A Bag of RATs: VenomRAT vs. AsyncRAT

By Anna Širokova

Published: 2024-11-21 · Archived: 2026-04-05 21:44:27 UTC

Introduction

Remote access tools (RATs) have long been a favorite tool for cyber attackers, since they enable remote control over compromised systems and facilitate data theft, espionage, and continuous monitoring of victims. Among the well-known RATs are VenomRAT and AsyncRAT. These are open-source RATs and have been making headlines for their frequent use by different threat actors, including Blind Eagle/APT-C-36, Coral Rider, NullBulge, and OPERA1ER. Both RATs have their roots in QuasarRAT, another open-source project, which explains their similarities. However, as both have evolved over time, they have diverged in terms of functionalities and behavior, which affects how attackers use them and how they are detected.

Interestingly, as these RATs evolved, some security vendors have started to blur the line between them, often grouping detections under a single label, such as AsyncRAT or AsyncRAT/VenomRAT. This indicates how closely related the two are, but also suggests that their similarities may cause challenges for detection systems. We took a closer look at recent samples of each RAT to examine how they differ, if at all.

This comparison explores the core technical differences between VenomRAT and AsyncRAT by analyzing their architecture, capabilities, and tactics.

Here's a comparison table between VenomRAT and AsyncRAT based on the findings

Capability	**VenomRAT**	**AsyncRAT**
AMSI Bypass	✔ Patches AmsiScanBuffer in amsi.dll (In-memory patching) T1562.001	✘ Not implemented
ETW Bypass	✔ Patches EtwEventWrite in ntdll.dll (In-memory patching) T1562.006	✘ Not implemented
Keylogging	✔ Advanced keylogger with filtering and process tracking T1056.001	✔ Basic keylogger with clipboard logging T1056.001
Anti-analysis Techniques	✔ Uses WMI for OS detection, VM check T1497.001	✔ VM, sandbox, and debugger detection T1497
Hardware Interaction	✔ Collects CPU, RAM, GPU, and software data using WMI T1082	✔ Collects system data via Win32_ComputerSystem T1082
Process discovery	✔ This the capability to obtain a listing of running processes T1057	✘ Not implemented

Capability	VenomRAT	AsyncRAT
Anti-process Monitoring	✔ Terminates system monitoring and security processes T1562.009	✘ Not implemented
Webcam Access	✔ Camera detection and access T1125	✘ Not implemented
Dynamic API Resolution	✔ DInvokeCore class for dynamic API resolution T1027.007	✘ Not implemented
Encrypts the configuration	✔ 16-byte salt ("VenomRATByVenom") T1027.013	✔ 32-byte binary salt T1027.013
Error Handling	✔ Silent failures with basic try-catch	✔ Sends detailed error reports to C2 T1071

Technical analysis

In this technical analysis, we compare two specific RAT samples:

- **VenomRAT**: 1574d418de3976fc9a2ba0be7bf734b919927d49bd5e74b57553dfc6eee67371
- **AsyncRAT**: caf9e2eac1bac6c5e09376c0f01fed66eea96acc000e564c907e8a1fbd594426

Both AsyncRAT and VenomRAT are open-source remote access tools developed in C# and built on the .NET Framework (v4.0.30319). A preliminary analysis based on CAPA results revealed several shared characteristics between the two. For example, both RATs use standard libraries like **System.IO**, **System.Security.Cryptography**, and **System.Net** for file handling, encryption, and networking. They also have common cryptographic components such as **HMACSHA256, AES,** and **SHA256Managed**, indicating similar encryption routines. Indeed, upon closer code examination, we found that their encryption classes were identical, with only one minor difference: AsyncRAT uses a 32-byte binary salt, while VenomRAT uses a 16-byte salt derived from the string "VenomRATByVenom." Additionally, both RATs share similarities in configuration handling, mutex creation, and parts of their anti-analysis class.

However, the CAPA analysis also highlighted distinct differences between the two. Certain features present in one RAT were notably absent in the other. To verify, we manually reviewed code in both samples and described the differences below.

Keylogging and System Hooking

In the samples we analyzed the keylogger was present only in VenomRAT. However, the open-source version of AsyncRAT has a keylogger plugin. We therefore decided to investigate whether the VenomRAT keylogger implementation is the same as AsyncRAT's implementation. Our findings suggest that the keylogging functionality is different. We summarized a comparative analysis of their keylogging implementations in the table below. Additionally, the VenomRAT keylogger configuration file **DataLogs.conf** and log files are saved in the user's **%AppData%\MyData** folder.

