AveMariaRAT

TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Contents

Overview

AveMariaRAT is a type of malicious software, also known as Warzone RAT. It is typically used to gain remote access capabilities by infecting systems. This Trojan was first spread through malicious phishing campaigns in 2018 and has since become more visible. Methods such as social engineering, email attachments, and malicious websites are used to infect users. Remote Access Tools (RATs) like AveMariaRAT can pose a serious risk to users' computer systems by being used by cybercriminals for espionage, data theft, and other malicious activities. This malicious software, once it infects a computer, exhibits behaviors such as:

- Remote control access,
- Downloading and deleting files,
- Recording keystrokes,
- Monitoring system information,
- Gaining access to data on browsers.

AveMariaRAT.exe Analysis

Name	AveMariaRAT.exe
MD5	d802bc50f7321efb13358d27280910ca
SHA256	45c59e6d1a36e978efffba98230fe70262b68748ff190562d2f2b8ccad7c43c7
File Type	Portable Executable 32 (x86)

The MD5, SHA256, and other such information about the malware are listed in the table above. The original name of the malware is "45c59e6d1a36e978efffba98230fe70262b68748ff190562d2f2b8ccad7c43c7.exe", but for ease of analysis, it has been renamed to **"AveMariaRAT.exe".**

Static Analysis

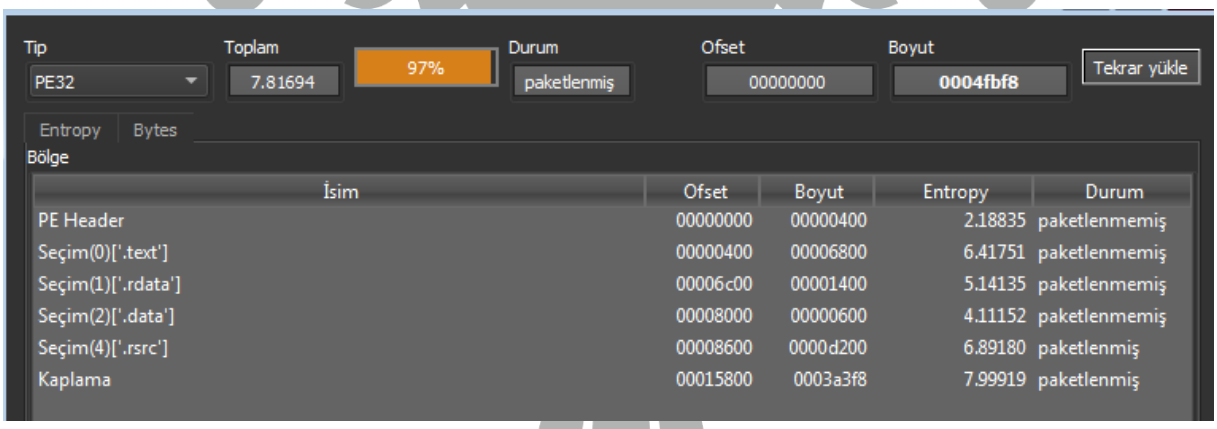

Figure 1 Examination of the Malware in the DIE Tool

When the AveMariaRAT.exe malware is examined in the DIE tool, it appears to **be packed.**

Advapi32.dll	Shell32.dll	Ole32.dll
Comctl32.dll	User32.dll	Gdi32.dll
Kernel32.dll		

Table 1 Some DLLs Used by the Malware

Table 1 shows some of the DLLs used by the malware.

Dynamic Analysis

Uxtheme.dll	Userenv.dll	Setupapi.dll
Apphelp.dll	Propsys.dll	Dwmapi.dll
Cryptbase.dll	Oleacc.dll	Clbcatq.dll
Ntmarta.dll		

Table 2 Dynamically Extracted DLLs

Some of the dynamically loaded DLLs are shown in Table 2.

```
push 0                                                                  
inc ecx                                                                 
neg ecx                                                                 
sbb ecx,ecx                                                             
and ecx,eax                                                             
push ecx                                                                
push dword ptr ss:[esp+14]                                              
push 0                                                                  
push 1                                                                  
push dword ptr ss:[esp+1C]      [esp+1C]:L"C:\\Users\\      \\AppData\\Local\\Temp\\qqscaq.po"
push dword ptr ss:[esp+1C]      [esp+1C]:L"C:\\Users\\      \\AppData\\Local\\Temp\\qqscaq.po"
call dword ptr ds:[<&CreateFileW>]                                      
ret C                                                                   
push ebp                                                                
mov ebp,esp                                                             
push ecx                                                                
```

Figure 2 Creation of the "qqscaq.po" File Using the CreateFileW API

The malware creates a file named **"qqscaq.po"** in the **"C:\Users\%username%\AppData\Local\Temp"** location using the **CreateFileW** API.

```
mov ecx,eax
push 0
inc ecx
neg ecx
sbb ecx,ecx
and ecx,eax
push ecx
push dword ptr ss:[esp+14]
push 0
push 1
push dword ptr ss:[esp+1C]          [esp+1C]:L"C:\\Users\\      \\AppData\\Local\\Temp\\alaiw.exe"
push dword ptr ss:[esp+1C]          [esp+1C]:L"C:\\Users\\      \\AppData\\Local\\Temp\\alaiw.exe"
call dword ptr ds:[<&CreateFileW>]
ret C
push ebp
mov ebp,esp
push ecx
```

Figure 3 Creation of the "alaiw.exe" File Using the CreateFileW API

After that, the malware uses the CreateFileW API again to create an executable file (PE) named **"alaiw.exe"** in the **"C:\Users\%username%\AppData\Local\Temp"** location.

```
00403F8C   57                push edi
00403F8D   83C0 69           add eax,69
00403F90   68 C5404000       push avemariarat.4040C5
00403F95   0FB7C0            movzx eax,ax
00403F98   57                push edi
00403F99   50                push eax
00403F9A   FF35 60A24200     push dword ptr ds:[42A260]
00403FA0   FF15 2C824000     call dword ptr ds:[<&DialogBoxParamW>]
00403FA6   6A 05             push 5
00403FA8   8BF0              mov esi,eax
00403FAA   E8 5CD4FFFF       call avemariarat.40140B
00403FAF   6A 01             push 1
00403FB1   E8 B1FCFFFF       call avemariarat.403C67
00403FB6   8BC6              mov eax,esi
```

Figure 4 Execution of Shellcode Using the DialogBoxParamW API

The malware executes the **shellcode** stored in memory using the **DialogBoxParamW** WinAPI call, as shown in Figure 4.

```
push eax
xor eax,eax
push avemariarat.426750
push eax
push eax
push 4000000
push eax
push eax
push eax
push dword ptr ss:[ebp+8]           [ebp+8]:L"\"C:\\Users\\      \\AppData\\Local\\Temp\\alaiw.exe\" "
push eax
call dword ptr ds:[<&CreateProcessW>]
test eax,eax
je avemariarat.405C8A
push dword ptr ss:[ebp-C]           [ebp-C]:L"\"C:\\Users\\      \\AppData\\Local\\Temp\\alaiw.exe\" "
call dword ptr ds:[<&CloseHandle>]
mov eax,dword ptr ss:[ebp-10]
```

Figure 5 Execution of "alaiw.exe" Using the CreateProcessW API

Then, it executes the previously created **"alaiw.exe"** using the **CreateProcessW** API.

alaiw.exe Analysis

Name	alaiw.exe
MD5	fa0be3eb24b13d060a0ae4e25c22ef1c
SHA256	54152ed7b7386c7a7bef26fafcc72fe3d51ddfbb677292bd9d1261b2c6199ebd
File Type	Portable Executable 32 (x86)

The MD5, SHA256, and other such information of the alaiw.exe file created in the **"C:\Users\%username%\AppData\Local\Temp"** directory within AveMariaRAT.exe are listed in the table above.

