密级	公开

华硕路由器 9999 端口远程命令执行研究报告 V1

[第一版 2015/01/12 下午]

知道创宇安全研究团队

1. 更新情况

版本	时间	描述
第一版	2015/1/12 下午	第一版完成。

2. 漏洞概要

2014 年 10 月 3 日，国外安全研究员 Joshua J. Drake 在他 github（https://github.com/jduck）提交了针对华硕路由器的一个远程命令执行漏洞 poc（https://github.com/jduck/asus-cmd）。该漏洞随后被编号为 CVE-2014-9583。

知道创宇安全研究团队在第一时间对该命令执行漏洞进行了研究和分析。

a) 漏洞描述

华硕路由器 R 系列路由器使用开源路由器系统 Asuswrt (https://github.com/RMerl/asuswrt-merlin)，开源代码给我们随后的漏洞分析带来很多方便，不用逆向分析。在 Asuswrt 中存在 infosvr（https://github.com/RMerl/asuswrt-merlin/tree/master/release/src/router/infosvr）进程，该进程监听在 0.0.0.0 IP 上，监听本机任何 IP 的 9999 UDP 端口。Infosvr 自身的授权机制不完整，在 infosvr 处理用户提交的数据时也没有适合的过滤，而且使用了 system()函数执行部分请求，最终导致远程命令执行漏洞。

b) 漏洞影响

据 Joshua J. Drake 在 github 上的分析，受影响的版本如下：

Router	**Firmware Version**	**Verified By**
RT-AC66U	3.0.0.376.2524-g0013f52	Joshua J. Drake
RT-N66U	3.0.0.4.376_1071-g8696125	Friedrich Postelstorfer
RT-AC87U	3.0.0.4.378_3754	David Longenecker
RT-N56U	3.0.0.4.374_5656	@argilo

不过，开源路由器系统 Asuswrt 项目支持如下所有路由器硬件型号，所以建议如下型号路由器用户检测是否存在漏洞：

- RT-N16
- RT-AC56U
- RT-N66U
- RT-AC66U
- RT-AC68U
- RT-AC68P
- RT-AC87U

c) 漏洞分析

代码文件：

https://github.com/RMerl/asuswrt-merlin/blob/34b5933112d7164b68add63fee63f007a0569309/release/src/router/infosvr/infosvr.c

在代码 162 行处，infosvr 绑定到了 0.0.0.0 IP 的 9999 UDP 端口上，如图：

```
162         bzero((char *)&serv_addr , sizeof(serv_addr));
163         serv_addr.sin_family            = AF_INET;
164         serv_addr.sin_addr.s_addr       = htonl(INADDR_ANY);
165         serv_addr.sin_port              = htons(SRV_PORT);
166
167         int flag=1;
168         setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (char *)&flag, sizeof(flag));
169         if (bind(sockfd,(struct sockaddr *)&serv_addr , sizeof(serv_addr)) < 0)
170         {
171                 PRINT("can't bind\n");
172                 perror("ERR:");
173                 exit(0);
174         }
175
```

在代码 186 行处，infosvr 对传入的请求交给 processReq()函数处理，如图：

```
176         while (TRUE)
177         {
178                 int     res;
179                 fd_set  fdvar;
180
181                 FD_ZERO(&fdvar);
182                 FD_SET(sockfd, &fdvar);
183
184                 tv.tv_sec  = 2;
185                 tv.tv_usec = 0;
186                 res = select( sockfd + 1 , &fdvar , NULL , NULL , &tv);
187
188                 if (res == 1)
189                 {
190 //                      printf("got packet\n");
191                         processReq(sockfd);
192                 }
193         }
194 }
```

ProcessReq()函数的功能就是接收 512 字节的请求数据，并在代码 227 行处把数据交给 processPacket()函数处理，如图：

```
196 int processReq(int sockfd)
197 {
198 //    IBOX_COMM_PKT_HDR*  phdr;
199     int          /*iLen , iRes , iCount , */iRcv;
200     int          fromlen;
201     char                *hdr;
202     char                pdubuf[INFO_PDU_LENGTH];
203     struct sockaddr_in  from_addr;
204
205     memset(pdubuf,0,sizeof(pdubuf));
206
207     //PRINT("Enter processReq\n");
208
209     if ( waitsock(sockfd , 1 , 0) == 0)
210     {
211         return -1;
212     }
213
214     fromlen = sizeof(from_addr);
215
216     //Receive the complete PDU
217     iRcv = RECV(sockfd , pdubuf , INFO_PDU_LENGTH , (struct sockaddr *)&from_addr , &fromlen  , 1);
218
219     if (iRcv != INFO_PDU_LENGTH)
220     {
221         closesocket(sockfd);
222         return (-1);
223     }
224
225     hdr = pdubuf;
226
227     processPacket(sockfd, hdr);
228 /*                                          J++
229     closesocket(sockfd);
230 */
231 }
```

processPacket()函数位于文件：

https://github.com/RMerl/asuswrt-merlin/blob/34b5933112d7164b68add63fee63f007a0569309/release/src/router/infosvr/common.c

