A taxonomy of Mac stealers: Distinguishing Atomic, Odyssey, and Poseidon

By susannah.matt@redcanary.com

Published: 2025-10-09 · Archived: 2026-04-06 00:17:20 UTC

At this point, the threat of macOS-based threats has been well-documented. In 2024, Red Canary noted a surge in adversaries targeting macOS devices to obtain sensitive data from browsers, extensions, and other applications, but 2025 has seen a continued fracturing of those families.

The macOS stealer landscape in particular is dominated by a few key players whose similarities can often make them challenging to distinguish. Poseidon Stealer, prominent during 2024 and early 2025, was sold and rebranded as Odyssey Stealer, an evolution that saw it share significant code and features with another prevalent stealer, Atomic Stealer (aka AMOS).

While their core functionalities and targets remain the same, there's been enough subtle changes over the past 10 months, it felt like a good time to dig into the dynamic world of macOS stealers, tracing the lineage and technical nuances of Atomic, Poseidon, Odyssey, highlighting how these threats adapt to defensive measures, circumvent system protections, and why granular analysis is crucial for effective protection.

AppleScript: The universal language of compromise

One of the primary reasons macOS stealers like Atomic, Poseidon, and Odyssey bear such striking resemblances is their shared reliance on AppleScript.

AppleScript, a powerful scripting language built directly into macOS, is designed to automate tasks, control applications, and interact with various system components. For legitimate users, it's a tool for efficiency and customization. For adversaries, the native integration represents a golden opportunity. Because AppleScript is a

built-in part of the operating system, scripts can often bypass conventional security checks that might flag external, less trusted executables. The system is inherently designed to trust its own components, and AppleScript falls into that category.

A cybercriminal writing a stealer in AppleScript is akin to a Windows threat actor coding an entire stealer using PowerShell. Both languages leverage built-in functionalities and benefit from the system's trust, making them harder to detect by traditional signature-based methods and less likely to trigger immediate alerts from endpoint detection and response (EDR) solutions.

> Get detection opportunities for malicious AppleScript activity in Red Canary's Threat Detection Report.

This foundation means that many of these stealers present a very similar operational footprint—a cascade of AppleScript commands designed to scour a system for valuable data—making differentiation a challenge for security analysts presented with raw endpoint telemetry.

A shared heritage: Similarities between Atomic and Poseidon

Atomic and Poseidon have consistently topped the list of popular macOS stealers. What makes the two particularly intriguing is an alleged history of collaboration, code sharing, and, in some cases, outright copying.

While minor elements such as specific file names used during their operation might differ, the core AppleScript logic—the sequence of commands, the methods for data retrieval, and the general structure of the malicious code—remain largely identical. This genetic overlap has presented a headache for security researchers and incident responders. Without direct access to the malware's command-and-control (C2) infrastructure, distinguishing between Atomic and Poseidon based solely on endpoint telemetry can be a difficult task.

The core challenge for defenders has become finding subtle, yet reliable, indicators that can precisely attribute a stealer to a specific family. Such granular attribution is vital for informing threat intelligence, understanding adversary tactics, and ultimately enabling more precise detection and mitigation strategies.

A brief calm: Gatekeeper's intervention and Poseidon's retreat

The narrative of macOS stealers isn't just about their internal evolution; it's also a story of a continuous cat-and-mouse game with Apple's security enhancements. A significant turning point occurred in fall 2024. Prior to this, a vulnerability existed in macOS that allowed the bypass of Apple's Gatekeeper, something adversaries frequently leveraged for their initial deployment of Atomic and Poseidon onto systems.

Gatekeeper is a fundamental macOS security feature designed to ensure that only approved software—specifically, applications that have been signed by a registered developer and notarized by Apple—can execute. Before Apple fixed this, there was a trivial way to bypass Gatekeeper; users could right-click on an executable and run it, circumventing its protective mechanisms.

While simple, the method was so effective and widely abused that malicious DMGs (disk images) distributing Atomic Stealer even included background images with explicit instructions, coaching unsuspecting users to right-click and "Open" the malicious application rather than double-click, which would have triggered Gatekeeper's warning.

> The narrative of macOS stealers isn't just about their internal evolution; it's also a story of a continuous cat-and-mouse game with Apple's security enhancements.

