

# REMCOS RAT

Technical Analysis Report



## Contents

Introduction.....	2
Summary .....	3
remcos.exe Analyses .....	3
eqvkptgalnkxk7m.dll Analyses .....	5
Alt Süreç: remcos.exe Analyses.....	8
Network Analyses.....	12
Behaviour Graph.....	12
Created Files .....	13
8flrd9lwsu1n.....	13
8gkf4fmyfcah3 .....	13
Remcos.exe Yara Rule .....	15
eqvkptgalnkxk7m.dll Yara Rule .....	16

## Introduction

Marketed by Germany-based firm Breaking Security as legitimate software for remotely administering Windows systems, Remcos or Remote Control and Surveillance is now widely used by threat actors in numerous malicious campaigns. Remcos is an advanced remote access Trojan (RAT) that can be used to fully control and monitor any Windows computer from XP and later.

Information generally collected and sent to servers:

- Computer Information (OS version, computer name, system type, product name, gateway)
- User information (user access, user profile, username, user domain)
- Processor information (processor revision number, processor level, processor identifier, processor architecture)

#### General Behaviors

- Antivirus products are bypassed
- Provides persistence on the targeted machine
- Runs as legitimate process by injecting it into windows process
- Gains administrator privileges and disables user account control (UAC)

#### Summary

As soon as the malware runs on the computer, it creates and writes three data files, a .dll file. When the main process runs, it loads and calls eqvkptgalnkxk7m.dll. The eqvkptgalnkxk7m.dll file is analyzed and the three data files are written, and the same .exe file is run as a subprocess and the main process is closed.

The sub-process uses heaven's gate technique, so the process runs 64-bit code from time to time. It steals user information by making records and readings in the registry and creates a key for the application. The process creates a keylogger and records it in two ways, online and offline. Offline recordings are kept on the computer, while online recordings are transferred to the target server. The information transferred to the target server includes Google Chrome and FireFox cookies and databases where user passwords are kept.

#### remcos.exe Analyses

MD5 hash: bab85d677cb634a42a890266e000fd79

SHA1 hash: bcca157ab3520b1104411e86ea78f6a2efbb58ef

Sha256 hash: 0c0a9b0df586ceb12e6b76f86473a2bf2db7cb9d8101dc90217959e9d12d48b4

The screenshot shows the assembly view of the debugger. The current instruction is:

```
004010E5    push    $000000C0
004010E6    push    $00000003
004010E7    mov     eax, [rbx+4]
004010E8    mov     [rbx], eax
004010E9    mov     r15, rdx
004010EA    mov     r14, rsi
004010EB    mov     r13, rdi
004010EC    mov     r12, rcx
004010ED    mov     r11, rbx
004010EE    mov     r10, rbp
004010EF    mov     r9, rsi
004010F0    mov     r8, rdi
004010F1    mov     rax, r15
004010F2    mov     rdx, r14
004010F3    mov     rsi, r13
004010F4    mov     rdi, r12
004010F5    mov     rbp, r11
004010F6    mov     rbp, r10
004010F7    mov     rbp, r9
004010F8    mov     rbp, r8
004010F9    mov     rbp, rax
004010FA    mov     rbp, rdx
004010FB    mov     rbp, rsi
004010FC    mov     rbp, rdi
004010FD    mov     rbp, rbp
004010FE    mov     rbp, rbp
004010FF    mov     rbp, rbp
```

The stack dump window shows the following values:

Address	Value
004010E5	000000C0
004010E6	00000003
004010E7	00000000
004010E8	00000000
004010E9	00000000
004010EA	00000000
004010EB	00000000
004010EC	00000000
004010ED	00000000
004010EF	00000000
004010F0	00000000
004010F1	00000000
004010F2	00000000
004010F3	00000000
004010F4	00000000
004010F5	00000000
004010F6	00000000
004010F7	00000000
004010F8	00000000
004010F9	00000000
004010FA	00000000
004010FB	00000000
004010FC	00000000
004010FD	00000000
004010FE	00000000
004010FF	00000000

Creates a .tmp file named Random

It creates 8gkf4fmyfcah3 and 8flrd9lwsu1n.