Feature	VenomRAT	AsyncRAT
Low-level keyboard hook (WH_KEYBOARD_LL)	✔	✔
Keystroke Processing	✔	✔
Window/Process Tracking	Tracks both process and window title	Tracks window title only
Clipboard Logging	✘	✔
Log Transmission	Periodic log sending to C2	Continuous log sending to C2
Filtering Mechanism	✔	✘
Error Handling	Silent failures with basic try-catch	Sends detailed error reports to C2
Additional Features	Focused on keystrokes	Handles both keystrokes and clipboard
Thread Management	✘	✔

Anti-Analysis

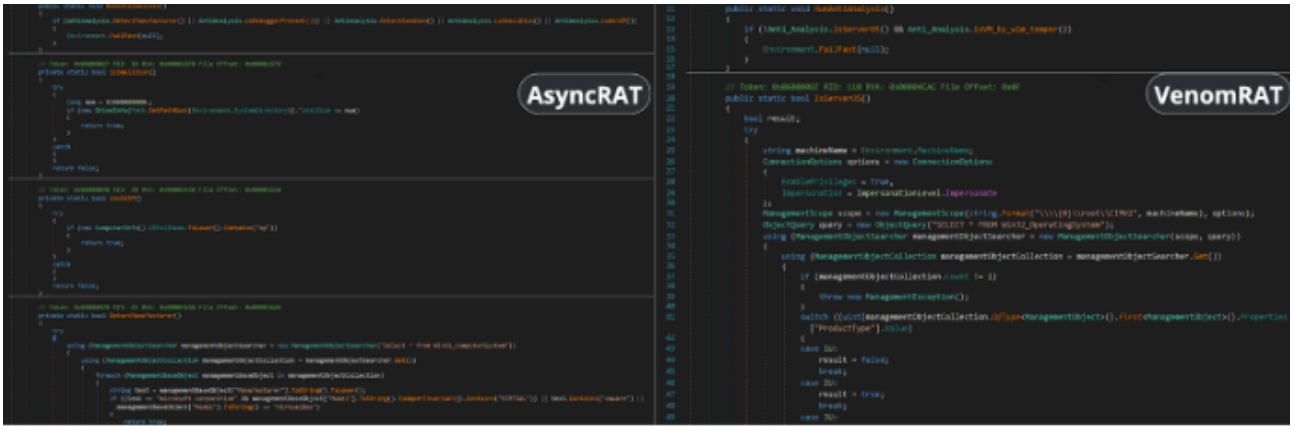
Both AsyncRAT and Venom RAT have similar implementations of the anti-analysis classes. However, we can see notable differences. AsyncRAT focuses on a broad spectrum of detection techniques, including:

- **Virtual Machine Detection**: It checks for known system manufacturer names such as VMware,VirtualBox, or Hyper-V.
- **Sandbox Detection**: It looks for sandbox-related DLLs, such as **SbieDll.dll** from Sandboxie.
- **Debugger Detection**: AsyncRAT uses **CheckRemoteDebuggerPresent** to detect if it's being monitored by a debugger.
- **Disk Size Check**: It avoids execution on machines with less than 60GB disk size.

On the other hand, VenomRAT uses a more targeted approach. The virtual machine detection method in **VenomRAT** relies on querying system memory through **WMI** (Windows Management Instrumentation) to query system memory via **Win32_CacheMemory**. The method relies on counting cache memory entries, and if the number is less than 2 cache memories, it assumes the system is a virtual machine (VM). However, modern VMs are more sophisticated, and simply relying on counting cache memories may not be effective.

The other difference is, instead of targeting debuggers or sandboxes, VenomRAT attempts to avoid running on server operating systems by querying the **Win32_OperatingSystem WMI** class and checking the **ProductType**, which differentiates between desktop and server environments. We summarized class differences in the table below.

Feature	AsyncRAT AntiAnalysis Class	Venom RAT Anti_Analysis Class
VM Detection	✔	✔
Sandbox Detection	✔	✘
Debugger Detection	✔	✘
Operating System Detection	✔	✔
Process Discovery	✘	✔



Hardware Interaction

VenomRAT has hardware interaction capabilities, allowing it to gather detailed system information through **WMI queries** with **ManagementObjectSearcher** objects. These features are encapsulated in the **CGRInfo** class, which enables the collection of CPU, RAM, GPU, and software data:

- **GetCPUName()**: Retrieves the CPU name and the number of cores
- **GetRAM()**: Fetches the total installed physical memory (RAM)
- **GetGPU()**: Obtains the GPU name and driver version
- **GetInstalledApplications()**: Scans the Windows Registry to compile a list of installed applications
- **GetUserProcessList()**: Collects information on all running processes with visible windows

The collected data is sent back to the command-and-control (C2) server. This class is absent in both the version of AsyncRAT we analyzed and the open-source version.

DcRAT joined the party with AntiProcess and Camera classes

VenomRAT includes two notable classes absent in AsyncRAT: the AntiProcess and Camera classes.

