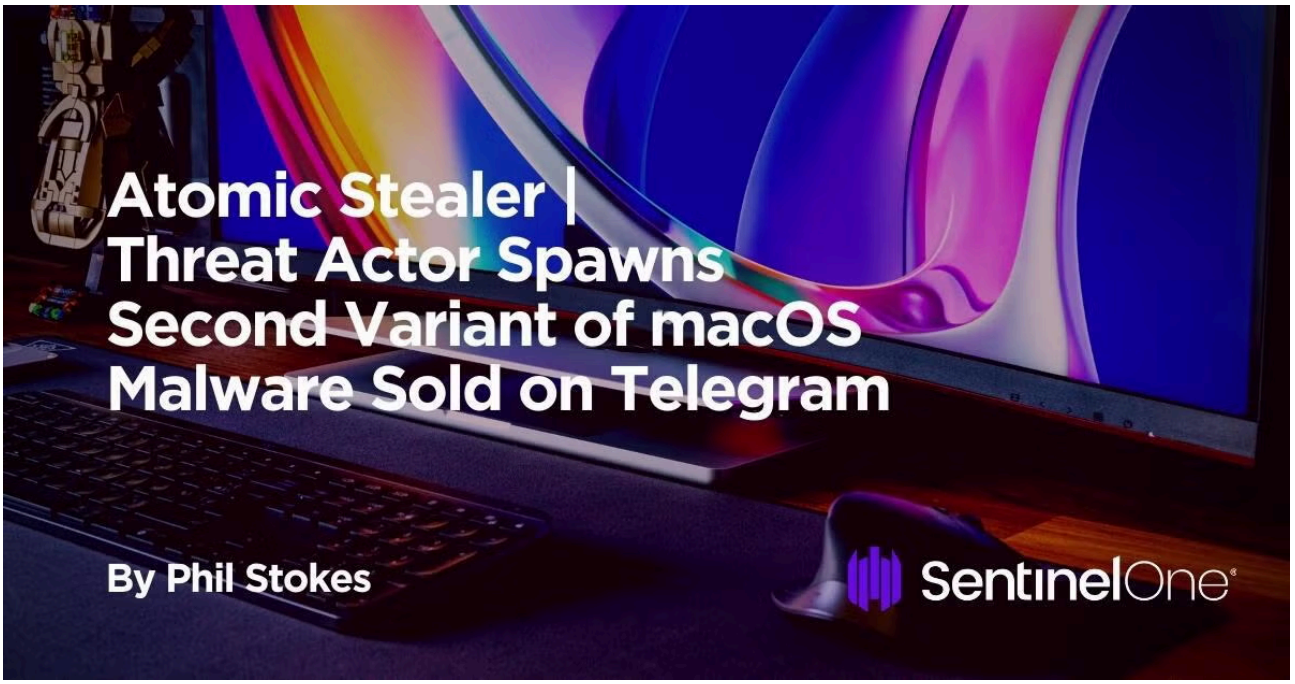
Atomic Stealer | Threat Actor Spawns Second Variant of macOS Malware Sold on Telegram

By Phil Stokes

Published: 2023-05-03 · Archived: 2026-04-02 10:39:28 UTC

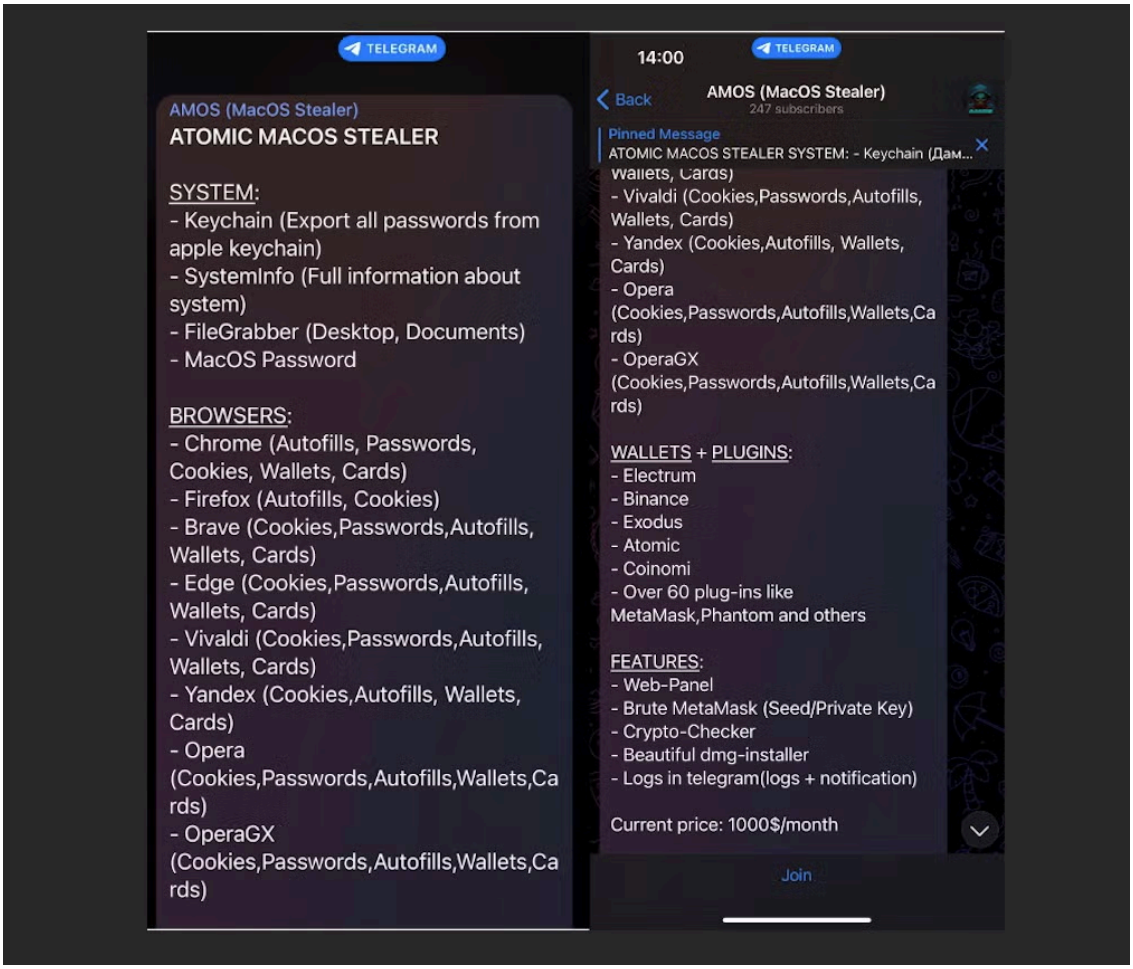
Recent weeks have seen a number of macOS-specific infostealers appear for sale in crimeware forums, including Pureland, MacStealer and Amos Atomic Stealer. Of these, Atomic Stealer has offered by far the most complete package, promising cybercriminals a full-featured if not particularly sophisticated infostealer. Atomic can grab account passwords, browser data, session cookies, and crypto wallets, and in the version being advertised on Telegram, threat actors can manage their campaigns through a web interface rented out from the developer for $1000 per month.

The threat actor, however, has been busy looking for other ways to target macOS users with a different version of Atomic Stealer. In this post, we take a closer look at how Atomic Stealer works and describe a previously unreported second variant. We also provide a comprehensive list of indicators to aid threat hunters and security teams defending macOS endpoints.