Writes the contents of the random .temp file.

```
[004080F0] remcos.exe!$0D07=$167  
0$0D07 remcos.exe!$0D07->kernel32.CreateFileA  
  
[004080F0] remcos.exe!$0D07->kernel32.CreateFileA
```

Creates C:\Users\zorro\AppData\Local\Temp\nsu6D84.tmp\eqvkptgalnkxk7m.dll

```

    push remcos_404000
    push remcos_40B860
    push remcos_425000
    push remcos_40213A
    push dword ptr ss:[ebp+8]
    call es1
    add esp,14
    jmp remcos_40213A
    push dword ptr ss:[ebp+8]
    push remcos_40534F
    push remcos_40534F
    call remcos_40534F
    cmp dword ptr ss:[ebp-1C],ebx
    .neq .parsec_parse2

```

FPU Gzile

EAX	70901000	<eqvkptgalnkxk7m.Hkcoedc1xfkckd1>
ECX	00000000	eqvkptgalnkxk7m.70900000
EDX	00000000	eqvkptgalnkxk7m.70900000
EBP	0018FD80	&C:\users\zorro\AppData\Local\Temp\
ESP	0018FD7C	
EDI	70901000	<eqvkptgalnkxk7m.Hkcoedc1xfkckd1>
EDF	70901000	eqvkptgalnkxk7m.70900000

Loads and calls the export function of the eqvkptgalnkxk7m.dll file. When the eqvkptgalnkxk7m.dll file finishes its work, the process closes.

## eqvkptgalnkxk7m.dll Analyses

MD5: 8BC76CEF2080A1A62F8DF3A4E0CC8014

SHA1: 80B5675C93067E88B3F984813909092128303479

SHA-256: 84167EE64B337266442B7CB230DDF207203C7E53074CEF10F9755D50348B66B7

```

    push ebp
    mov ebp,esp
    sub esp,228
    mov dword ptr ss:[ebp-228],0
    mov dword ptr ss:[ebp-224],0
    mov dword ptr ss:[ebp-220],0
    mov dword ptr ss:[ebp-21C],0
    call dword ptr ds:[&IsDebuggerPresent]
    test eax,eax
    je eqvkptgalnkxk7m.70901037
    call dword ptr ds:[&DebugBreak]
    lea eax,dword ptr ss:[ebp-218]
    push eax

```

Checks if it is in debug with IsDebuggerPresnt.

```

    push 300
    mov ecx,dword ptr ss:[ebp-4]
    push ecx
    push 0
    call dword ptr ds:[&VirtualAlloc]
    mov dword ptr ss:[ebp+8],eax
    push 0
    lea edx,dword ptr ss:[ebp-224]
    push edx

```

Varsayılan (stdcall)

1:	[esp]	00000000
2:	[esp+4]	00001A05
3:	[esp+8]	00003000
4:	[esp+C]	00000040
5:	[esp+10]	00000000

Allocates virtual memory for the 8gkf4fmymfcah3 and 8flrd9lwsu1n file. It uses the AV bypass technique by loading the Kernel32.dll/Ntdll.dll files with the parsing algorithm (Figure 1) with the writing algorithm it uses (Figure 2).