This era of easy bypass, a golden age for macOS stealer deployment, came to an abrupt end when Apple shipped a crucial update to macOS in October 2024 that successfully patched this critical Gatekeeper vulnerability. Simultaneously, a notable shift occurred in the cybercriminal underground: the persona behind Poseidon, known as "Rodrigo4," reportedly decided to exit the game, selling off his stealer and supposedly retiring from the malicious activity.

This confluence of events led to a relative period of silence from macOS stealers lasting from October 2024 to about March 2025. The primary Gatekeeper bypass was closed, significantly hindering traditional deployment methods, and one of the two dominant stealer families was seemingly out of business. It was a brief, welcome, respite that highlighted the impact that both platform-level security improvements and the volatile, often unpredictable, nature of the cybercriminal market can have.

Re-emergence and evolution: The birth of Odyssey and Atomic's catch-up

The period of calm proved to be short-lived. Stealer activity on macOS started back up again in March 2025, signaling a renewed and adapted threat. This resurgence was marked by a shift in deployment methodologies. With the Gatekeeper bypass firmly closed, adversaries were forced to innovate and find new ways to trick users into executing their malware. This led to widespread adoption of paste-and-run or "ClickFix" methodologies relying heavily on user interaction and exploiting human trust rather than technical vulnerabilities for initial access.

Crucially, this period also saw the re-emergence of a familiar threat under a new guise: Odyssey Stealer.

When the developer behind Poseidon Stealer, Rodrigo4, sold the stealer In the fall of 2024, it turned out to be a temporary hiatus, or it could be argued, a tactical rebranding.

> Odyssey, in essence, is a sophisticated variation or updated iteration of the original Poseidon, designed to operate in the post-Gatekeeper bypass era.

This is consistent with Red Canary Intelligence's observations. Poseidon did not appear in Red Canary's data from November 2024 until March 2025. From March 2025 until September 2025, Red Canary associated newer Odyssey stealer instances with Poseidon's profile as Odyssey is an evolution of Poseidon.

This new generation of stealers wasn't merely a re-skinning of old code. Odyssey Stealer brought with it a host of enhanced capabilities designed specifically to evade detection, improve resilience, and ensure persistence on compromised systems. These new features included:

- **Anti-sandboxing mechanisms:** Tools and logic to detect and avoid execution within analysis environments like virtual machines or emulators, thereby frustrating security researchers.
- **Persistence with launch daemons:** Leveraging macOS's launchd system—a core service management framework—to ensure the stealer runs automatically after system reboots and maintains its presence on the machine.
- **Botnet component:** Adding functionality for persistent remote execution and control, indicating a move towards more complex capabilities beyond simple, one-time data exfiltration. This suggests the ability for ongoing surveillance or multi-stage attacks.

The threat landscape is a highly competitive one, even among cybercriminals. Shortly after Odyssey introduced these new, advanced features, Atomic Stealer worked the same features into their tools, underscoring the continuous arms race as well as the quick adoption and integration of effective new techniques within the malicious ecosystem.

The devil in the details: Distinguishing stealers at the endpoint

Given the similarities in AppleScript and the rapid feature adoption that saw both Atomic and Odyssey incorporating similar capabilities, how can defenders accurately distinguish between these evolving threats without relying on external C2 intelligence that may be difficult to obtain?

Minor distinctions appear in the HTTP request headers that these stealers generate:

- **Case sensitivity:** Atomic's `curl` commands often contain `BuildID` (with a capital `B` and `ID`), Odyssey's often shows `buildid` (lowercase `b` and `id`). While trivial, these case differences can be indicative of the stealer family.
- **Header variations:** Atomic's commands often contain a generic user header, whereas Odyssey specifies username. Additional commands, including `cl: 0`, a custom HTTP header likely used to identify the compromised machine and `cn: 0`, another custom HTTP header included in the request to the adversaries' server, have also been spotted; Red Canary has observed them in Atomic Stealer but not others, providing further points of distinction.

Red Canary has also observed Odyssey and Atomic differ slightly when it comes to their choice of file names, URLs, and command-line options for exfiltration using `curl`, seen in the table below.

Shared tactics and evolving anti-analysis techniques

Despite these differences, these macOS stealers share many tactical similarities, indicating a common understanding of valuable targets and effective evasion methods. Both Atomic and Odyssey, for instance, typically target the same software families, often focusing on high-value data sources like browser extensions (which can store cookies, session tokens, and cached credentials), cryptocurrency wallets, and other applications holding sensitive personal and financial information. They also exhibit similar file-name patterns when collecting data from common user directories like desktop and documents, suggesting standardized approaches to data reconnaissance and collection.