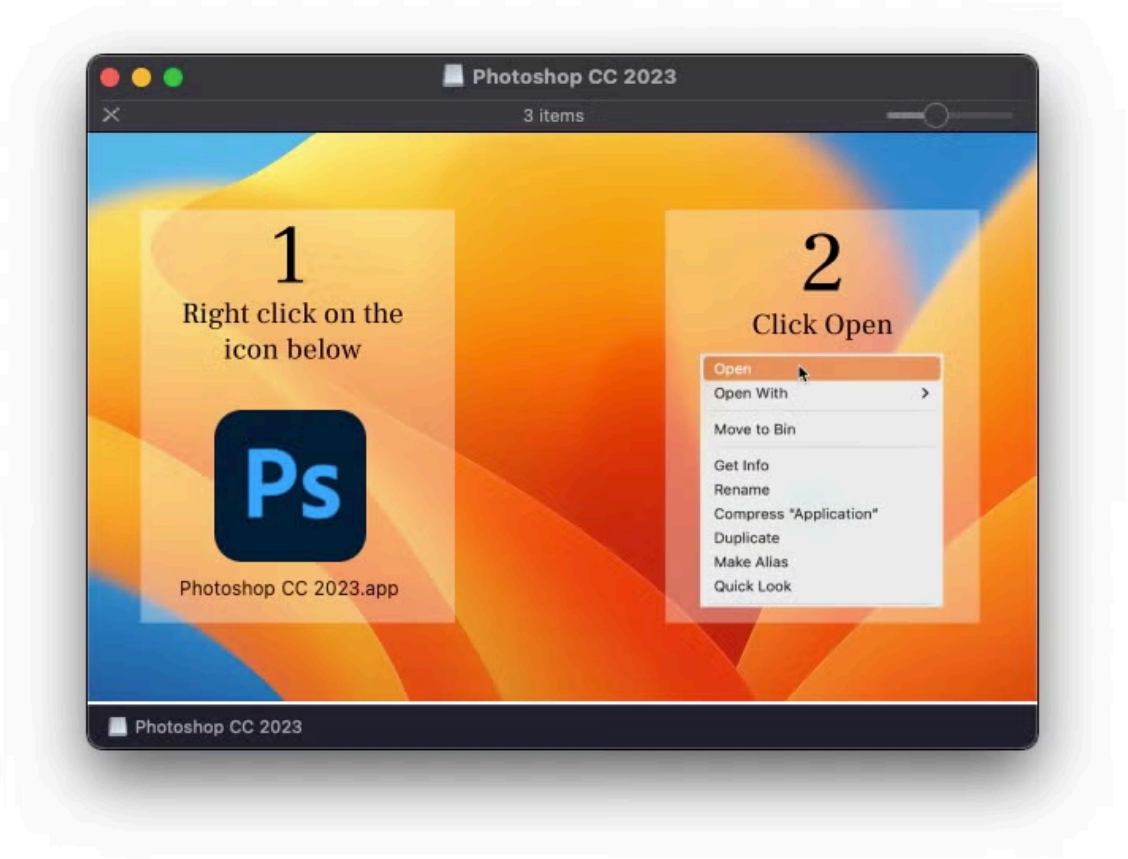
How is Atomic Stealer Distributed?

Cybercriminals are currently being offered “Amos Atomic MacOS Stealer” via a dedicated Telegram channel. In the channel, which was opened on April 9th, the author offers to rent access to a web panel and provide a disk-image based installer for $1000/month.



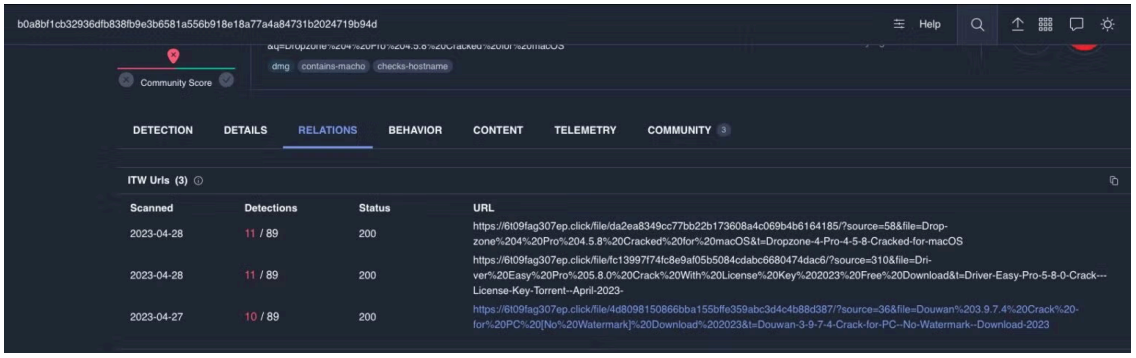
Atomic Stealer as advertised on Telegram

Payload distribution is left up to the crimeware actor renting the package, so methods vary, but so far observed samples have been seen masquerading as installers for legitimate applications like the Tor Browser or pretending to offer users cracked versions of popular software including Photoshop CC, Notion, Microsoft Office and others.



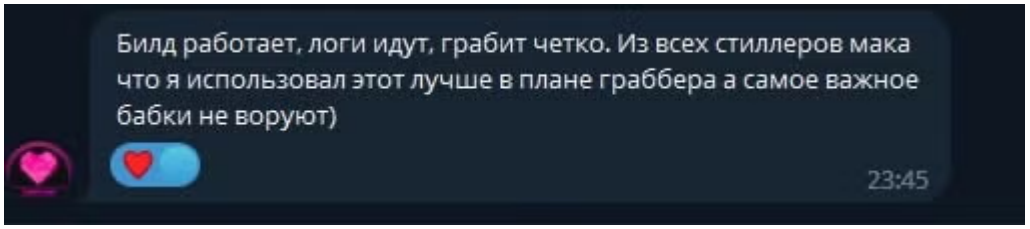
Atomic MacStealer masquerades as legitimate applications

Malvertising via Google Ads has also been noted privately among researchers as a distribution vector for Atomic Stealer.



Some Atomic Stealer ITW URLs (Source: VirusTotal)

The Atomic Stealer channel currently has over 300 subscribers, with some posts – possibly planted – appearing to endorse the efficacy of the malware.



A Telegram message seems to endorse Atomic MacStealer

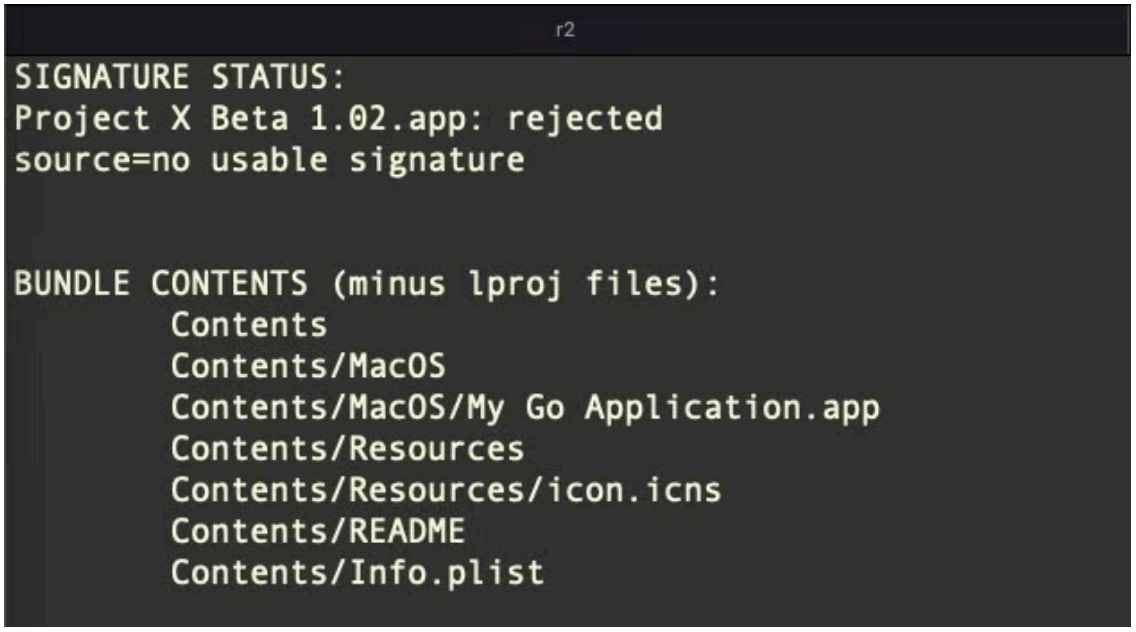
A Google translation of the Russian text reads "*The build works, the logs go, it robs clearly. Of all the poppy stealers that I used, this one is better in terms of a grabber, and most importantly, grandmas do not steal)*".