Figure 1

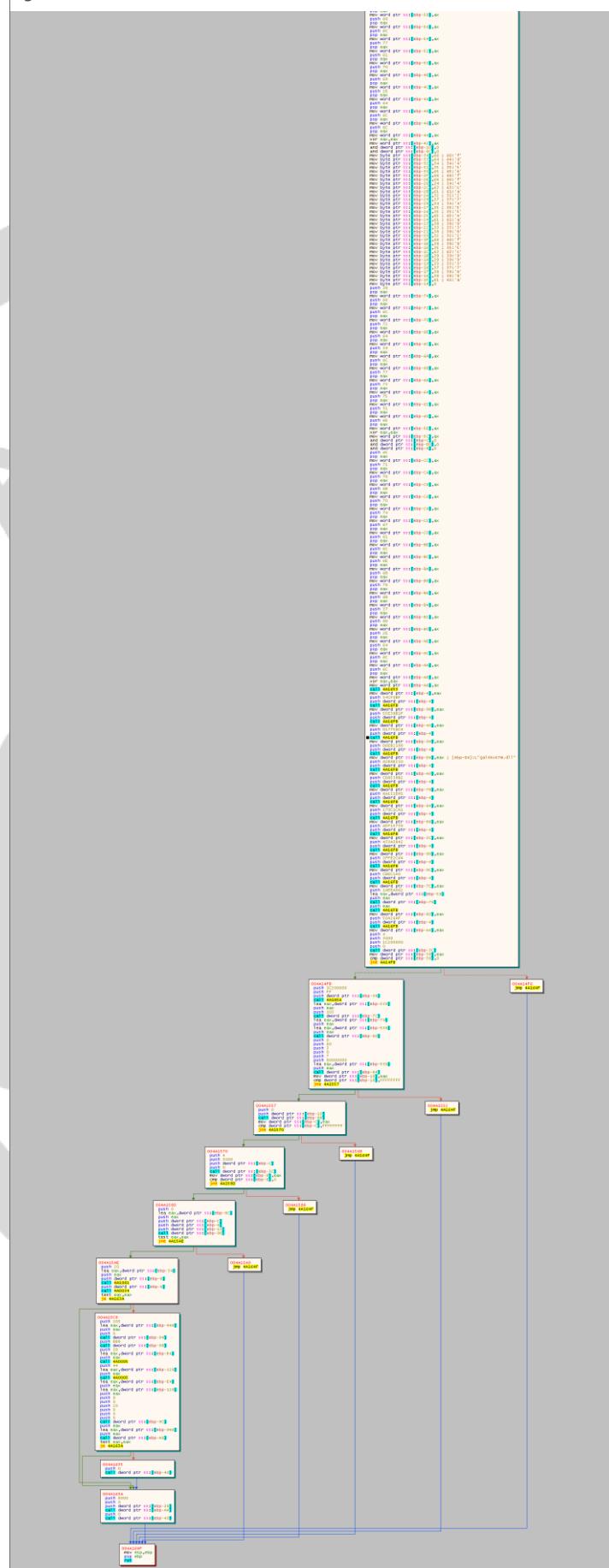
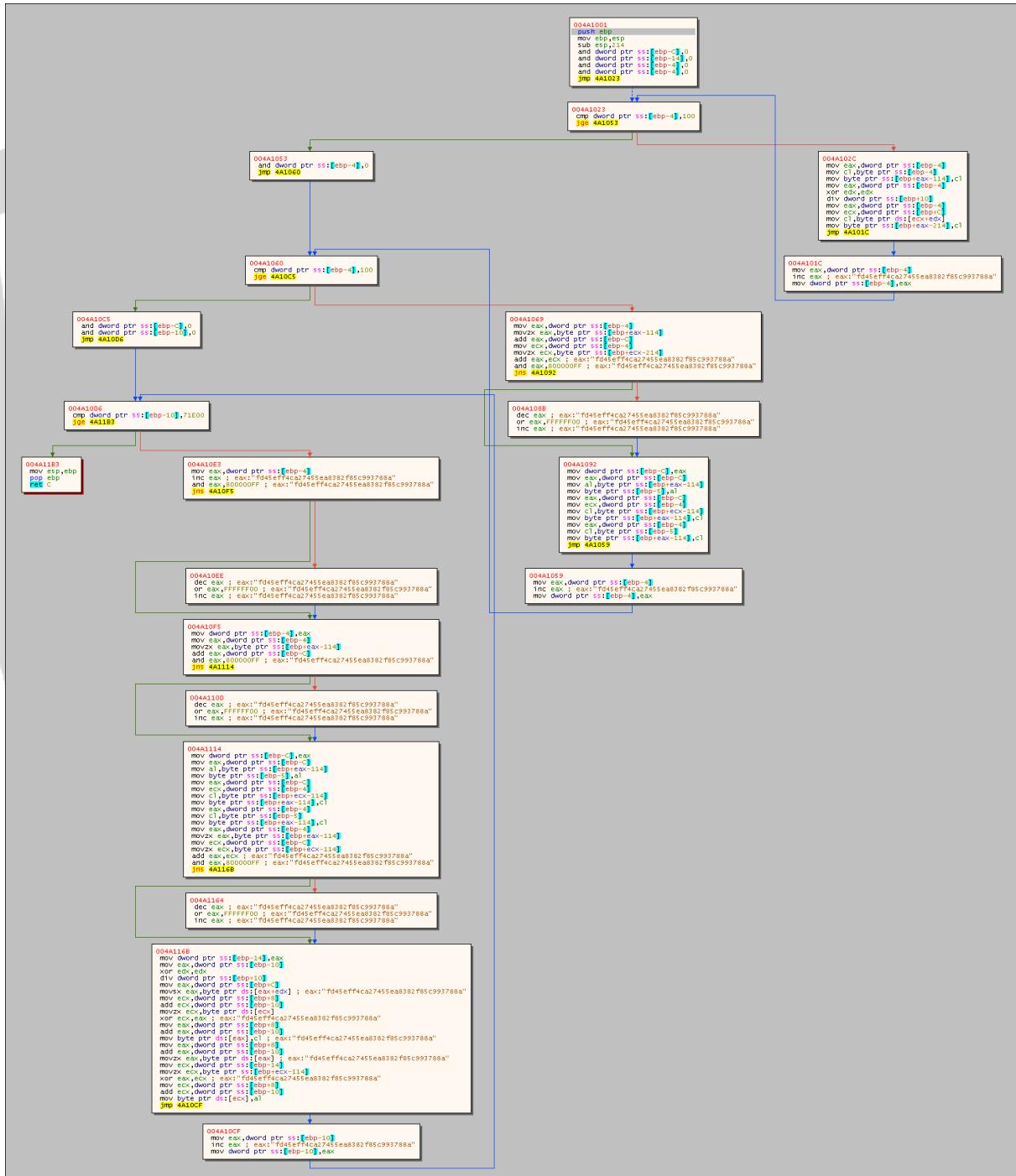