Static Analysis

When examined with the DIE tool, alaiw.exe appears to **be packed.**

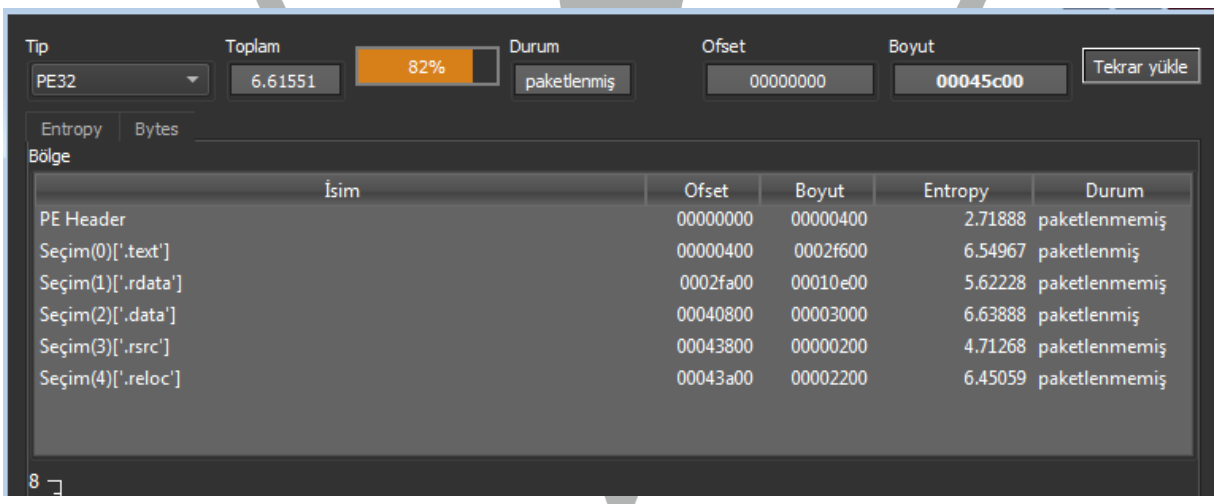

Figure 6 Examination of alaiw.exe in the DIE Tool

Kernel32.dll	Winspool.drv	Crypt32.dll
Loadperf.dll	Wininet.dll	Rtutils.dll
User32.dll		

Table 3 Some DLLs Used

Table 3 shows some of the DLLs used by the malware.

Dynamic Analysis

The malware decrypts the previously created encrypted **"qqscaq.po"** file using the **"vtwkwntewuzvb"** key phrase. In this way, it creates the shellcode. This code is then copied into memory allocated by the **VirtualAlloc** API using the memmove function. Then, it executes the shellcode using the **EnumTimeFormatsA** API.

```
.text:00B61B37
.text:00B61B37 loc_B61B37:
.text:00B61B37 cmp     [ebp+var_18], 1FB9h ; Compare Two Operands
.text:00B61B3E jnb     short loc_B61B89 ; Jump if Not Below (CF=0)
```

```
.text:00B61B89
.text:00B61B89 loc_B61B89:             ; flProtect
.text:00B61B89 push    40h ; '@'
.text:00B61B8B push    3000h           ; flAllocationType
.text:00B61B90 push    1FB9h           ; dwSize
.text:00B61B95 push    0               ; lpAddress
.text:00B61B97 call    ds:VirtualAlloc ; Indirect Call Near Procedure
.text:00B61B9D mov     [ebp+lpTimeFmtEnumProc], eax
.text:00B61BA0 push    1FB9h           ; Size
.text:00B61BA5 push    offset byte_BA2000 ; Src
.text:00B61BAA mov     edx, [ebp+lpTimeFmtEnumProc]
.text:00B61BAD push    edx             ; void *
.text:00B61BAE call    _memmove        ; Call Procedure
.text:00B61BB3 add     esp, 0Ch        ; Add
.text:00B61BB6 push    0               ; dwFlags
.text:00B61BB8 push    0               ; Locale
.text:00B61BBA mov     eax, [ebp+lpTimeFmtEnumProc]
.text:00B61BBD push    eax             ; lpTimeFmtEnumProc
.text:00B61BBE call    ds:EnumTimeFormatsA ; Indirect Call Near Procedure
.text:00B61BC4 push    offset aEnterYourExpre ; "Enter your expression (use spaces betwe"...
.text:00B61BC9 push    offset unk_BA5208
.text:00B61BCE call    sub_B62380      ; Call Procedure
.text:00B61BD3 lea     ecx, [ebp+var_5C] ; Load Effective Address
.text:00B61BD6 push    ecx
.text:00B61BD7 push    offset dword_BA5190
.text:00B61BDC call    sub_B644A0      ; Call Procedure
.text:00B61BE1 mov     byte ptr [ebp+var_4], 1
.text:00B61BE5 lea     edx, [ebp+var_5C] ; Load Effective Address
.text:00B61BE8 push    edx
.text:00B61BE9 call    sub_B61530      ; Call Procedure
.text:00B61BEE fstp    [ebp+var_3C]    ; Store Real and Pop
.text:00B61BF1 movsd   xmm0, [ebp+var_3C] ; Move Scalar Double-Precision Floating-Point Values
.text:00B61BF6 movsd   [ebp+var_44], xmm0 ; Move Scalar Double-Precision Floating-Point Values
.text:00B61BFB push    offset aResult  ; "Result: "
.text:00B61C00 push    offset unk_BA5208
.text:00B61C05 call    sub_B62380      ; Call Procedure
.text:00B61C0A mov     [ebp+var_28], eax
.text:00B61C0D sub     esp, 8          ; Integer Subtraction
.text:00B61C10 movsd   xmm0, [ebp+var_44] ; Move Scalar Double-Precision Floating-Point Values
.text:00B61C15 movsd   [esp+70h+var_70], xmm0 ; Move Scalar Double-Precision Floating-Point Values
.text:00B61C1A mov     ecx, [ebp+var_28]
.text:00B61C1D call    sub_B66420      ; Call Procedure
.text:00B61C22 mov     [ebp+var_2C], eax
.text:00B61C25 push    offset sub_B64250
.text:00B61C2A mov     ecx, [ebp+var_2C]
.text:00B61C2D call    sub_B665A0      ; Call Procedure
.text:00B61C32 jmp     short loc_B61C69 ; Jump
```

```
.text:00B61B40 mov     ecx, [ebp+var_18]
.text:00B61B43 mov     dl, byte_BA2000[ecx]
.text:00B61B49 mov     [ebp+var_11], dl
.text:00B61B4C mov     eax, [ebp+var_24]
.text:00B61B4F add     eax, [ebp+var_1C] ; Add
.text:00B61B52 mov     cl, [eax]
.text:00B61B54 mov     [ebp+var_12], cl
.text:00B61B57 movzx   edx, [ebp+var_12] ; Move with Zero-Extend
.text:00B61B5B movzx   eax, [ebp+var_11] ; Move with Zero-Extend
.text:00B61B5F shl     eax, 5           ; Shift Logical Left
.text:00B61B62 movzx   ecx, [ebp+var_11] ; Move with Zero-Extend
.text:00B61B66 sar     ecx, 3           ; Shift Arithmetic Right
.text:00B61B69 or      ecx, eax         ; Logical Inclusive OR
.text:00B61B6B xor     ecx, edx         ; Logical Exclusive OR
.text:00B61B6D mov     edx, [ebp+var_18]
.text:00B61B70 mov     byte_BA2000[edx], cl
.text:00B61B76 mov     eax, [ebp+var_1C]
.text:00B61B79 add     eax, 1           ; Add
.text:00B61B7C cdq                      ; EAX -> EDX:EAX (with sign)
.text:00B61B7D mov     ecx, 0Dh
.text:00B61B82 idiv    ecx              ; Signed Divide
.text:00B61B84 mov     [ebp+var_1C], edx
.text:00B61B87 jmp     short loc_B61B2E ; Jump
```

```
.text:00B61B2E
.text:00B61B2E loc_B61B2E:
.text:00B61B2E mov     eax, [ebp+var_18]
.text:00B61B31 add     eax, 1           ; Add
.text:00B61B34 mov     [ebp+var_18], eax
```