These fake applications are made with a fork of Appify, a legitimate script that can be found on Github for making a barebones macOS application. All Atomic infostealers currently contain the same Go-based main executable that weighs in at around 51.5MB as a Universal binary targeting both Intel and arm64 architectures.

```
[0x012c0bcd]> .(meta)
file     My Go Application.app
type     Executable file
class    MACH064
arch     x86
compiler clang
lang     c
51.6M
[Sections]

nth paddr            size vaddr           vsize perm md5, entropy                                type             name
-----------------------------------------------------------------------------------------------------------------------
0   0x00001000   0x23f579 0x01001000   0x23f579 -r-x 38d1acfebb21208337b705baa728e2c8 6.16609423 REGULAR          0.__TEXT.__text
1   0x00240580      0x258 0x01240580      0x258 -r-x 40520b2d17f7b91840cdbefb32c320b2 4.80426892 SYMBOL_STUBS     1.__TEXT.__symbol_stub1
2   0x002407e0    0xe8756 0x012407e0    0xe8756 -r-x 5b1610a1e74ac2c2c05a68bdbd24cad8 5.01490112 REGULAR          2.__TEXT.__rodata
3   0x00328f40     0x15f4 0x01328f40     0x15f4 -r-x 2b459898e4e45b520bd4fed8b26d1a3a 4.40716826 REGULAR          3.__TEXT.__typelink
4   0x0032a540      0x890 0x0132a540      0x890 -r-x dd2e4c2e2908d0d4f820e2d291b37d23 4.90535331 REGULAR          4.__TEXT.__itablink
5   0x0032add0        0x0 0x0132add0        0x0 -r-x                                             REGULAR          5.__TEXT.__gosymtab
6   0x0032ade0   0x143430 0x0132ade0   0x143430 -r-x b4adcf7d29e1435eac776b18f62bbfe4 6.05306129 REGULAR          6.__TEXT.__gopclntab
7   0x0046f000      0x110 0x0146f000      0x110 -rw- ce26d1a02275c1879836b13895cebc3e 3.79792704 REGULAR          7.__DATA.__go_buildinfo
8   0x0046f110      0x320 0x0146f110      0x320 -rw- 3a2e168a42fa79c56976c388568c8c45 1.61890425 NONLAZY_POINTERS 8.__DATA.__nl_symbol_ptr
9   0x0046f440  0x1352d10 0x0146f440  0x1352d10 -rw-                                             REGULAR          9.__DATA.__noptrdata
10  0x017c2160     0xadf0 0x027c2160     0xadf0 -rw- 0da78689d2df0499474020c91edd29aa 1.88234116 REGULAR          10.__DATA.__data
11  0x00000000        0x0 0x027ccf60    0x30e80 -rw-                                             ZEROFILL         11.__DATA.__bss
12  0x00000000        0x0 0x027fdde0     0x6450 -rw-                                             ZEROFILL         12.__DATA.__noptrbss
13  0x017cd000      0x129 0x02805000      0x129 -r-- 9ee38aeba19f4d46fcd9eda4661325d2 0.00000000 REGULAR          13.__LINKEDIT.__zdebug_abbrev
14  0x017cd129    0x61d61 0x02805129    0x61d61 -r-- 35eae473441af51527ec26d6a0adf2da 7.98816943 REGULAR          14.__LINKEDIT.__zdebug_line
15  0x0182ee8a    0x13a5b 0x02866e8a    0x13a5b ---- 56190729b49edc92a8f0bea8255d4c70 7.94776605 REGULAR          15.__PAGEZERO.__zdebug_frame
16  0x018428e5       0x2a 0x0287a8e5       0x2a ---- 212de006081fe321532470fd28faece1 5.29707933 REGULAR          16.__PAGEZERO.__debug_gdb_scri
17  0x0184290f    0xa8c37 0x0287a90f    0xa8c37 ---- 1c71d1e59bd4480083e9dff171d416c0 7.99801346 REGULAR          17.__PAGEZERO.__zdebug_info
18  0x018eb546    0x772a3 0x02923546    0x772a3 ---- c38fb3b7ada44546963b58c3ede64394 7.99139860 REGULAR          18.__PAGEZERO.__zdebug_loc
19  0x019627e9    0x1bfae 0x0299a7e9    0x1bfae ---- 22e6a840b3367fab558a14e9d6b484c0 7.90101923 REGULAR          19.__PAGEZERO.__zdebug_ranges

0x01000020  cmd        0 0x19 LC_SEGMENT_64
0x01000068  cmd        1 0x19 LC_SEGMENT_64
0x010002e0  cmd        2 0x19 LC_SEGMENT_64
0x01000508  cmd        3 0x19 LC_SEGMENT_64
0x01000780  cmd        4 0x19 LC_SEGMENT_64
0x010007c8  cmd        5 0x32 LC_BUILD_VERSION
0x010007e0  cmd        6 0x5 LC_UNIXTHREAD
0x01000898  cmd        7 0x2 LC_SYMTAB
0x010008b0  cmd        8 0xb LC_DYSYMTAB
0x01000900  cmd        9 0xe LC_LOAD_DYLINKER
0x01000920  cmd       10 0xc LC_LOAD_DYLIB
0x01000958  cmd       11 0xc LC_LOAD_DYLIB
0x01000990  cmd       12 0xc LC_LOAD_DYLIB
0x010009f8  cmd       13 0xc LC_LOAD_DYLIB
[Linked libraries]
/usr/lib/libSystem.B.dylib
/usr/lib/libresolv.9.dylib
/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation
/System/Library/Frameworks/Security.framework/Versions/A/Security
```

Anatomy of the Atomic Stealer binary (variant A)

Despite the heft, no attempt has been made to deliver a working copy of the spoofed apps. Aside from the Appify README, the Bundle contains nothing more than the Go infostealing binary, an icon file and a Info.plist.



Anatomy of an Atomic Stealer application bundle

The application bundles currently being distributed are all built with the default Appify bundle identifier, `Appify by Machine Box.My Go Application`, potentially a deliberate ploy by the author in the hope that detections might be considered false positives.

Execution Behavior of Variant A

Atomic does not attempt to gain persistence, an increasing trend since Apple added login item notifications in macOS Ventura, relying instead on a one-hit smash and grab methodology.

Atomic Stealer uses a crude but effective means of extracting the user's login password via AppleScript spoofing.

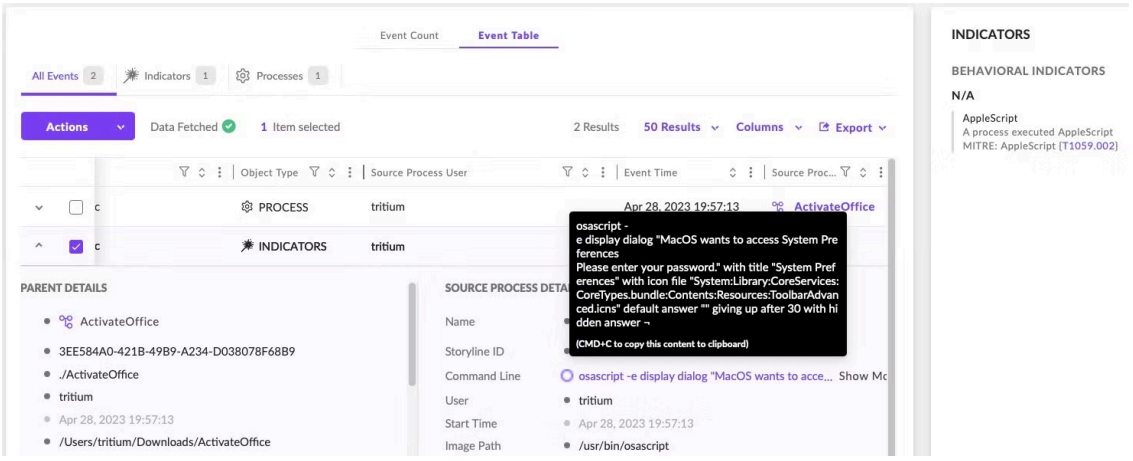
This involves creating a dialog box with osascript and passing the `hidden answer` parameter to the `display dialog` command. These dialog boxes contain an ordinary text field, but the parameter displays the user's typed characters as dots in the text field similar to a genuine authentication dialog. However, the password remains captured in plain text and can be seen in the system logs as such – a good reason why legitimate software developers should never use this insecure method to actually obtain user credentials.

```
display dialog "MacOS wants to access System Preferences

You entered invalid password.

Please enter your password." with title "System Preferences" with icon file "System:Library:CoreServ
```

The dialog box message contains grammatical and syntactic errors, suggesting the developer's first language is not English. The dialog box is generated using an infinite loop: Clicking the "Cancel" button simply pops the dialog box again. If the "OK" button is clicked, the malware checks to see that the user entered a valid password via `/usr/bin/dscl` utility and the `-authonly` option.

