybersecurity industry.

ABOUT THE AUTHOR

John Moutos, threat intelligence team lead at ThreatLocker, is a cybersecurity specialist with roots in enterprise penetration testing and offensive security research. John studied at the Oregon Institute of Technology and earned his bachelor's from the SANS Technology Institute. At ThreatLocker, he leads the threat intelligence program, turning research into practical detections and guidance that helps organizations anticipate and counter emerging attacks.