Figure 7 Decryption and Execution of Shellcode

```
Adres    Hex                                                UNICODE
0053F434 0D 30 4D 77 48 7A 2E 00 00 00 00 00 2A 00 00 2A    ......*.
0053F444 C4 00 2E 00 3D 00 00 3D 00 00 00 00 64 F4 53 00    Ä.=....S
0053F454 00 00 00 00 80 00 00 00 58 00 00 00 10 02 00 12    ....X...
0053F464 10 9A 30 00 01 00 00 00 00 00 00 00 0A 00 00 00    .0......
0053F474 0E 00 00 00 F8 95 30 00 F8 95 30 00 F3 95 30 00    ...0.0.0
0053F484 00 00 00 00 2C 00 00 00 00 00 2E 00 34 F5 53 01    ..,.....
0053F494 DC F3 53 00 10 9A 30 00 3C F7 53 00 CD 4D 51 77    .S.0.S..
0053F4A4 F1 20 78 04 FE FF FF FF 36 34 4D 77 61 34 4D 77
0053F4B4 12 00 00 00 20 00 00 00 F2 95 30 00 F0 95 30 00    .. ..0.0
0053F4C4 12 00 00 00 09 00 00 00 00 00 00 00 79 00 75 00    .....yu
0053F4D4 65 00 61 00 61 00 6A 00 73 00 73 00 6F 00 78 00    eaajssox
0053F4E4 78 00 68 00 64 00 64 00 00 00 8D 76 6D 00 69 00    xhdd..mi
0053F4F4 72 00 72 00 62 00 77 00 77 00 67 00 70 00 2E 00    rrbwwgp.
0053F504 65 00 78 00 65 00 00 00 70 00 6C 00 75 00 70 00    exe.plup
0053F514 70 00 79 00 69 00 69 00 65 00 6E 00 00 00 8C 76    pyiien..
0053F524 71 00 71 00 73 00 63 00 61 00 71 00 2E 00 70 00    qqscaq.p
0053F534 6F 00 00 00 00 00 00 00 00 00 00 00 00 88 01 00    o.......
0053F544 58 00 00 00 50 34 2E 00 00 00 00 00 78 F5 53 00    X......S
0053F554 78 F5 53 00 27 91 8C 76 F8 95 30 00 58 00 00 00    .S...OX.
0053F564 50 4E 30 00 00 00 14 00 00 00 00 00 F8 95 30 00    .0.....0
0053F574 00 00 00 00 78 F6 53 00 1F B2 8C 76 1F 04 00 00    ...S...
0053F584 00 00 14 00 00 00 00 00 D6 F5 53 00 00 00 00 00    .....S..
0053F594 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .......
```

Figure 8 Decrypted Strings within the Shellcode

The malware then places the strings **yueaajssoxxhdd**, **mirrbwwgp.exe**, **pluppyiien**, and **qqscaq.po** in memory using **deobfuscation.**

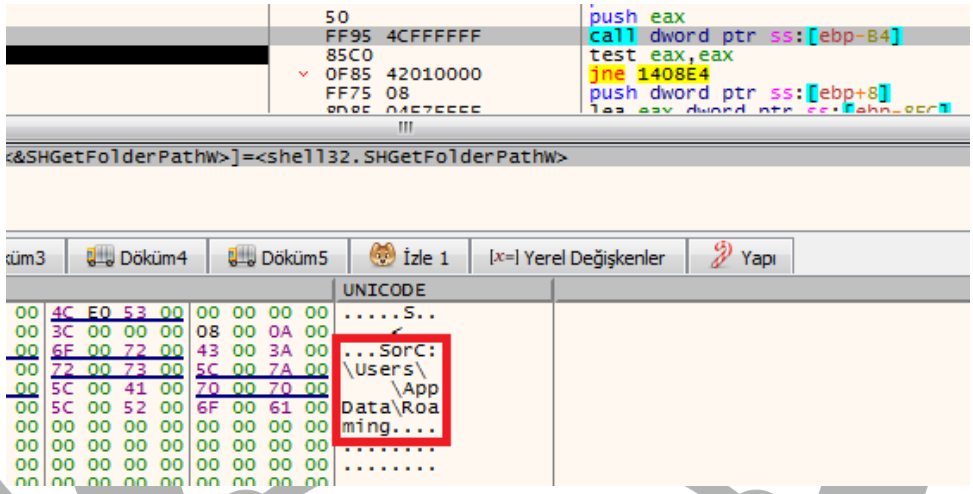

Figure 9 Accessing File Path Using the SHGetFolderPathW API

The malware accesses the file path **"C:\Users\%username%\AppData\Roaming"** using the **SHGetFolderPathW** API.