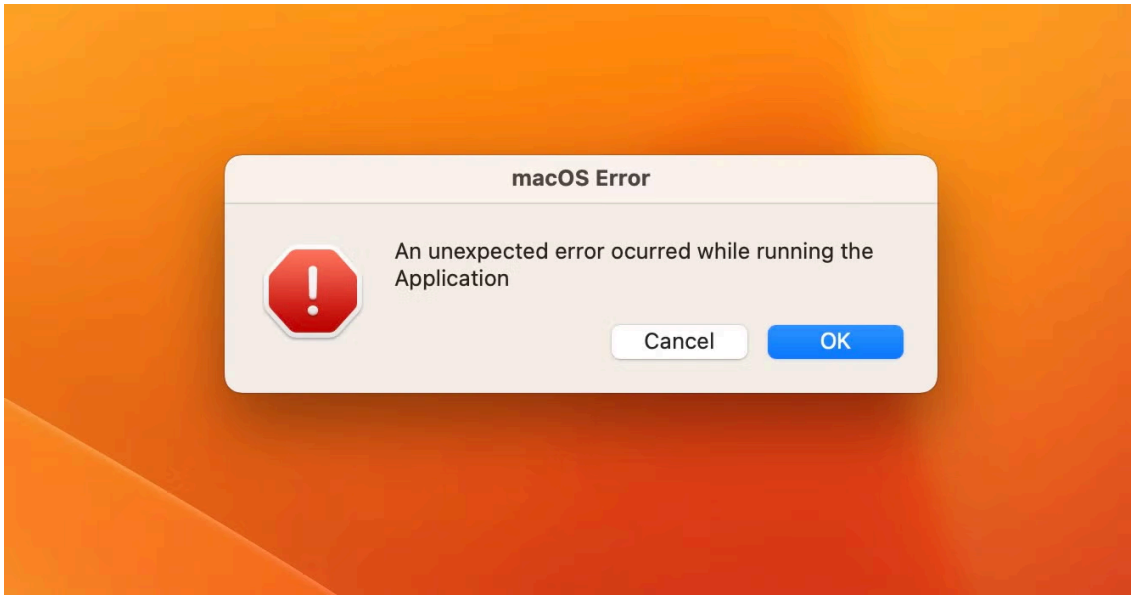
```
  authonly
    Usage: authonly [user [password]]

    Used to verify the password of a named user, or of "root" if no user
is specified.  If a password is supplied, then that password is used for
    authentication, otherwise the command prompts for a password.
```

The dialog box repeatedly pops until the correct password is supplied. All of this occurs via the command line utility `osascript`, so it is easily visible to defenders monitoring command line activity.



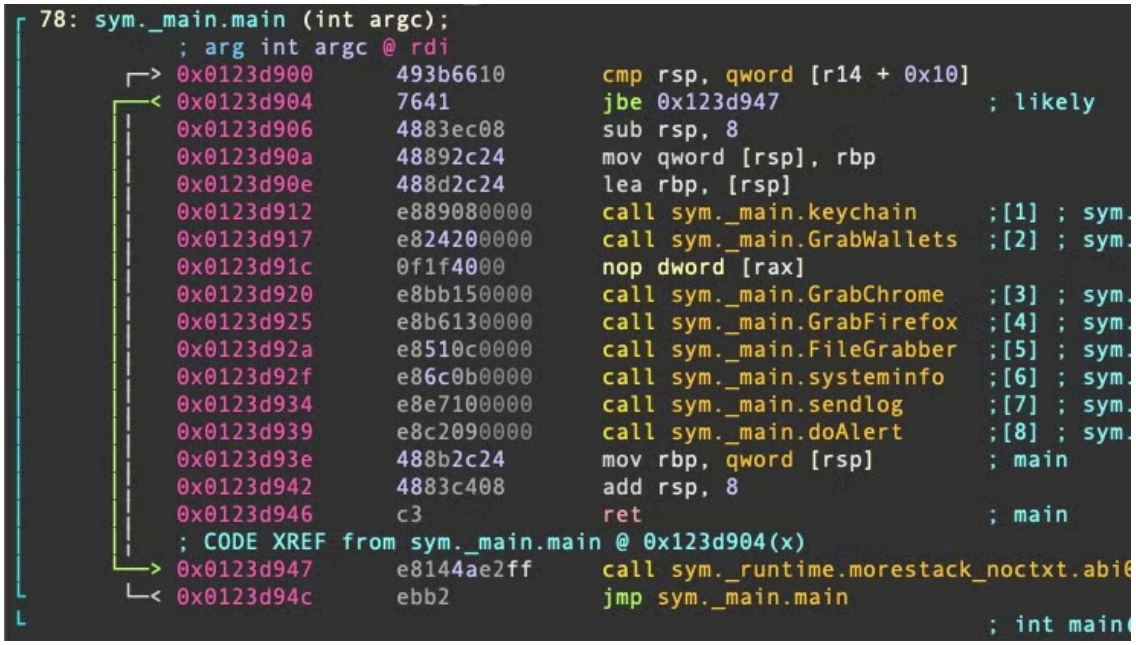
SentinelOne console reveals Atomic Stealer command line activity

Amos Atomic is hardcoded to throw the user an error message after it has stolen the user's password and gone about its business of stealing various credentials. Here and elsewhere, the malware author's lack of familiarity with English and AppleScript provide clues that should raise suspicions: namely, the misspelling of "occurred" and the fact that a genuine error message shouldn't contain a 'Cancel' button.



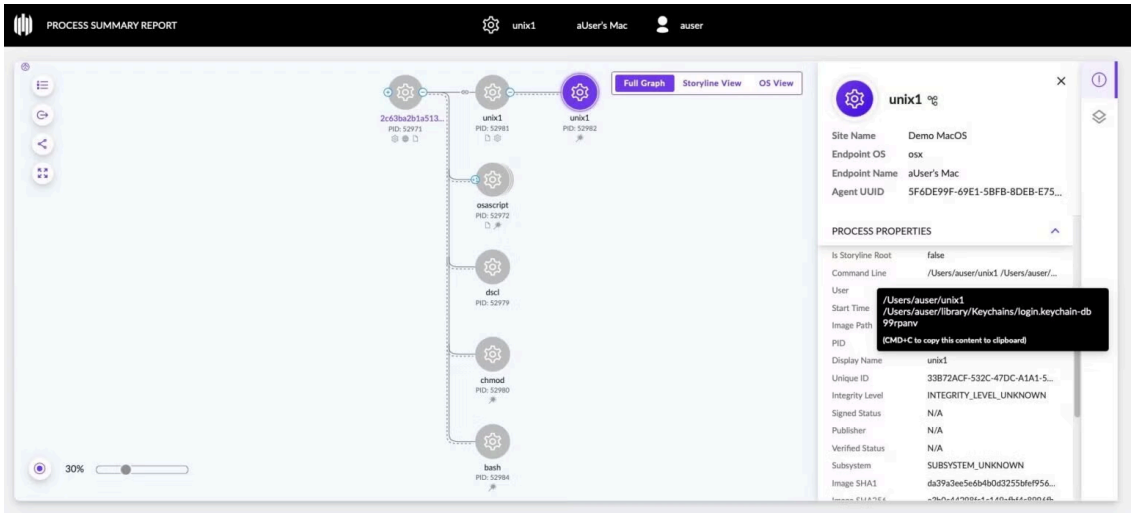
Amos Atomic throws an error message and quits after successfully stealing user data

Written in Go, the disassembled source code reveals a comprehensive suite of functions to achieve the infostealers primary aim: financially-motivated cybercrime.



Infostealing functions in Amos Atomic

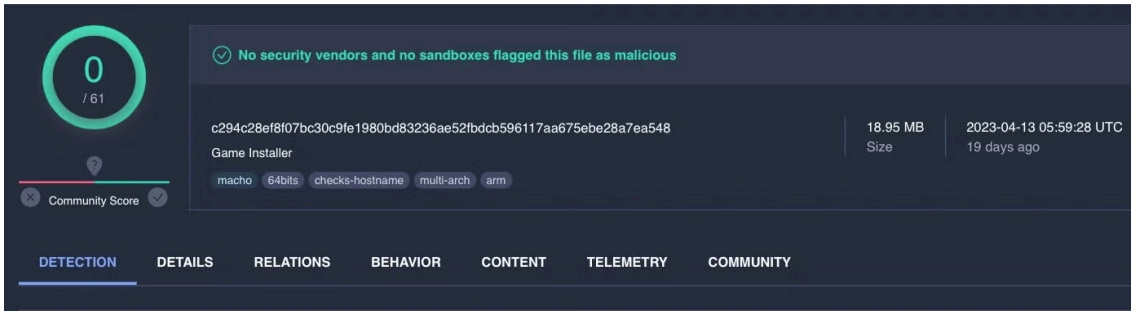
The malware contains logic to steal the user's keychain and crypto wallet contents, including those for Atomic, Binance, Electrum and Exodus. A process called 'unix1' is spawned in memory to obtain the keychain. Atomic stealer also targets both Chrome and Firefox browsers and has an extensive hardcoded list of crypto-related browser extensions to attack. A detailed walk through of the functions above has been previously described here.