Figure 2



After these processes, the application starts a suspend process under itself.

```
lea eax,dword ptr ss:[ebp+8]
push eax
push dword ptr ss:[ebp+8]
push dword ptr ds:[eax+4]
mov es1,es0
test es1,es0
jne .L1
push eax
call kernelbase.770EF302
push eax
call kernelbase.77117B82
push eax
call kernelbase.770EF30B
mov ecx,dword ptr ss:[ebp+8]
xor eax,ecx
```

Queries the system architecture after the suspend process started

After the child process runs, the main exe closes.

## Alt Süreç: remcos.exe Analyses

The remcos.exe run as a subprocess used heaven's gate technique with hooking in <ntdll.ZWContinue>.

```
Command
-wow64!Wow64LdrpInitialize+0x111:
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\SYSTEM32\wow64cpu.dll -
00000000'736ee385 ff59d5bf      call    qword ptr [wow64+0xle8e (00000000'736ee1e8)] ds:00000000'736ee1e8([wow64cpu!CpuNotifyDllLoad (00000000'74cd1a24)])
wow64cpu!CpuNotifyDllLoad:
00000000'74cd1a24 c20000      ret     0
0:001> p
```

Loads the x64 system libraries.

```
kernel32!IsProcessorFeaturePresent:  
755a5135 8bf1    mov    edi,edi  
755a5137 55    push   ebp  
755a5138 b8ec  mov    ebp,esp  
755a513a 8b4508  mov    eax,dword ptr [ebp+8]  
755a513d 03f840  cmp    eax,40h  
755a5141 730a    jne    kernel32!IsProcessorFeaturePresent+0x18 (755a514d)  
755a5142 0fb6807402fe7f  movzx  eax,byte ptr SharedUserData+0x274 (7ffe0274)[eax]  
755a5149 5d    pop    ebp  
755a514a c20400  ret    4  
755a514d 33c0  xor    eax,eax  
755a514e 8b18    jmp    kernel32!IsProcessorFeaturePresent+0x14 (755a5149)  
755a5151 90    nop  
755a5152 90    nop  
755a5153 90    nop  
755a5154 90    nop  
  
command  
755a513a 8b4508  mov    eax,dword ptr [ebp+8] ss:002b:0018ff10=0000000a  
0:000> x86> p  
kernel32!IsProcessorFeaturePresent+0x8:  
755a513d 03f840  cmp    eax,40h  
0:000>!!!  
0:000>x86>
```

Tried to detect debugging with IsDebuggerFeaturePresent api.