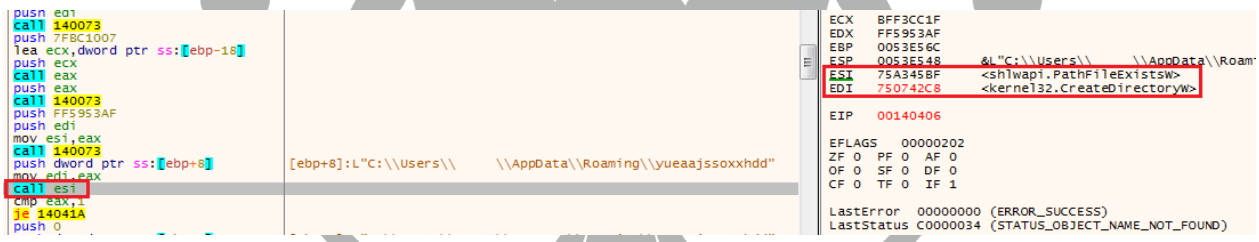

Figure 10 "PathFileExistW" and "CreateDirectoryW" APIs

It checks for the existence of the directory **"C:\Users\%username%\AppData\Roaming\yueaajssoxxhdd"** using the **PathFileExistW** API, and if it doesn't exist, creates it using the **CreateDirectoryW** API.

Figure 11 Dynamic API Resolution

The call instructions shown in Figure 11, respectively, resolve the **LoadLibrary**, **PathFileExistsW**, **CreateFileW**, **GetFileSize**, **VirtualAlloc**, **ReadFile**, **CloseHandle**, and **WriteFile** APIs and store their addresses in memory.

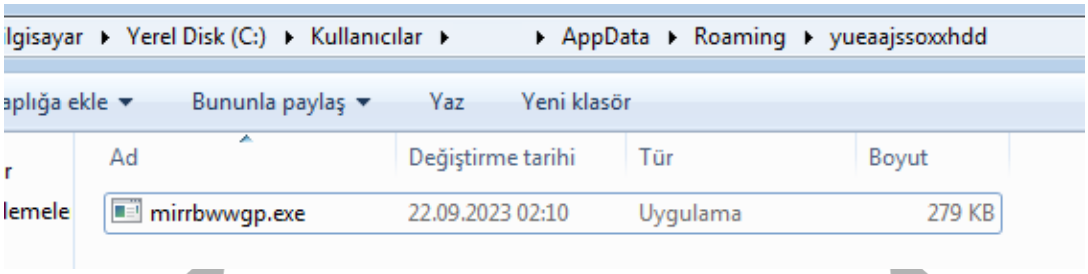

Figure 12 alaiw.exe Copying Itself as mirrbwwgp.exe

Using these APIs, it copies itself as **"mirrbwwgp.exe"** into the directory **"C:\Users\%username%\AppData\Roaming\yueaajssoxxhdd"**.

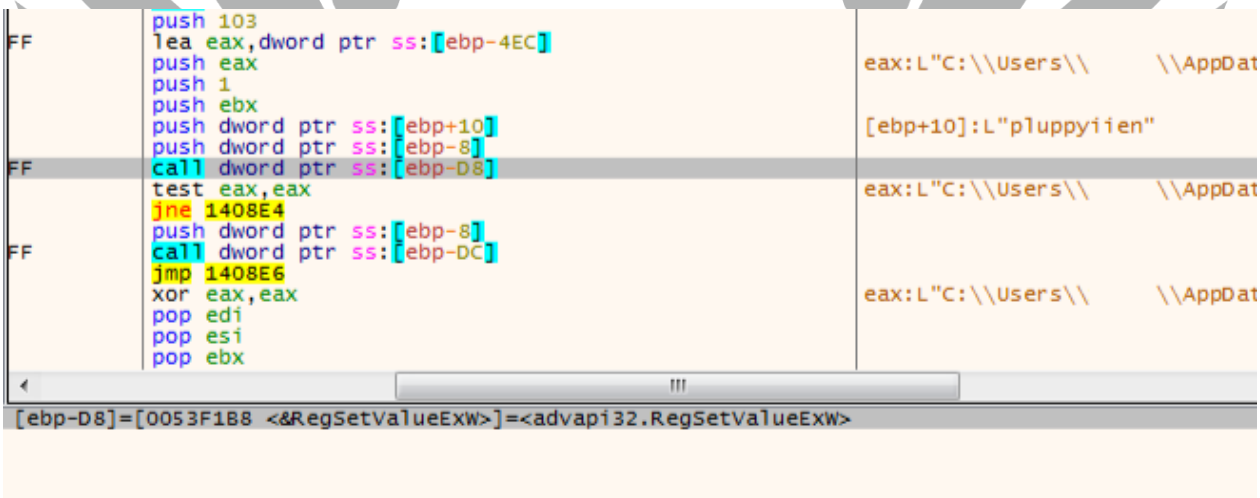

Figure 13 Ensuring Persistence with RegSetValueExW

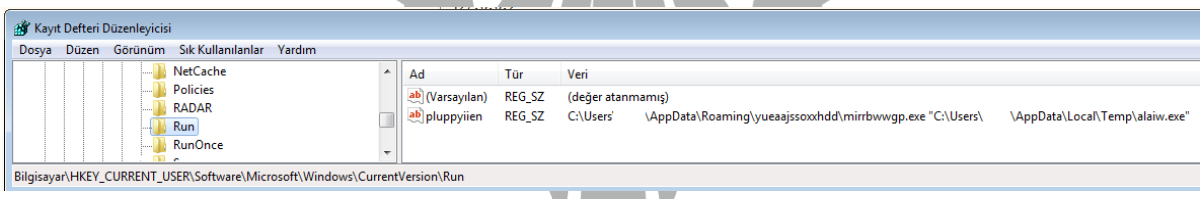

Figure 14 Creation of the "pluppyiien" Key

To ensure its persistence, it accesses the registry and creates the **"pluppyiien"** key at the location **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run**. This way, the malware executes itself every time the system starts.

Figure 15 Algorithm Used to Decrypt Data

The malware then uses a unique algorithm to decrypt **encrypted data**.

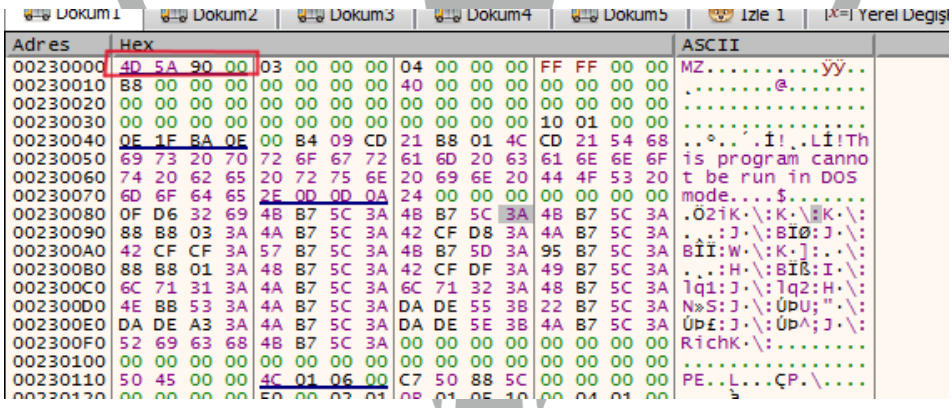

Figure 16 The Memory Region Containing the Executable File

When this data is examined in the memory region, it appears to be an **executable file**.