Atomic Stealer execution chain

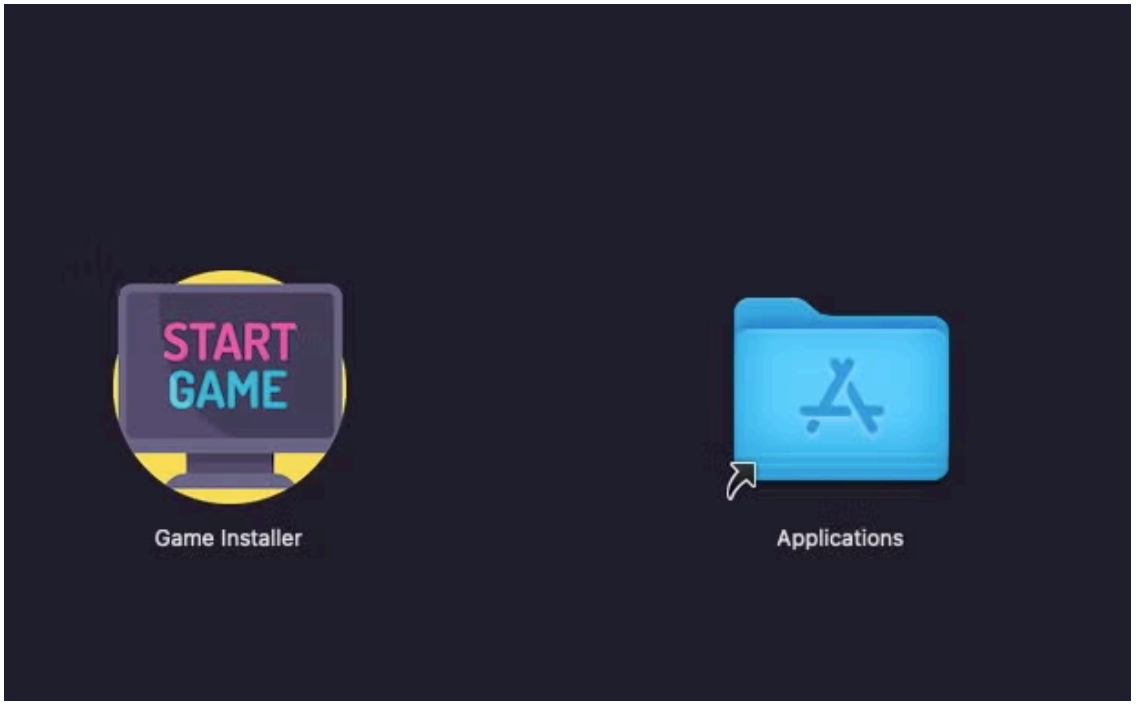
Atomic Stealer Variant B

Pivoting off the IP address `37.220.87.16` seen in some Atomic Stealer samples leads to another variant of the stealer, `c70fdf4362eb56032793ab08e6aeb892f1bd4a9b`, currently undetected on VirusTotal, masquerading as a Game Installer.



A previously undiscovered variant of Atomic Stealer

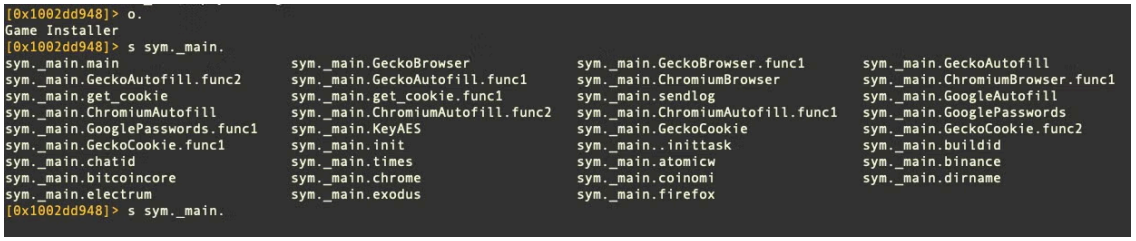
This version is not distributed in an application bundle, but rather as a raw Go binary. The unsigned “Game Installer” Mach-O was uploaded to VirusTotal on April 13th and is contained in a disk image called “ALMV_launcher”. The DMG mounts with the name “Game Installer” and contains a binary of the same name, displaying an icon showing the text “Start Game”.



Background image of the ALMV_launcher.dmg

As the universal binary is unsigned, it will need to be manipulated by the user on both Intel and arm64 architectures in order to run.

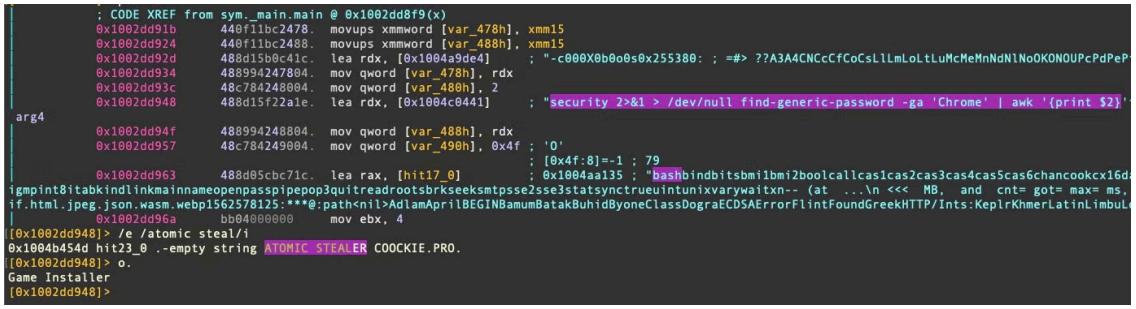
Variant B’s list of Go main functions differs from the version being packaged and sold on Telegram and shows a larger number of functions focusing on Firefox and Chromium browsers. Variant B also targets Coinomi wallets.



Atomic Stealer variant B primary functions

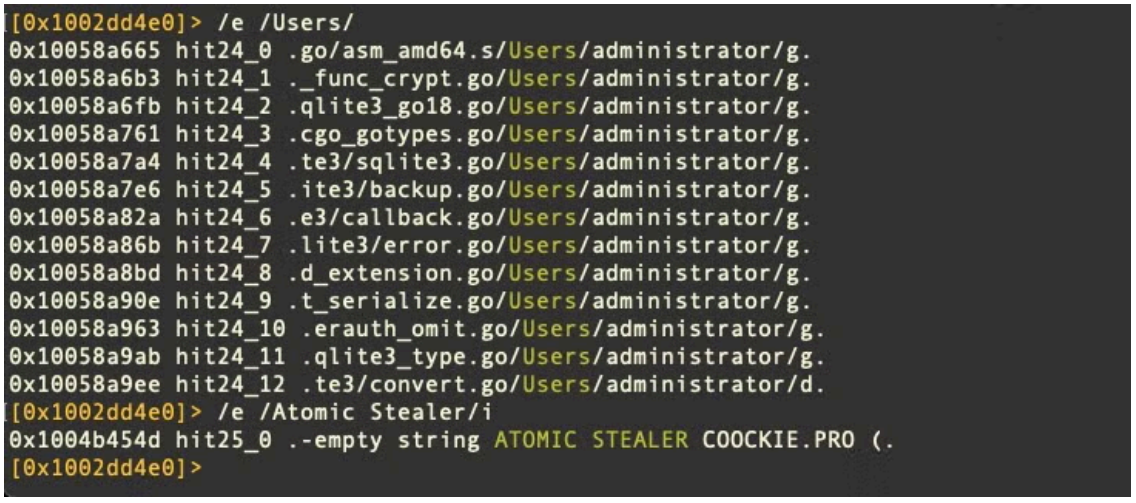
Both variant A and B utilize the `/usr/bin/security` utility to find Chrome passwords.

```
security 2>&1 > /dev/null find-generic-password -ga 'Chrome' | awk '{print $2}
```