0040c74c a3d094600 mov dword ptr [image00000000\_00400000+0x699d0] (004699d0)].eax  
0040c751 be70b54600 mov esi,offset image00000000\_00400000+0xc6b570 (0046b570)  
0040c756 397c2414 cmp dword ptr [esp+14h].edi  
0040c758 7554 jne image00000000\_00400000+0xc7b4 (0040c7b4)  
0040c75a 41545450 push 2EH  
0040c75e 8bc8 mov eax,esi  
0040c760 e8c45fffff call image00000000\_00400000+0xe1e29 (00401e29)  
0040c765 8bc1 mov ecx,esi  
0040c767 e80958ffff call image00000000\_00400000+0x1f75 (00401f75)  
0040c76f 41545450 push 2EH  
0040c770 4443 je image00000000\_00400000+0xc7b4 (0040c7b4)  
0040c771 393d24ad4600 cmp dword ptr [image00000000\_00400000+0x6ad24] (0046ad24).edi  
0040c779 7430 je image00000000\_00400000+0xc7b4 (0040c7b4)  
0040c779 393dd0994600 cmp dword ptr [image00000000\_00400000+0x699d0] (004699d0)].edi  
0040c77f 741f je image00000000\_00400000+0xc794 (0040c794)  
0040c780 51545450 push esi  
0040c782 41545450 push ecx  
0040c783 e88fe1ffff call image00000000\_00400000+0xa917 (0040a917)  
0040c788 83f8ff cmp eax,0FFFFFFFh  
0040c78b 7527 jne image00000000\_00400000+0xc7b4 (0040c7b4)  
0040c78d e89298ffff call image00000000\_00400000+0x024 (00406024)  
0040c792 eb20 jmp image00000000\_00400000+0xc7b4 (0040c7b4)  
0040c794 680c74500 push offset image00000000\_00400000+0x5e70c (0045e70c)

Command

0040c748 7407 je image00000000\_00400000+0xc751 (0040c751) [br=0]  
0:000:x86:p image00000000\_00400000+0xc74a:  
0040c74a ffd0 call eax,[SHELL32!IsUserAnAdmin (762c43e5)]

Registers

Reg	Value
gs	2b
fs	53
es	2b
ds	2b
edi	0
esi	46b5ac
ebx	1
eax	74467700

Processes and Threads

- 000:690 image00000000\_00400000
  - 000:d24

Calls

Raw args	Func info	Source	Addrs	Headings	Nonvolatile regs	Frame
Source args						
More Less						

WARNING: Stack unwind information not available.

Queries whether the user is an administrator or not.

The screenshot shows the Immunity Debugger interface with three main panes: Memory, Disassembly, and Command. The Memory pane at the top displays a dump of memory starting at address 00000000:0023faa8, showing assembly instructions and their corresponding hex and ASCII values. The Disassembly pane below it shows the assembly code for the current function, including calls to ADVAPI32!RegCreateKeyA. The Command pane at the bottom contains the assembly code for the function, with labels like .L76f3cc69\_8bf1 and .L76f3cc69\_8bf2.

Creates key Software\Remcos-9LK17Y

Retrieves the product name and information.

Creates exepath and license record for HKU Remcos key.

The screenshot shows the Immunity Debugger interface with several windows open:

- Memory**: Displays memory dump in hex, ASCII, and Decimal formats. The dump shows assembly code for the `kernel32!CreateFileW+0x0e` function.
- Disassembly**: Shows the assembly code for the same function, with the instruction at offset 0x004082 ff1550055a75 highlighted.
- Command**: Shows the command line input and output for the debugger.
- Registers**: Shows the CPU registers (eax, ebx, etc.) and their values.
- Processes and Threads**: Shows a list of processes and threads.
- Calls**: Shows the call stack.

The assembly pane highlights the instruction at offset 0x004082 ff1550055a75, which is a `call dword ptr [kernel32+0x10550]` instruction. The command pane shows the assembly command for this instruction: `call dword ptr [kernel32+0x10550] ds:002b:755a0550={ntdll_773e0000!RtlInit`.

Creates a log.dat file and pulls what will be written to the file.

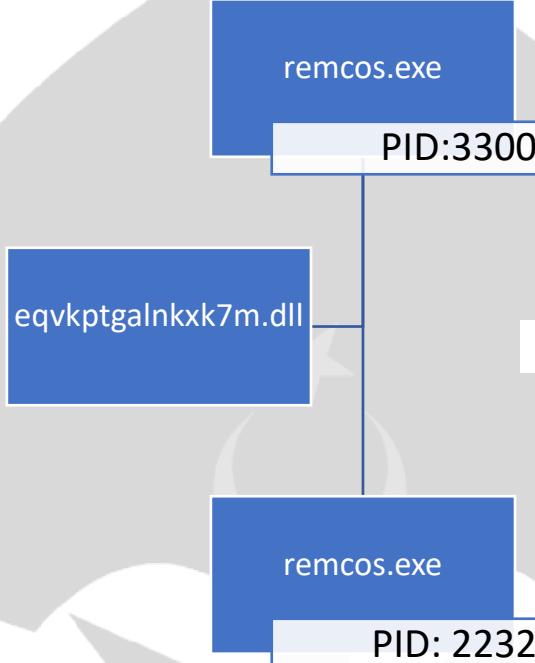
With syscall, it makes a system call to the NtOpenSection api and creates the log.dat file.