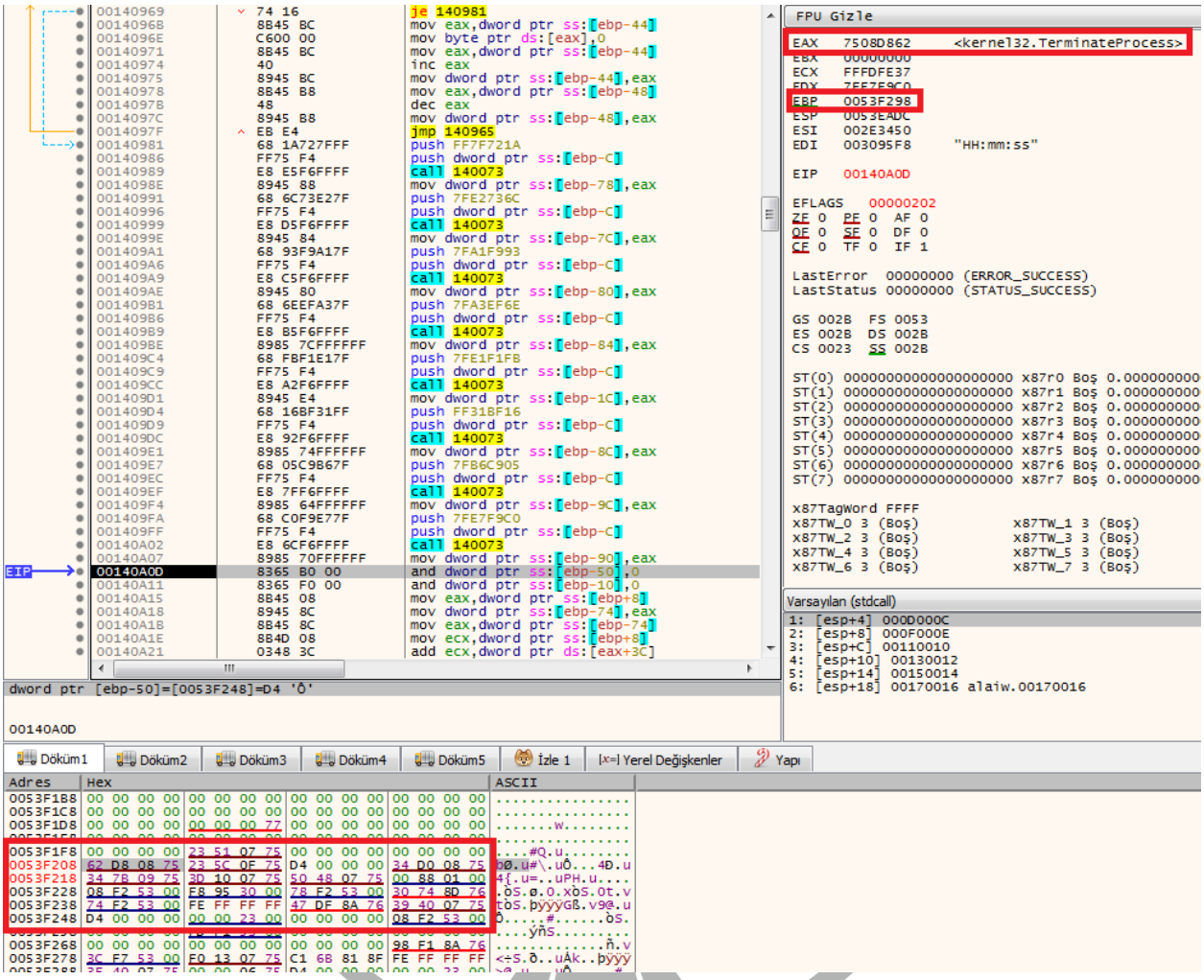

Figure 17 Dynamic API Resolution

Then, it resolves the **GetModuleFileNameW**, **CreateProcessW**, **GetThreadContext**, **ReadProcessMemory**, **CloseHandle**, **SetThreadContext**, **GetCommandLineW**, and **TerminateProcess** APIs in sequence with the call instructions shown in Figure 17 and stores their addresses in memory.

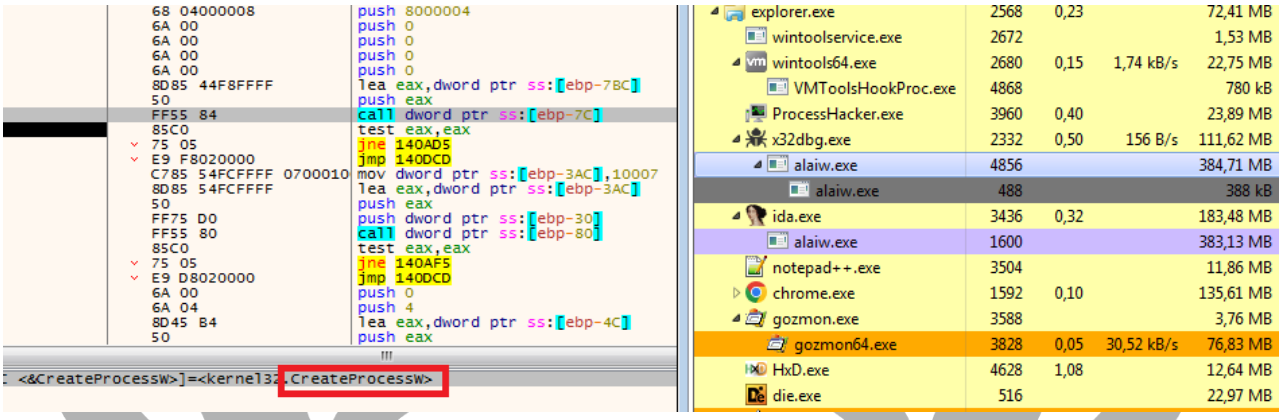

Figure 18 File Running in Suspend Mode

Using the **GetModuleFileNameW** API, it gets the path to the file it is located in. Then, it starts alaiw.exe in **suspend mode** as a child process by giving this file path as a parameter to the **CreateProcessW** API.

It writes the data decrypted with the algorithm in Figure 15 into alaiw.exe running in **suspend mode** using the GetThreadContext, ReadProcessMemory, and SetThreadContext APIs. While alaiw.exe is running as a **child process**, it continues to run as the **parent process** after this process.