Atomic Stealer B calls the */usr/bin/security* utility to find Chrome passwords

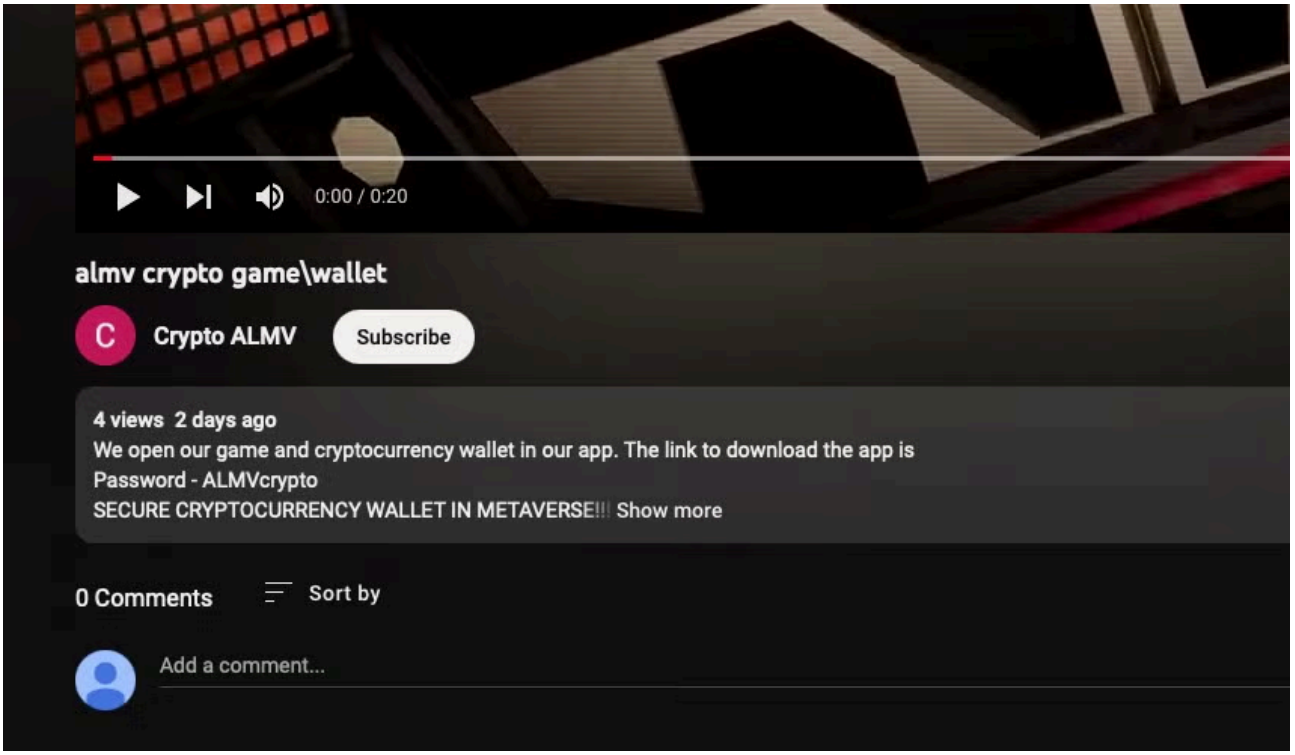
In Variant B, the user name “administrator” appears from the development machine; this differs from variant A, which included the username “iluhaboltov”. The string “ATOMIC STEALER COOCKIE” is also found in variant B but not A.



The “ATOMIC STEALER” string is hardcoded into the malware

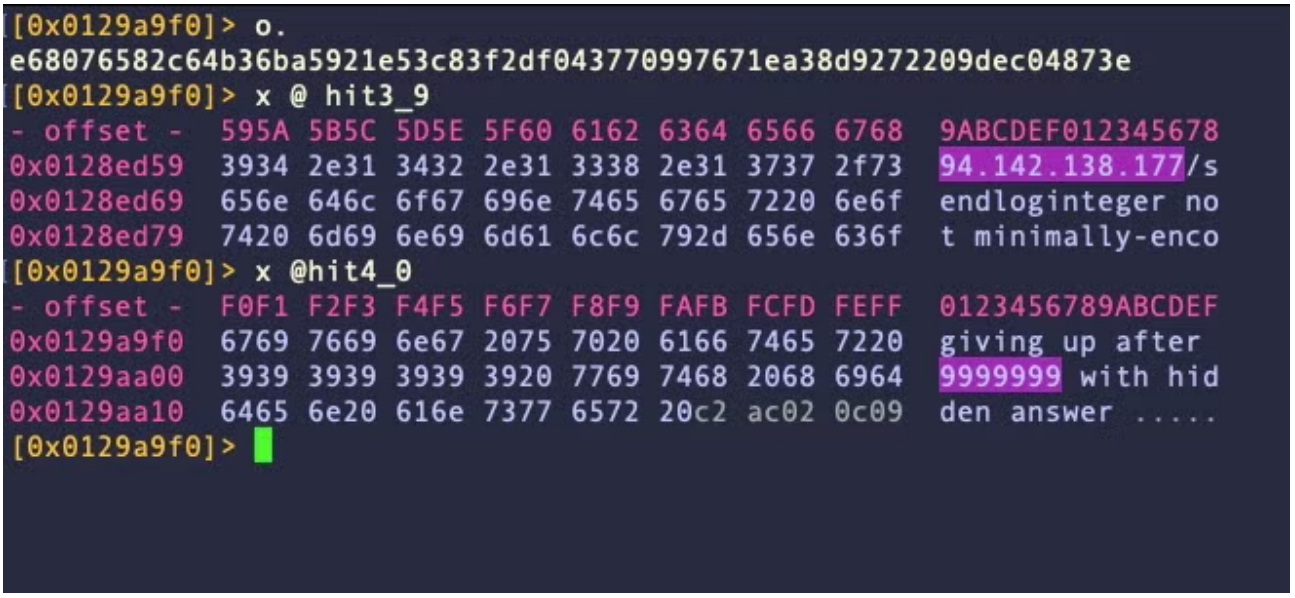
Unlike the package offered in the Atomic Telegram channel, this version of Atomic stealer is more selective in the information it tries to steal and seems to be aimed specifically at games and users of cryptocurrency.

An associated Youtube channel by user @Crypto-ALMV was created on April 29th, apparently advertising a product that offers cryptowallet access within a game. The channel, user, and video appear to be in the early stages of development and may indicate a campaign that is yet to be launched.

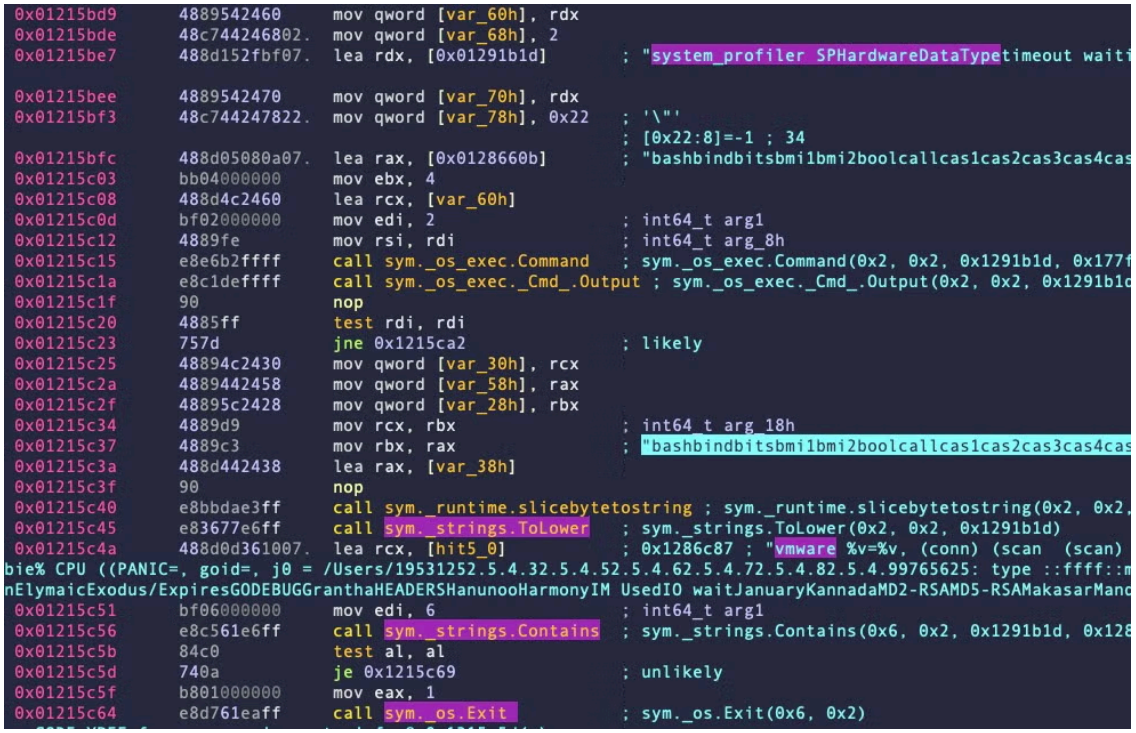


Atomic Stealer Variant C

A third variant appeared in mid-May that reduces the file size of these Go bins markedly to between 4.45 and 9.13MB. Variant C changes the C2 to `http[:]//94[.]142[.]138[.]177/` and uses a slightly different form of the "display dialog" command, adding a timeout of 9999999, although its uncertain that this timeout would actually work, as Apple events ordinarily have a max timeout of around 150 seconds.