## Network Analyses

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.4	23.19.227.243	TCP	66	49736 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	2.998975	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49736 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	8.999669	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49736 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	22.033375	192.168.2.4	23.19.227.243	TCP	66	49751 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	25.047821	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49751 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	31.048259	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49751 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	44.081608	192.168.2.4	23.19.227.243	TCP	66	49758 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
8	47.088906	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49758 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	53.081751	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49758 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10	66.114794	192.168.2.4	23.19.227.243	TCP	66	49768 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	69.129576	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49768 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	75.130104	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49768 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
13	88.149004	192.168.2.4	23.19.227.243	TCP	66	49769 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
14	91.147146	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49769 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
15	97.147725	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49769 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
16	110.166344	192.168.2.4	23.19.227.243	TCP	66	49772 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
17	113.180383	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49772 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18	119.188890	192.168.2.4	23.19.227.243	TCP	66	[TCP Retransmission] 49772 → 2404 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

It sends a TCP request to the specified address (23.19.227.243), but there is no active server yet.

## Behaviour Graph



## Created Files

1. 8flrd9lwsu1n

2. 8gkf4fmyfcah3

8flrd9lwsu1n

MD5: A114322491F0917F2DC0BEE0A10E0088

SHA1: 41D03DE78CD844A167F0F4C5459FABC96FC66A2C

SHA-256: 7A90F1D6B97352E9727A9D440B0692A00F3857FF89ED406558C27655D0E9B0D3

8gkf4fmyfcah3

MD5: 6E170AC18CCCE60E6E19EB3E3E3A87A0

SHA1: B66BF0E08EC51FDA7628F9CA428B0819A3571FE6

SHA-256: EC9A222C20B4EBF39DA394D153B3825FDE3B6C70E418FC2607D287894255FC83

## Protection Methods

Up-to-date anti-virus software should be used.

The operating system should be kept up to date.

File and printer sharing services should be disabled. If these services are required, strong passwords or Active Directory authentication should be used.

Multi-factor authentication should be used.

Users' permissions to install and run unwanted software applications should be restricted. Users should not be added to the local administrators group unless necessary.

Care should be taken when opening e-mail attachments.

Unnecessary services should be disabled on agency workstations and servers.

Suspicious email attachments should be scanned or removed.

Users' web browsing habits should be monitored and access to sites with negative content should be restricted.

Care should be taken when using removable media (eg USB flash drives, external drives, CDs).

All software downloaded from the internet should be scanned before running.

Awareness of the latest threats should be maintained and appropriate access control lists should be implemented.

## Remcos.exe Yara Rule

```
import "hash"

rule REMCOS
{
    meta:
        author = "Bilal"
        first_date = "17.07.2021"
        report_date = "25.07.2021"
        description = "REMCOS"
        file_name = "remcos.exe"

    strings:
        $text_a = "nsq5333.tmp"
        $text_b = "8gkf4fmyfcah3"
        $text_c = "8flrd9lwsu1n"
        $text_d = "eqvkptgalnkxk7m.dll"
        $text_e = "Software\Remcos-9LK17Y"
        $text_licence = "06E773688BF2BDC55493602ED1695EE8"
        $text_f = "Log.dat"
        $text_ip = "23.19.227.243"

    condition:
        hash.md5(0, filesize) == "bab85d677cb634a42a890266e000fd79" or all
            text_a[0,2] and text_a[7,10]

}
```

## eqvkptgalnkxk7m.dll Yara Rule

```
import "hash"

rule REMCOS
{
    meta:
        author = "Bilal"
        first_date = "17.07.2021"
        report_date = "25.07.2021"
        description = "REMCOS"
        file_name = "eqvkptgalnkxk7m.dll"

    strings:
        $text_a = "8gkf4fmyfcah3"
        $text_b = "8flrd9lwsu1n"

    condition:
        hash.md5(0, filesize) == "8BC76CEF2080A1A62F8DF3A4E0CC8014" or all
}
```

ZAYOTE

**Prepared BY**

BİLAL BAKARTEPE

[linkedin.com/in/bilal-bakartepe](https://linkedin.com/in/bilal-bakartepe)