The memory region where the codes were added to alaiw.exe (Figure 16) was **dumped** and continued to be examined.

warzone160.exe Analysis

Name	warzone160.exe
MD5	bfa56fb7698757d5316e3cd458008541
SHA256	a4470593f2ebc45b1be6f2d432c90f1a5120dab98427bb6aed8319235b52d4cb
File Type	Portable Executable 32 (x86)

The **dumped** file has been named **"warzone160"**. Information such as its MD5 and SHA256 is listed in the table above.

Static Analysis

When warzone160.exe is examined in the DIE tool, it appears to **be packed**.

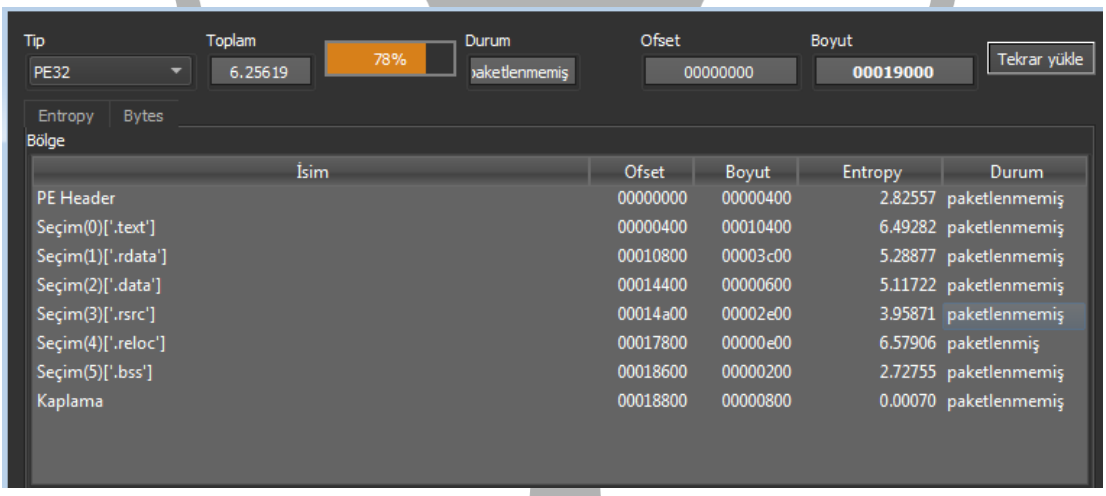

Figure 19 Examination of warzone160.exe in the DIE Tool

Dynamic Analysis

4587	14113.566432	192.168.96.132	194.180.48.209	TCP	66	49559 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
4588	14115.942755	194.180.48.209	192.168.96.132	TCP	60	9409 → 49559 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
4589	14116.449734	192.168.96.132	194.180.48.209	TCP	66	[TCP Retransmission] 49559 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
4590	14118.840999	194.180.48.209	192.168.96.132	TCP	60	9409 → 49559 [RST, ACK] Seq=34622698 Ack=1 Win=64240 Len=0
4591	14119.350810	192.168.96.132	194.180.48.209	TCP	62	[TCP Retransmission] 49559 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM
4592	14121.253851	VMware_cc:b3:5d	VMware_fb:4d:28	ARP	42	Who has 192.168.96.2? Tell 192.168.96.132
4593	14121.254087	VMware_fb:4d:28	VMware_cc:b3:5d	ARP	60	192.168.96.2 is at 00:50:56:fb:4d:28
4594	14121.661752	194.180.48.209	192.168.96.132	TCP	60	9409 → 49559 [RST, ACK] Seq=4197400481 Ack=1 Win=64240 Len=0
4595	14122.876918	192.168.96.1	239.255.255.250	SSDP	212	M-SEARCH * HTTP/1.1
4596	14123.878035	192.168.96.1	239.255.255.250	SSDP	212	M-SEARCH * HTTP/1.1
4597	14124.878321	192.168.96.1	239.255.255.250	SSDP	212	M-SEARCH * HTTP/1.1
4598	14125.879197	192.168.96.1	239.255.255.250	SSDP	212	M-SEARCH * HTTP/1.1
4599	14126.028754	192.168.96.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4600	14126.676157	192.168.96.132	192.168.96.2	DNS	81	Standard query 0xdde3 A septembre.duckdns.org
4601	14126.678106	192.168.96.2	192.168.96.132	DNS	97	Standard query response 0xdde3 A septembre.duckdns.org A 194.180.48.209
4602	14126.678334	192.168.96.132	194.180.48.209	TCP	66	49560 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
4603	14127.030799	192.168.96.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4604	14128.032056	192.168.96.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4605	14129.033636	192.168.96.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4606	14129.053151	194.180.48.209	192.168.96.132	TCP	60	9409 → 49560 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
4607	14129.559685	192.168.96.132	194.180.48.209	TCP	66	[TCP Retransmission] 49560 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
4608	14131.933621	194.180.48.209	192.168.96.132	TCP	60	9409 → 49560 [RST, ACK] Seq=1647120699 Ack=1 Win=64240 Len=0
4609	14132.445641	192.168.96.132	194.180.48.209	TCP	62	[TCP Retransmission] 49560 → 9409 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM
4610	14134.763000	194.180.48.209	192.168.96.132	TCP	60	9409 → 49560 [RST, ACK] Seq=4199943123 Ack=1 Win=64240 Len=0
4611	14136.329689	192.168.96.1	192.168.96.255	UDP	86	57621 → 57621 Len=44

Figure 20 Data Obtained in Wireshark

The malware is continuously attempting to communicate with the command-and-control server by trying to connect to the socket **194[.]180[.]48[.]209[:]9409**. This process is repeated constantly because the **connection cannot be established**.

When the malware is running, it has been observed that it attempts to connect to the domain **"septembre[.]duckdns[.]org"**. However, the server is not active, so the connection cannot be established. Therefore, the analysis has been continued **statically**.

```
push 1C
pop edx                                          edx:EntryPoint
lea ecx,dword ptr ss:[ebp-10]
call warzone160.D1D51C
push warzone160.D22938                           D22938:L"\\Google\\Chrome\\User Data\\Default\\Login D
lea ecx,dword ptr ss:[ebp-10]
call warzone160.D13230
push dword ptr ss:[ebp-10]
call dword ptr ds:[<&PathFileExistsW>]
```

Figure 21 Google Chrome Browser

```
push eax
push 20019
push 0
push warzone160.D2351C                           D2351C:"software\\Aerofox\\FoxmailPreview"
push 80000001
call dword ptr ds:[<&RegOpenKeyExA>]
test eax,eax
jne warzone160.D18912
```