However, their evolution also reflects a growing sophistication in anti-analysis techniques, designed to thwart security researchers and automated sandboxes.

Username checks

Some stealers, including Odyssey, might include logic to check for specific usernames such as `maria`,— VirusTotal's macOS sandbox— `jackiemac`, or `root` —Triage, Recorded Future's macOS sandbox name. By detecting these common sandbox usernames, the malware can choose not to execute its full payload or to behave benignly, effectively evading dynamic analysis and revealing its true malicious intent only on genuine victim systems.

Hardware-based sandboxing evasion

A particularly advanced and effective technique involves checking the system's processor architecture. Newer variants of Atomic often have mechanisms in place where it refuses to run on macOS systems with an Intel processor. This is a clever evasion tactic because many sandbox systems available for macOS systems use Intel Core processors. By explicitly targeting ARM architecture execution only (i.e., Apple's silicon Macs), these stealers can bypass a significant portion of automated analysis environments, ensuring the malicious payload is only delivered and executed on genuine consumer systems running newer hardware, where detection might be less mature.

Deployment method evolution: From DMGs to `bash` scripts

The methods of initial access for macOS stealers have undergone a transformation, primarily driven by Apple's continuous improvements to its built-in security features, most notably Gatekeeper.

Over the last several months, these stealers have shifted their focus to "paste and run" or "ClickFix" methodologies; this involves tricking users into executing commands directly in the terminal, often through various deceptive means:

Fake Homebrew websites, repositories on GitHub

Threat actors create fraudulent websites that mimic legitimate software repositories or package managers, such as Homebrew. These sites then prompt users to copy and paste seemingly benign installation commands into macOS

Terminal, which secretly downloads and executes the stealer.

Direct `bash` scripts

Adversaries don't even have to deploy executable binaries; they can lure users to execute a `bash` script directly, sometimes via a simplified `curl` in `bash` command. By executing a command directly in Terminal, it bypasses Gatekeeper; the sequence downloads a script from a remote server and immediately pipes it into the bash interpreter, executing it without ever saving it to disk or triggering Gatekeeper.

This highlights the importance of robust user education regarding Terminal commands and the need for advanced endpoint detection solutions that can scrutinize and analyze terminal activity for suspicious patterns, not just file execution.

Remediation: Consistency amidst differentiation

While distinguishing between stealer families with detail is crucial for threat intelligence, tracking, and proactive defense, the immediate remediation steps for compromised macOS systems largely remains consistent regardless of the specific stealer identified.

A successful stealer compromise, by nature, indicates a deep compromise of user data and system integrity. That means effective remediation typically necessitates the following.

Operating system reset/re-imaging

This is often the most secure approach to ensure all malicious components, including any persistence mechanisms like launch daemons or hidden files, are completely removed from the system.

Credential resets

All accounts accessed or potentially exfiltrated by the stealer must have their passwords reset. This includes browser logins, iCloud accounts, email accounts, financial services, and any other sensitive services the user accessed from the compromised machine. This may also include re-issuing certificates or keys and revoking any existing logon sessions as these stealers can exfiltrate many kinds of files and browser cookies. Multi-factor authentication should also be enabled or reviewed for all critical accounts.

Security posture review

A thorough post-incident analysis is essential. This involves identifying the initial vector of compromise (e.g., social engineering, drive-by download) to understand how the stealer gained access. This information is crucial for implementing preventative measures and patching security gaps to prevent similar incidents in the future. Consider educating users on Transparency, Consent, and Control (TCC) controls in macOS and presenting scenarios when users may not want to bypass TCC to preserve their own security and privacy.

Staying ahead in the macOS security race

The evolution of macOS stealers like Atomic, Poseidon, and Odyssey paints a clear picture of an increasingly sophisticated, adaptable, and persistent threat landscape.

While immediate remediation actions for a compromised system often remain consistent, regardless of the specific stealer involved, the ability to precisely differentiate between stealer families at the endpoint can help defenders.

Being able to perform a more granular analysis—particularly through the examination of subtle C2 communication parameters and evolving deployment methods—can help enable the more accurate threat intelligence, attribution of attacks to specific groups, and facilitate the creation of proactive, adaptive security strategies.