processPacket()函数在代码 202 行处把 512 字节的请求数据(pdubuf 字符指针)转换成 IBOX_COMM_PKT_HDR 结构（phdr），攻击者如果想触发漏洞，需要按照这个数据结构来发送数据包，如图：

```
177  char *processPacket(int sockfd, char *pdubuf)
178  {
179      IBOX_COMM_PKT_HDR   *phdr;
180      IBOX_COMM_PKT_HDR_EX *phdr_ex;
181      IBOX_COMM_PKT_RES_EX *phdr_res;
182      PKT_GET_INFO *ginfo;
183
184  #ifdef WAVESERVER       // eric++
185      int fail = 0;
186      pid_t pid;
187      DIR *dir;
188      int fd, ret, bytes;
189      unsigned char tmp_buf[15];  // /proc/XXXXXX
190      WS_INFO_T *wsinfo;
191  #endif
192  //#ifdef WL700G
193      STORAGE_INFO_T *st;
194  //#endif
195  //    int i;
196      char ftype[8], prinfo[128]; /* get disk type */
197      int free_space;
198  #if defined(RTCONFIG_WIRELESSREPEATER) || defined(RTCONFIG_PROXYSTA)
199      char tmp[100], prefix[] = "wlXXXXXXXXXXXXXX";
200  #endif
201
202      phdr = (IBOX_COMM_PKT_HDR *)pdubuf;
203      phdr_res = (IBOX_COMM_PKT_RES_EX *)pdubuf_res;
```

IBOX_COMM_PKT_HDR 结构在文件：

https://github.com/RMerl/asuswrt-merlin/blob/34b5933112d7164b68add63fee63f007a0569309/release/src/router/infosvr/iboxcom.h.wiresshd

IBOX_COMM_PKT_HDR 结构在代码 81 行处定义，如图：

```
74  //Packet Header Structure
75  typedef struct iboxPKT
76  {
77          BYTE            ServiceID;
78          BYTE            PacketType;
79          WORD            OpCode;
80          DWORD           Info; // Or Transaction ID
81  } IBOX_COMM_PKT_HDR;
```

想要进入触发漏洞的关键代码区域，需要通过 common.c 文件代码 207 行处的 if 判断，如图：

```
202     phdr = (IBOX_COMM_PKT_HDR *)pdubuf;
203     phdr_res = (IBOX_COMM_PKT_RES_EX *)pdubuf_res;
204
205 //    fprintf(stderr,"Get: %x %x %x\n", phdr->ServiceID, phdr->PacketType, phdr->OpCode);
206
207     if (phdr->ServiceID==NET_SERVICE_ID_IBOX_INFO &&
208         phdr->PacketType==NET_PACKET_TYPE_CMD)
209     {
```

所以，我们要设定攻击代码的前两字节分别为常量 NET_SERVICE_ID_IBOX_INFO 和 NET_PACKET_TYPE_CMD 的值，即：\x0C\x15（十进制是 12 和 21）。

```
36  //Packet Type Section
37  #define NET_SERVICE_ID_BASE     10
38  #define NET_SERVICE_ID_LPT_EMU          NET_SERVICE_ID_BASE + 1
39  #define NET_SERVICE_ID_IBOX_INFO        NET_SERVICE_ID_BASE + 2
40
41
42  //Packet Type Section
43  #define NET_PACKET_TYPE_BASE    20
44  #define NET_PACKET_TYPE_CMD     NET_PACKET_TYPE_BASE + 1
45  #defin T_PACKET_TYPE_RES        NET_PACKET_TYPE_BASE + 2
46
```

来到 common.c 文件代码 222 行处，infosvr 作者应该是想对随后能执行 system()函数代码的数据做个授权，或者验证。该处作者使用了 MAC 验证（代码 227 行处）和密码验证，不过不知为何密码验证代码被注释掉了（代码 240 行处），而针对 MAC 验证的代码也属于摆设状态，代码如图：

```
222         if (phdr->OpCode!=NET_CMD_ID_GETINFO && phdr->OpCode!=NET_CMD_ID_GETINFO_MANU)
223         {
224                 phdr_ex = (IBOX_COMM_PKT_HDR_EX *)pdubuf;
225
226                 // Check Mac Address
227                 if (memcpy(phdr_ex->MacAddress, mac, 6)==0)
228                 {
229                         _dprintf("Mac Error %2x%2x%2x%2x%2x%2x\n",
230                                 (unsigned char)phdr_ex->MacAddress[0],
231                                 (unsigned char)phdr_ex->MacAddress[1],
232                                 (unsigned char)phdr_ex->MacAddress[2],
233                                 (unsigned char)phdr_ex->MacAddress[3],
234                                 (unsigned char)phdr_ex->MacAddress[4],
235                                 (unsigned char)phdr_ex->MacAddress[5]
236                                 );
237                         return NULL;
238                 }
239
240                 // Check Password
241                 //if (strcmp(phdr_ex->Password, "admin")!=0)
242                 //{
243                 //      phdr_res->OpCode = phdr->OpCode | NET_RES_ERR_PASSWORD;
244                 //      _dprintf("Password Error %s\n", phdr_ex->Password);
245                 //      return NULL;
246                 //}
247                 phdr_res->Info = phdr_ex->Info;
248                 memcpy(phdr_res->MacAddress, phdr_ex->MacAddress, 6);
249         }
```

