



DOH in Godlua Backdoor

Godlua Botnet

On April 24, 2019, our Threat Detection System highlighted a suspicious ELF which is a Lua-based Backdoor, we named it Godlua Backdoor as the Lua byte-code file loaded by this sample has a magic number of "God".

Common practice to deal with botnets

kill the C2s – Bots become headless

Godlua Backdoor is one of the most sophisticated botnet we have seen

A redundant communication mechanism for C2 connection,

hardcoded c2 name,

getting c2 from Pastebin.com,

getting c2 from GitHub.com

DNS TXT

DNS over HTTPS

Reference:

[1] <https://blog.netlab.360.com/an-analysis-of-godlua-backdoor-en/>

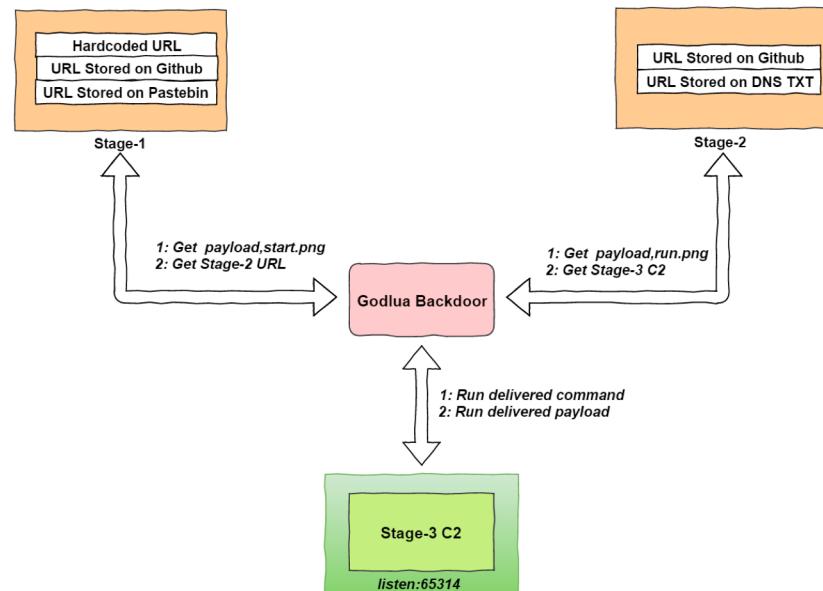
Godlua C2 redundant mechanism

In order to get the real C2 address, Godlua goes through 3 steps

Step1: retrieve a “start.png” from either a hardcoded ciphertext, Github project description, or Pastebin text. Decrypts the “start.png” and executes it to get a stage-2 URL

Step2: retrieve a “run.png” from either a Github project file or DoH. Decrypts the “run.png” and executes it to get a C2 domain

Step3: Ask Cloudflare DoH API to get the corresponding C2 address



Godlua Stage-1

The backdoor uses 3 different ways to store the Stage-1 URL. hardcoded ciphertext, Github project description, and Pastebin text.

For example - request <https://pastebin.com/raw/vSDzq3Md>, and get the following content
`/tbLY0TsMUnC+iO9aYm9yS2eayKIKLQyFPOaNxSCnZpBw4RLGnJOPcZXHaf/aoj`

After decryption a Stage-1 URL shows up:
<https://img2.cloudappconfig.com/%s.png>

Adding a “start” string, the url changes to <https://img2.cloudappconfig.com/start.png> (actually a lua code) , Godlua downloads and decrypts it , within it , a new domain t.cloudappconfig.com comes up, Godlua runs this lua code to go into Stage-2

DOH in Godlua Stage-2

Here at stage-2, Godlua retrieves a run.png from either a Github project file or DoH. Decrypts the run.png and executes it to get a C2 domain

For example - with the previous domain t.cloudappconfig.com in hand, a DNS TXT DoH request sent to cloudflare DoH api (`name=t.cloudappconfig.com&type=TXT`) and an answer img1.cloudappconfig.com comes up

```
└$ curl -H 'accept: application/dns-json' -i "https://cloudflare-dns.com/dns-query?name=t.cloudappconfig.com&type=TXT"
HTTP/2 200
date: Wed, 04 Mar 2020 03:19:45 GMT
content-type: application/dns-json
content-length: 345
access-control-allow-origin: *
cache-control: max-age=300
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
server: cloudflare
cf-ray: 56e8751c9a54ebc9-LAX

{"Status": 0, "TC": false, "RD": true, "RA": true, "AD": false, "CD": false, "Question": [{"name": "t.cloudappconfig.com.", "type": 16}], "Answer": [{"name": "t.cloudappconfig.com.", "type": 16, "TTL": 300, "data": "\\"6TmRMwDw5R/sNSEhjCByEw0Vb44nZhEUyUpUR4LcijflukjdAv+vqqMuYOFaOOpC7Ktyyr6nUOqO9XnDpudVmbGoTeJD6hYrw72YmiOS9dX5M/sPNmsw/eY/XDYYzx5\\\""}]}
```

TXT record (Base64 + AES)

6TmRMwDw5R/sNSEhjCByEw0Vb44nZhEUyUpUR4LcijflukjdAv+vqqMuYOFaOOpC7Ktyyr6nUOqO9XnDpudVmbGoTeJD6hYrw72YmiOS9dX5M/sPNmsw/eY/XDYYzx5/

After decrypt the TXT record (C2 config plaintext)

```
{"u":"http:\/\/img1.cloudappconfig.com\/%s.png","c":"img1.cloudappconfig.com::43.224.225.220:"}
```

Adding a “run” string, the url changes to <https://img1.cloudappconfig.com/run.png> (actually a lua code) , within it , a C2 domain c.cloudappconfig.com shows up, Godlua downloads, decrypts and runs this lua code to go into Stage-3

DOH in Godlua Stage-3

Godlua requests the cloudflare DOH API to get the DNS A record for the c2 domain c.cloudappconfig.com.