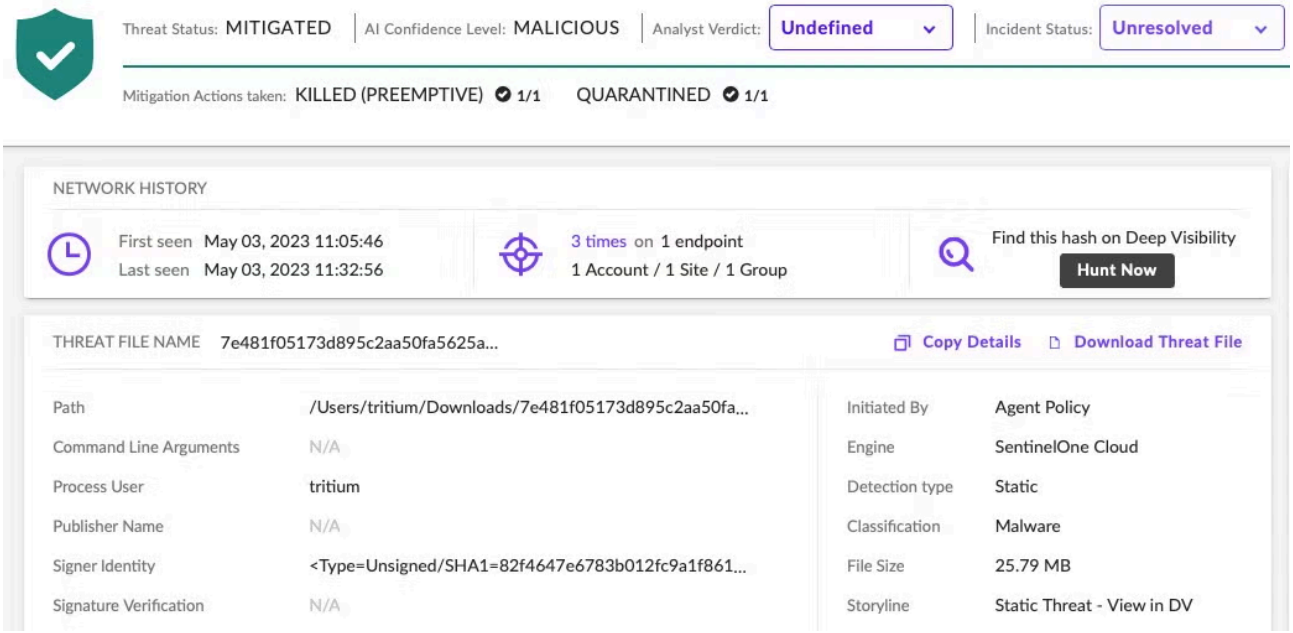
Variant C is most similar to Variant A, but lighter and with some rudimentary attempt at anti-analysis in the `sym._main.systeminfo` function, which does not appear in earlier variants. This function queries the built-in `system_profiler` tool's output for `SPHardwareDataType`, converts the output to lowercase, then searches it for the substring "vmware". If the substring is found, the malware then exits.



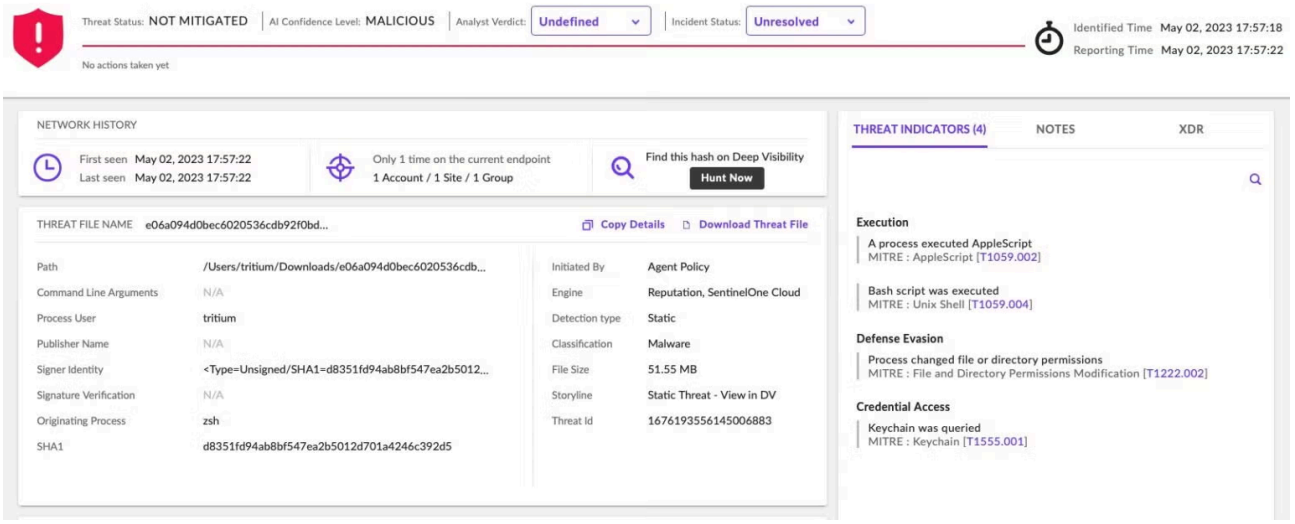
Atomic Stealer Variant C attempts to test if the malware is running in a Virtual Machine.

How to Protect Against Atomic Stealer

SentinelOne customers are protected against all known versions of Atomic Stealer. When the agent is set to 'Protect' mode, Atomic Stealer is prevented from executing.



In Detect Only mode, the malware's execution causes an alert and behavioral and threat indicators are available in the console.



Threat hunters and security teams not protected by SentinelOne are encouraged to review the list of Indicators of Compromise provided at the end of this post.

Conclusion

Infostealers targeted at Mac users have become increasingly viable for threat actors now that Macs have reached widespread use in organizations, both for work and personal use. As many Mac devices lack good external security tools that can provide both visibility and protection, there is plenty of opportunity for threat actors to develop and market tools to aid cybercriminals.

Atomic Stealer's advertised price suggests there is money to be made by "selling shovels" as cybercrime actors rush for the 'Gold' of data that can be harvested by tricking users into running untrustworthy software. However, the existence of a second variant that appears to be aimed at infecting users first-hand suggests the threat actor isn't averse to a bit of gold digging, too.