Figure 22 Foxmail Email Service

```
push eax
call warzone160.D11052
add esp,14
lea edx,dword ptr ss:[ebp-2BC]                   edx:EntryPoint
mov ecx,warzone160.D22F0C                        D22F0C:L"thunderbird.exe"
call warzone160.D1ABE2
pop ecx
lea eax,dword ptr ss:[ebp-2BC]
push eax
lea ecx,dword ptr ss:[ebp-28]
call warzone160.D133AB
lea eax,dword ptr ss:[ebp-98]
push eax
lea eax,dword ptr ss:[ebp-2BC]
push eax
call dword ptr ds:[<&GetBinaryTypeW>]
push ecx
lea eax,dword ptr ss:[ebp-28]
mov ecx,esp
push eax
call warzone160.D133F3
mov ecx,edi
call warzone160.D1A190
test eax,eax
jne warzone160.D192A1
push ecx
lea eax,dword ptr ss:[ebp-28]
mov ecx,esp
push eax
call warzone160.D133F3
mov ecx,edi
call warzone160.D1A190
test eax,eax
jne warzone160.D192A1
mov esi,dword ptr ss:[ebp-14]
jmp warzone160.D19702
push warzone160.D22F2C                           D22F2C:L"\\Thunderbird\\"
lea ecx,dword ptr ss:[ebp+8]
call warzone160.D13230
lea eax,dword ptr ss:[ebp+8]
push eax
lea ecx,dword ptr ss:[ebp-20]
call warzone160.D133F3
push warzone160.D22EA0                           D22EA0:L"profiles.ini"
lea ecx,dword ptr ss:[ebp-20]
call warzone160.D13230
push warzone160.D22E38                           D22E38:L"Profile"
lea ecx,dword ptr ss:[ebp-24]
call warzone160.D133AB
push eax
lea ecx,dword ptr ss:[ebp-14]
call warzone160.D131FD
mov ecx,dword ptr ss:[ebp-24]
call warzone160.D158FB
push ebx
lea ecx,dword ptr ss:[ebp-14]
call warzone160.D13038
push dword ptr ss:[ebp-20]
push esi
jmp warzone160.D196D1
mov esi,dword ptr ss:[ebp-34]
lea ecx,dword ptr ss:[ebp-38]
inc esi
push warzone160.D22E38                           D22E38:L"Profile"
mov dword ptr ss:[ebp-34],esi
call warzone160.D133AB
```

Figure 23 Thunderbird Email Service

When static analysis and string scanning were performed, as seen in Figures 21, 22, and 23, it was understood that the malware was targeting data such as passwords and **client settings** from some browsers and **email** services.

```
00012CA0  6E 00 74 00 56 00 65 00 72 00 73 00 69 00 6F 00  n.t.V.e.r.s.i.o.
00012CB0  6E 00 5C 00 52 00 75 00 6E 00 5C 00 00 00 00 00  n.\.R.u.n.\.....
00012CC0  63 6D 64 2E 65 78 65 20 2F 43 20 70 69 6E 67 20  cmd.exe /C ping
00012CD0  31 2E 32 2E 33 2E 34 20 2D 6E 20 32 20 2D 77 20  1.2.3.4 -n 2 -w
00012CE0  31 30 30 30 20 3E 20 4E 75 6C 20 26 20 44 65 6C  1000 > Nul & Del
00012CF0  20 2F 66 20 2F 71 20 00 22 00 00 00 53 00 4F 00   /f /q ."...S.O.
00012D00  46 00 54 00 57 00 41 00 52 00 45 00 5C 00 5F 00  F.T.W.A.R.E.\._.
00012D10  72 00 70 00 74 00 6C 00 73 00 00 00 49 00 6E 00  r.p.t.l.s...I.n.
00012D20  73 00 74 00 61 00 6C 00 6C 00 00 00 5C 00 53 00  s.t.a.l.l...\.S.
00012D30  79 00 73 00 74 00 65 00 6D 00 33 00 32 00 5C 00  y.s.t.e.m.3.2.\.
00012D40  63 00 6D 00 64 00 2E 00 65 00 78 00 65 00 00 00  c.m.d...e.x.e...
```

Figure 24 CMD Command

The **Cmd** command shown in Figure 24 aims to complicate the detection of the attack through ping requests sent to an **invalid** IP address. Additionally, the **"Del /f /q"** command is intended to delete the malware without permission in case it is detected.

```
call dword ptr ds:[<&GetAsyncKeyState>]
test ax,ax
mov dl,bl
setne cl
call warzone160.D17949
test al,al
lea ecx,dword ptr ds:[esi+20]
call dword ptr ds:[<&wsprintfW>]
add esp,C
lea ecx,dword ptr ss:[ebp-14]
call warzone160.D17966
mov ebx,dword ptr ss:[ebp-4]
jmp warzone160.D1790B
cmp esi,66                                  66:'f'
ja warzone160.D1771D
je warzone160.D17713
cmp esi,20                                  20:' '
ja warzone160.D17691
je warzone160.D17687
cmp esi,11
ja warzone160.D1765B
je warzone160.D17781
sub esi,8
je warzone160.D17651
sub esi,1
je warzone160.D17647
sub esi,4
je warzone160.D1763D
sub esi,3
je warzone160.D1790B
jmp warzone160.D178A2
mov ecx,warzone160.D22814                   D22814:L"[ENTER]\r\n"
jmp warzone160.D17906
mov ecx,warzone160.D22838                   D22838:L"[TAB]"
jmp warzone160.D17906
mov ecx,warzone160.D22828                   D22828:L"[BKSP]"
jmp warzone160.D17906
sub esi,12
je warzone160.D177D8
dec esi
sub esi,1
je warzone160.D1767D
sub esi,7
jne warzone160.D178A2
mov ecx,warzone160.D22870                   D22870:L"[ESC]"
jmp warzone160.D17906
mov ecx,warzone160.D22860                   D22860:L"[CAPS]"
jmp warzone160.D17906
mov ecx,warzone160.D22810
jmp warzone160.D17906
cmp esi,62                                  62:'b'
ja warzone160.D176E2
je warzone160.D176D8
sub esi,2D
je warzone160.D176CE
sub esi,1
je warzone160.D176C4
sub esi,32
je warzone160.D176BA
sub esi,1
jne warzone160.D178A2
mov ecx,warzone160.D228A0
jmp warzone160.D17906
mov ecx,warzone160.D2289C
jmp warzone160.D17906
mov ecx,warzone160.D22890                   D22890:L"[DEL]"
jmp warzone160.D17906
mov ecx,warzone160.D2287C                   D2287C:L"[INSERT]"
jmp warzone160.D17906
mov ecx,warzone160.D228A4
jmp warzone160.D17906
sub esi,63
```

Figure 25 Keylogger

Figure 25 shows that the malware records special **keystrokes** like ENTER, TAB, BKSP, ESC, CAPS, DEL, and INSERT using the **GetAsyncKeyState** API.

```
ush ebp
ov ebp,esp
ub esp,18
ush esi
ush edi
or edi,edi
ea ecx,dword ptr ss:[ebp-C]
ush warzone160.D23688                    D23688:L"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters"
ov dword ptr ss:[ebp-4],edi
all warzone160.D133AB
ea eax,dword ptr ss:[ebp-4]
ov dword ptr ss:[ebp-14],edi
ush eax
ush 20119
ush edi
ush dword ptr ss:[ebp-C]
ov dword ptr ss:[ebp-10],edi
ush 80000002
all dword ptr ds:[<&RegOpenKeyExW>]
est eax,eax
ne warzone160.D1BFD0
ea eax,dword ptr ss:[ebp-14]
ush eax
ush warzone160.D236FC                    D236FC:L"ServiceDll"
ea ecx,dword ptr ss:[ebp-8]
all warzone160.D133AB
```

Figure 26 Remote Access

The malware accesses the ServiceDll record in the **"SYSTEM\CurrentControlSet\Services\TermService\Parameters"** path. This action enables remote access to the device through the **Remote Desktop Protocol (RDP)** and makes it possible to control the device.