For example - Godlua DOH request (*name=c.cloudappconfig.com&type=A*)

```
└$ curl -H 'accept: application/dns-json' -i "https://cloudflare-dns.com/dns-query?name=c.cloudappconfig.com&type=A"
HTTP/2 200
date: Wed, 04 Mar 2020 03:20:10 GMT
content-type: application/dns-json
content-length: 225
access-control-allow-origin: *
cache-control: max-age=300
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
server: cloudflare
cf-ray: 56e875b8e92898cf-LAX

{"Status": 0, "TC": false, "RD": true, "RA": true, "AD": false, "CD": false, "Question": [{"name": "c.cloudappconfig.com.", "type": 1}], "Answer": [{"name": "c.cloudappconfig.com.", "type": 1, "TTL": 300, "data": "43.224.225.220"}]}
```

The result indicates c.cloudappconfig.com has a A record 43.224.225.220 , with this final A record, Bots of the Godlua now can communicate with the C2.

Godlua's DoH request

The “get_dns_record” function used to request the Cloudflare DOH API.

```

0 [-]: TEST      R1 1          ; if not R1 then goto 2 else goto 3
1 [-]: JMP       R0 1          ; PC += 1 (goto 3)
2 [-]: LOADK    R1 K0          ; R1 := "A"
3 [-]: GETTABUP  R2 U0 K1      ; R2 := U0["request"]
4 [-]: NEWTABLE  R3 0 2        ; R3 := {} (size = 0,2)
5 [-]: GETUPVAL  R4 U1        ; R4 := U1
6 [-]: LOADK    R5 K3          ; R5 := "https://cloudflare-dns.com/dns-query?name=%s&type=%s"
7 [-]: MOVE     R6 R0          ; R6 := R0
8 [-]: MOVE     R7 R1          ; R7 := R1
9 [-]: CALL     R4 4 2        ; R4 := R4(R5 to R7)
10 [-]: SETTABLE R3 K2 R4      ; R3["url"] := R4
11 [-]: NEWTABLE R4 1 0        ; R4 := {} (size = 1,0)
12 [-]: LOADK    R5 K5          ; R5 := "Accept: application/dns-json"
13 [-]: SETLIST   R4 1 1        ; R4[0] := R5 ; R(a)[(c-1)*FPP+1] := R(a+i), 1 <= i <= b, a=4, b=1, c=1, FPP=50
14 [-]: SETTABLE R3 K4 R4      ; R3["httpheader"] := R4
15 [-]: LOADNIL   R4 0          ; R4 := nil
16 [-]: LOADBOOL  R5 1 0        ; R5 := true
17 [-]: CALL     R2 4 2        ; R2 := R2(R3 to R5)
18 [-]: TEST     R2 0          ; if R2 then goto 20 else goto 23
19 [-]: JMP       R0 3          ; PC += 3 (goto 23)
20 [-]: LEN      R3 R2          ; R3 := #R2
21 [-]: EQ       0 R3 K6          ; if R3 == 0 then goto 23 else goto 24
22 [-]: JMP       R0 1          ; PC += 1 (goto 24)
23 [-]: RETURN   R0 1          ; return
24 [-]: GETTABUP R3 U2 K7      ; R3 := U2["require"]
25 [-]: LOADK    R4 K8          ; R4 := " cJSON_safe"
26 [-]: CALL     R3 2 2        ; R3 := R3(R4)
27 [-]: GETTABLE R4 R3 K9      ; R4 := R3["decode"]
28 [-]: MOVE     R5 R2          ; R5 := R2
29 [-]: CALL     R4 2 3        ; R4 to R5 := R4(R5)
30 [-]: TEST     R5 1          ; if not R5 then goto 32 else goto 37
31 [-]: JMP       R0 5          ; PC += 5 (goto 37)
32 [-]: TEST     R4 0          ; if R4 then goto 34 else goto 37
33 [-]: JMP       R0 3          ; PC += 3 (goto 37)
34 [-]: GETTABLE R6 R4 K10      ; R6 := R4["Answer"]
35 [-]: TEST     R6 1          ; if not R6 then goto 37 else goto 38
36 [-]: JMP       R0 1          ; PC += 1 (goto 38)

```

Reference:

[1] <https://developers.cloudflare.com/1.1.1.1/dns-over-https/json-format/>

Take away

1. Godlua uses DoH to evade c2 detections. “Local” network no more has visibility into the C2 domains
2. Godlua is the first botnet we have observed to utilize DoH. It is safe to assume Godlua will not be the last one given the fact the DoH is picking up.
3. DNS monitoring|blocking as a cost effective, easy-scaled security protecting mechanism has been used widely for long time.
4. DoH will essentially renders the above defending mechanism useless.



Q&A