Indicators of Compromise

Communications

94[.]142[.]138[.]177/sendlog
amos-malware[.]ru/sendlog
37[.]220.87[.]16:5000/sendlog

October 2023 Update

078dd6122694cbc6e637a11fec77d6cab94bac3b
07fb38e48529490da73dcb9a0812bd3bb3337189
1e1981c43d6524f3d51409f884cfe2155b9c5252
23f7032a7f1dae759a56cfb6a89fc90a65fd7493
2dfc73283a2f9aad0945af8578990f5b4076b649
2ff09bc869c0fa6c5ec9538ffe654053f3dfb704
30d1f086986cf6376d83295a50038b763a280890
37d51ca4e21f228320ce88a0312ff0118c295b9f

37d51ca4e21f228320ce88a0312ff0118c295b9f
466e588f1145b1cb0a7445474a9216332343d4b3
4811fea467a04a2f64fc22d7ca24051202815206
4811fea467a04a2f64fc22d7ca24051202815206
509234ccfa77ada7db066fc66eb15d489a526904
52f045a46b7be149d033d4021ccd803cd05c284f
5428d609d3f9fe66a920f7c43d66d8e0ff6d6a10
551d43876a3fb2f502cd94c544cd566b9fbd348d
5595cd279af37197181c93dd0ae66446d01448f9
5bab9729665fcb4a78f5d119a8379bdf4e50dfdb
5d74b760265d3376895178839b620cd989fdca2d
5e6d5fa1d9db80e7d2ced115464e5b20e6bfb0e6
741989be0da31b7cab84793fcc2d1da6c6c34d7b
78e08e549bfa00d6d5a471fa3832d4f12c5e3a75
7de2cfa8e36d69f4ca875d93f83c783c9f05ee3f
86cae679347632f396bb90d95cdd577b5723d81c
88d8c85852d634500b944d22ab1dc0f849c8d52b
8b7c04861e9878325a5608d0a9ec07aa8b3885da
9272b8bab6f9adf4df9e3a5eaf5c1917b632b9a8
960abc20c9dcda33f41f5c6bede1f78f1019f316
9a22ae8f58667b82403bb2732fef1d94297e47ea
9c3dc069c1931192a64b9faae46f2a61777e5736
a440b484da9796dc1bf6e9744b0d74846b6eef19
a553dc777cce08c344f9fc2942d83062b1678e9b
a91f943c638eed9451cbc09f52b11b7f9c84f867
a958e561134e9403bf39660d90a94469f87b3645
af0ef2999b78e3c1a46a68e18ce6c2d48e131548
b438bc7ede48f24585ad86b41c3ca2ccb6a8d749
bce7f2ad0b32a27d894be639ed486b0dca107053
be0f15fb7e746121fb6ef02d3ed0f114b9d45c68
bf741ec139b0cb04063749f5d6f5ddf4e222bb37
c0995b46091decb389d626ef677483bc90560ab1
c233bab471f05fc767295251ccc1b8b4a3507eba
c2eac0f41492cf6792c5fa12d2a73da8dffe3b3d
c4545b290d062f939c769c48edf364e1c24c86fa
c525205a7083be332e30dcdc6290914eb271b441
cc0a546cff17ea7992b5747b8acf05eb10e0fbf1
cd6d09030433532d33bc886dae389b41c329f74c
cdbcc7c5d31e9ccad512540947cf9510dcb1d501
d4f5f1bcf0fa9571010f35863e55e5e837d48ebd
db03ebdc068a36626e37345813d0f28161a37d6b
e58987fac3b8cbdbd6e7408adbbfaeb5f60229e2

e95b90c253794f56d32b14b2849f329d3c50f122
eb15e8df6d93a797f1b6d8b90b6aaeea09638d5a
ed823a6bcd74a506822d161018d9f1903323f7e6
f025a8d501833ff4c35cd9444f2e49f05539da8e
f0e4b34ebfc5a0580a356f3b588c8185deed934f
f5682819b8adf51d7472b9487adf817dd0159216
fc71769a4bebb3f19e9a0111e16bd8fa2343ea13
fe3ca30da39f57ee5cdcf4ffe61afccc7bbfd181
ff6da20c870601023b8ebbdc3cb73e99226f237b

SHA1 Variant A DMG

0a87b12b2d12526c8ba287f0fb0b2f7b7e23ab4a
24c9f5c90ad325dae02aa52e2b1bac2857ae2faf
36997111b5e7aa81b430a72df9f54bac2a9695ba
7534b4ef7727d14b4fdd32d18651d32572c7747b

SHA1 Variant A Mach-O

0db22608be1172844c0ebf08d573ea4e7ef37308
2681a24f0ec0b1c153cc12d5d861c0c19c8383ea
385b9cc7d3147f049e7b42e97f242c5060fc9e97
46426409b9e65043b15ce2fcddd61213ff4e5156
48a0a7d4f0ae4b79b4f762857af3bbb02e8ab584
4f25d1a1aa18c8d85d555cd7a8f1cf2cf202af8c
58a3bddbc7c45193ecbefa22ad0496b60a29dff2
5d2e995fa5dce271ac5e364d7198842391402728
79007aabf9970e0aff7df52fd1c658b69f950c6f
793195d48cce96bb9b4fc1ee5bac03b371db75f7
82f4647e6783b012fc9a1f86108c644fcf491cf6
849cde22d1d188cc290bb527bbd7252ad07099af
9058ab6e05cb1f9ce77e4f8c18324a6827fb270d
97b19a82a32890d5ddaecac5a294cc3384309ea9
98f98a737a26c9dd1b27c474715976356ea4e18b
aab3a2897950e85a2b957f77d2f100e61e29061c
b42243d72765f142953bb26794b148858bff10a8
ca05f80fe44174d1089077f4b2303c436653226f
d5db5a11b9605d54cf66a153b0112b91c950d88f
d9d46ecfc1100d2b671ad97dc870e879d2634473
de465aad6cde9f0ce30fce0157bc18abf5a60d40
e114f643805394caece2326fb53e5d3a604a1aa9
f28025717f9db8a651f40c8326f477bf9d51a10f

SHA1 Variant B DMG

1f29b00c18bc0b7e1dfee5e79f8111da09f8fab8

SHA1 Variant B Mach-O

a02730f734032ed0f3b3705926b657aa4b88d720
c70fdf4362eb56032793ab08e6aeb892f1bd4a9b
e951b889aabca7ee5b0ff9d06a057884ed788b70

SHA1 Variant C Mach-O

00a20cf506e169b99e75e937e55c4b156a56304a
05138ad6617654e381b42ae37e1bf6bc552cd662
083e7453a1800ce808a38bda2f2d9344f1e6aff9
08713549eca50a3f4ee8c4dce32e713da1952423
10b3b243fcdd5368c13fbe84abbff7af0c13df51
148ed372fbf0b3bc19cc5c71977f61b8e41eb2da
14a87488243bf253f8165d4b42f4b739407c9906
19c9b3c9d0423c1817e165fd8315ce0a82034336
1a687586039804c905759e6bdc9fb16ab4a05741
1fc6a6a296103446edb51f5aba03f294a01ebc07
22bd2457a284ac88963e6e87eafbb7f7060605c6
2cbd24473f08bdce53a9ccc566ce817ea74e672f
3c8fc04ef41341ed60410959d7f9266e075a0c94
3dd4211432c79afa0534da3a88a6caab527282b0
3ec7e1274fd4f51deff02b51937953327034f5d6
40a97e141613e90907ed4dfa9c648e9ac05c5939
447c5949a04436f1ac479ea391a8cac38456bf8b
449bcac2b26d632d5a1d4f38b80349a6a440050b
4a9222757521855b9f6b6ab35583f2bf629c53e8
4d0b8212ab2a4631d2dc1a75f29ba786a69f7b2e
5028e9ddac3eb80dd57b3fd0b1943b200a5af8d0
6a3b6bc02121e7849f380c6420431e6165a5d5fc
6b464209db5802fbc510918c0cc5cc009cc8e966
7185a2eb6eb6873f82986c1e502678352ba1811b
75f8171a4636e2a518ae6709b3e86875f31ede59
7cece65179f21ea4d7e6e4778b0175418eb10171
7daedd153efa323eba2a22b843d400e515cf2e12
8751e7ca88e5c56ac928c70792e1fd33a6824d73
9b3b2270a7b1c6cd29ef6df13d9a2260b597f65f
9cadbd741f6e7547b0e6db38b47485dfd2a42948
9d62d9ea9ed7f49bcde0aef15bdba65888af737e
a1feed5da7c9363e3a5c67912c6a6d34c0f32997
a9a94ec7a6d06e5e44199160f756c7f728ca60b0
a9d71b86f4f0b356fd30d191692b805cb81d7e52
ad8be4808f7dd910cec11d7eed88933e3f50132a
c1c2c0630bbc8590e0f80e3bcf8c4d81de695284
c2861ae327abe194a39775f9dcbddf816eb3385f

c66fef5b2da022003386a3425c95adcf91bdacd4
c73ed38e8c9687add687ff7ef4639740f2f1a4a4
d7a69969f151fd1a712501a76f584580f3eab8a3
df0d85540e6d27858c7a750c11ead6c2cfc53e07
e2164b84808360299fee0ce3c303d9af1cfce8b2
e893136fda499d4534f9968eea14a39f6aabc9bf
ecd0361847c99008c1f181378ae99fb168463eed
f09021108fde30a9d51d0d47a02cf8ef24ef2e5a
fb77bce6ace6f6c506f5ae006fddd1a0b2e557da
fde1c0fa8a8ffc6ed704d4e082eb4ecba392d379