YARA Rules

```
import "hash"

rule avemariarat {

        meta:

                author = "Team-5"

    strings:

                $hex_1 = { 55 58 54 48 45 4D 45 00 55 53 45 52 45 4E 56 00 53 45 54
55 50 41 50 49 00 41 50 50 48 45 4C 50 00 50 52 4F 50 53 59 53 00 44 57 4D 41 50
49 00 43 52 59 50 54 42 41 53 45 00 4F 4C 45 41 43 43 00 43 4C 42 43 41 54 51 00
4E 54 4D 41 52 54 41 }

                $hex_2 = { 50 33 C0 68 50 67 42 00 50 50 68 00 00 00 04 50 50 50 FF
75 08 50 FF 15 ?? ?? ?? ?? }

                $str1 = "http://nsis.sf.net/NSIS_Error" wide

                $str2 = "\\Microsoft\\Internet Explorer\\Quick Launch" wide

                $api1 = "DialogBoxParamW" ascii

                $api2 = "RegSetValueExW" ascii

                $api3 = "CreateProcessW" ascii

                $api4 = "ExitProcess" ascii

                $api5 = "WriteFile" ascii

                $api6 = "FindNextFileW" ascii

    condition:

      hash.md5 (0, filesize) == "d802bc50f7321efb13358d27280910ca" or (all of ($str*)
and (5 of ($api*))) or (all of ($hex_*)) }
```

```
import "hash"

rule alaiw_d {

        meta:

                author = "Team-5"

                description = "AveMariaRAT"

                weight = “10”

    strings:

        $algorithm1 = { C1 E0 05 0F B6 4D EF C1 F9 03 0B C8 33 CA 8B
55 E8 88 8A ?? ?? ?? ?? } //Shellcode decryption algorithm

        $str1 = "vtwkwntewuzvb"

        $str2 = "find.exe"

        $str3 = "-w %ws -d C -f %s"

        $str4 = "\\System32\\cmd.exe"

        $str5 = "SELECT * FROM logins"

        $str6 = "Accounts\\Account.rec0"

        $str7 = "cmd.exe /C ping 1.2.3.4 -n 2 -w 1000 > Nul & Del /f /q"

        $str8 = "SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList"
```

```
        $w1 = "http://5.206.225.104/dll/msvcp140.dll" wide

        $w2 = "http://5.206.225.104/dll/softokn3.dll" wide

        $w3 = "http://5.206.225.104/dll/mozglue.dll" wide

        $w4 = "http://5.206.225.104/dll/vcruntime140.dll" wide

        $w5 = "http://5.206.225.104/dll/freebl3.dll" wide

        $w6 = "http://5.206.225.104/dll/nss3.dll" wide

        $w7 =
"C:\\Users\\louis\\Documents\\workspace\\MortyCrypter\\MsgBox.exe"
wide

        $w8 = "\\Google\\Chrome\\User Data\\Default\\Login Data" wide

        $w9 = "profiles.ini" wide

    condition:

      hash.md5(0,filesize)  ==  "fa0be3eb24b13d060a0ae4e25c22ef1c"  or
(((5 of $str*) or (7 of $w*)) or ($algorithm1 and (2 of $w*)))

}
```

```
rule warzone {
        meta:
                author = "Team-5"
    strings:
        $str1 = "find.exe"
        $str2 = "-w %ws -d C -f %s"
        $str3 = "\\System32\\cmd.exe"
        $str4 = "SELECT * FROM logins"
        $str5 = "Accounts\\Account.rec0"
        $str6 = "cmd.exe /C ping 1.2.3.4 -n 2 -w 1000 > Nul & Del /f /q"
        $str7 = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList"

        $w1 = "http://5.206.225.104/dll/msvcp140.dll" wide
        $w2 = "http://5.206.225.104/dll/softokn3.dll" wide
        $w3 = "http://5.206.225.104/dll/mozglue.dll" wide
        $w4 = "http://5.206.225.104/dll/vcruntime140.dll" wide
        $w5 = "http://5.206.225.104/dll/freebl3.dll" wide
```

```
        $w6 = "http://5.206.225.104/dll/nss3.dll" wide

        $w7                                                    =
"C:\\Users\\louis\\Documents\\workspace\\MortyCrypter\\MsgBox.exe"
wide

        $w8 = "\\Google\\Chrome\\User Data\\Default\\Login Data" wide

        $w9 = "profiles.ini" wide

        $e1 = "hostname"

        $e2 = "encryptedUsername"

        $e3 = "encryptedPassword"

        $v1 = "vaultcli.dll"

        $v2 = "VaultOpenVault"

        $v3 = "VaultCloseVault"

        $v4 = "VaultEnumerateItems"

        $v5 = "VaultGetItem"

        $v6 = "VaultFree"

    condition:

        ((5 of ($str*)) or (4 of ($w*))) or ((all of ($e*)) and (all of ($v*)))

}
```

MITRE ATTACK TABLE

Reconnaissance	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	C&C	Exfiltration
Gather Victim Network Information (T1590)	Command and Scripting Interpreter (T1059)	Create Account (T1136)	Process Injection (T1055)	Deobfuscate/Decode Files or Information (T1140)	OS Credential Dumping (T1003.008)	Application Layer Protocol (T1071.004)	Exfiltration Over C2 Channel (T1041)
Gather Victim Host Information (T1592)	Shared Modules (T1129)	Boot or Logon Autostart Execution (T1547)	Create or Modify System Process (T1543.003)	Masquerading (T1036)			
	Native API (T1106)						

Recommendations

1. Do not download files from unknown sources.
2. Do not click on links from unknown sources.
3. Be cautious when using unknown external devices.
4. Be technology literate.
5. Keep the operating system updated.
6. Do not open untrusted emails.

PREPARED BY

Barış TURAL	https://www.linkedin.com/in/baristural/
Betül ŞAHİN	https://www.linkedin.com/in/betulsahinn/
Zeynep ÖZDEMİR	https://www.linkedin.com/in/zeynep-ozdemir/