The AntiProcess class is an anti-monitoring and anti-detection component of VenomRAT. Malware uses the Windows API function **CreateToolhelp32Snapshot** to get a snapshot of all running processes and search for specific processes. We categorized the processes the malware is looking for below.

System Monitoring Tools that can prevent users from identifying or stopping VenomRAT.

- Taskmgr.exe
- ProcessHacker.exe
- procexp.exe

Security & Antivirus Processes: Terminating them reduces the risk of VenomRAT being detected or removed by security software.

- MSASCui.exe
- MsMpEng.exe
- MpUXSrv.exe
- MpCmdRun.exe
- NisSrv.exe

System Configuration Utilities: By targeting these, VenomRAT prevents users from adjusting security settings, inspecting registry changes, or manually removing the malware.

- ConfigSecurityPolicy.exe
- MSConfig.exe
- Regedit.exe
- UserAccountControlSettings.exe
- Taskkill.exe

If a matching process is found, it terminates it by its process ID (PID).

The Camera class is designed to detect webcams on a Windows system by querying the available system devices using COM interfaces. It retrieves a list of devices by category, specifically looking for video input devices. The class uses the **ICreateDevEnum** and **IPropertyBag** interfaces to enumerate and extract the device names.

However, both these classes, although absent in AasyncRAT, are not exclusive to VenomRAT only. Apparently they are exact copycats of yet another open-source RAT, DcRAT.

AMSI and ETW Bypass

This class was found only in the VenomRAT sample and is designed to bypass key Windows security mechanisms through in-memory patching. It specifically disables two critical Windows security features: AMSI (Antimalware Scan Interface) and ETW (Event Tracing for Windows), which are often used by antivirus software and monitoring tools to detect malware.

Key Functions:

- **AMSI Bypass**: The class patches the **AmsiScanBuffer** function within **amsi.dll** to prevent AMSI from scanning for malicious content.
- **ETW Bypass**: The class patches the **EtwEventWrite** function in **ntdll.dll**, which stops **ETW** from logging events related to the malware’s activity.

The patching process is performed in-memory. The class dynamically checks the system's architecture (32-bit or 64-bit) and loads the appropriate DLLs (**amsi.dll** and **ntdll.dll**) to apply the patches based on the platform. The techniques used by VenomRAT closely mirror those found in the SharpSploit project, an open-source tool often used by penetration testers and red teams to test and bypass security features in a controlled environment. SharpSploit contains classes for bypassing both AMSI and ETW using similar in-memory patching methods, which likely served as inspiration for VenomRAT's implementation.

This security bypass functionality makes VenomRAT more capable of evading modern security defenses.

Dynamic API resolution

VenomRAT has yet another class which is absent in AsyncRAT. The **DInvokeCore** class is implemented to dynamically resolve and call Windows API functions at runtime; this method bypasses traditional static imports, making it harder for antivirus and endpoint detection and response (EDR) systems to detect malicious activity.

Instead of statically importing Windows APIs, the class resolves function addresses at runtime (e.g., from **ntdll.dll** or **kernel32.dll**) using methods like **GetLibraryAddress** and **GetExportAddress**. This approach makes it difficult for static analysis tools to flag malicious behavior.

It uses the **NtProtectVirtualMemory** method to alter memory protection settings, allowing execution of code in memory regions that are normally non-executable—an effective method for in-memory execution of malicious payloads.

Implementation of **DInvokeCore** closely mirrors the open-source SharpSploit Generic class from the D/Invoke project by TheWover. The DInvokeCore class from VenomRAT appears to be a simplified version, which lacks some features but has core techniques for dynamic API invocation.

Conclusion

Our analysis was sparked by detection vendors grouping VenomRAT and AsyncRAT under the same label, blurring the lines between the two. While they indeed belong to the QuasarRAT family, they are still different RATs.

AsyncRAT appears to closely match the latest open-source release (v0.5.8). However, the VenomRAT seems to have evolved and added other capabilities, although a lot of them seem to be a copy-paste from another open-source RAT (DcRAT) and the SharpSploit project. Despite this, VenomRAT presents more advanced evasion techniques, making it a more sophisticated threat.

Therefore, it’s important for security vendors to treat them as distinct threats, recognizing that VenomRAT brings more advanced evasion capabilities, even if much of it isn’t truly unique. To help to resolve this confusion, we are sharing an updated VenomRAT YARA rule with the community, helping improve detection and response efforts.

Rapid7 customers

InsightIDR and Managed Detection and Response (MDR) customers have existing detection coverage through Rapid7's expansive library of detection rules. Rapid7 recommends installing the Insight Agent on all applicable

hosts to ensure visibility into suspicious processes and proper detection coverage. The following rule will alert on a wide range of malicious hashes tied to behavior in this blog: Suspicious Process - Malicious Hash On Asset

YARA rule

The VenomRAT YARA rule can be found on the Rapid7 Labs GitHub here.