在验证前，需要把 512 字节数据 pdubuf 转换成 IBOX_COMM_PKT_HDR_EX 数据结构，IBOX_COMM_PKT_HDR_EX 结构包含了 MAC 字段和密码字段，如图：

```
92  typedef struct iboxPKTEx
93  {
94          BYTE            ServiceID;
95          BYTE            PacketType;
96          WORD            OpCode;
97          DWORD           Info; // Or Transaction ID
98          BYTE            MacAddress[6];
99          BYTE            Password[32];
100 } IBOX_COMM_PKT_HDR_EX;
```

为什么说 MAC 验证是摆设呢？因为在 common.c 文件代码 227 行处，代码只是把 MacAddress 的前 6 字节拷贝到了字符数组 mac 处，返回是指向 mac 地址的指针，不会等于 0，所以该验证毫无任何用处，如图：

```
227                 if (memcpy(phdr_ex->MacAddress, mac, 6)==0)
228                 {
229                         _dprintf("Mac Error %2x%2x%2x%2x%2x%2x\n",
230                                 (unsigned char)phdr_ex->MacAddress[0],
231                                 (unsigned char)phdr_ex->MacAddress[1],
232                                 (unsigned char)phdr_ex->MacAddress[2],
233                                 (unsigned char)phdr_ex->MacAddress[3],
234                                 (unsigned char)phdr_ex->MacAddress[4],
235                                 (unsigned char)phdr_ex->MacAddress[5]
236                                 );
237                         return NULL;
238                 }
```

大胆猜测作者可能是想用 memcmp()函数，结果用错了，不得而知。

继续前进，在 common.c 文件代码 251 行处进入 switch()函数判断阶段，针对不同 OpCode 执行不同的分支代码，而当 OpCode 为 NET_CMD_ID_MANU_CMD 常量值（十进制 51，十六进制 33）时，才能执行 system()函数代码，所以，我们要设定攻击代码的前四字节为 \x0C\x15\x33\x00，如图：

```
251         switch(phdr->OpCode)
252         {
253                 case NET_CMD_ID_GETINFO_MANU:
254 //              case NET_CMD_ID_GETINFO:
255                     _dprintf("NET CMD GETINFO_MANU\n");        // tmp test
256                     ginfo=(PKT_GET_INFO *)(pdubuf_res+sizeof(IBOX_COMM_PKT_RES));
257                     memset(ginfo, 0, sizeof(ginfo));
```

```
428             case NET_CMD_ID_MANU_CMD:
429             {
430                 #define MAXSYSCMD 256
431                 char cmdstr[MAXSYSCMD];
432                 PKT_SYSCMD *syscmd;
433                 PKT_SYSCMD_RES *syscmd_res;
434                 FILE *fp;
435
436 printf("1. NET_CMD_ID_MANU_CMD:\n");
437 _dprintf("2. NET_CMD_ID_MANU_CMD:\n");
438 fprintf(stderr, "3. NET_CMD_ID_MANU_CMD:\n");
439
440                 syscmd = (PKT_SYSCMD *)(pdubuf+sizeof(IBOX_COMM_PKT_HDR_EX));
441                 syscmd_res = (PKT_SYSCMD_RES *)(pdubuf_res+sizeof(IBOX_COMM_PKT_RES_EX));
442
443                 if (syscmd->len>=MAXSYSCMD) syscmd->len=MAXSYSCMD;
444                 syscmd->cmd[syscmd->len]=0;
445                 syscmd->len=strlen(syscmd->cmd);
446                 fprintf(stderr,"system cmd: %d %s\n", syscmd->len, syscmd->cmd);
```

在 common.c 文件 NET_CMD_ID_MANU_CMD 分值代码中，代码 440 行出代码把 pdubuf 减去 IBOX_COMM_PKT_HDR_EX 结构的数据，剩余部分转换成 PKT_SYSCMD 结构，作为命令执行数据，PKT_SYSCMD 结构如图：

```
113 typedef struct iboxPKTCmd
114 {
115         BYTE            len;
116         BYTE            cmd[480];
117 } IBOX_COMM_PKT_SYSCMD;
```

最终，在 common.c 文件代码 514 行处，syscmd 结构中 cmd 字段被赋值给 cmdstr，在代码 515 行处，cmdstr 作为命令被 system()函数执行，如图：

```
512  #endif
513                          {
514                              sprintf(cmdstr, "%s > /tmp/syscmd.out", syscmd->cmd);
515                              system(cmdstr);
516
517                              fprintf(stderr,"rund: %s\n", cmdstr);
518                              fp = fopen("/tmp/syscmd.out", "r");
```

d) 漏洞重现

漏洞测试脚本：

http://www.exploit-db.com/exploits/35688/

下载存在漏洞的华硕路由固件：

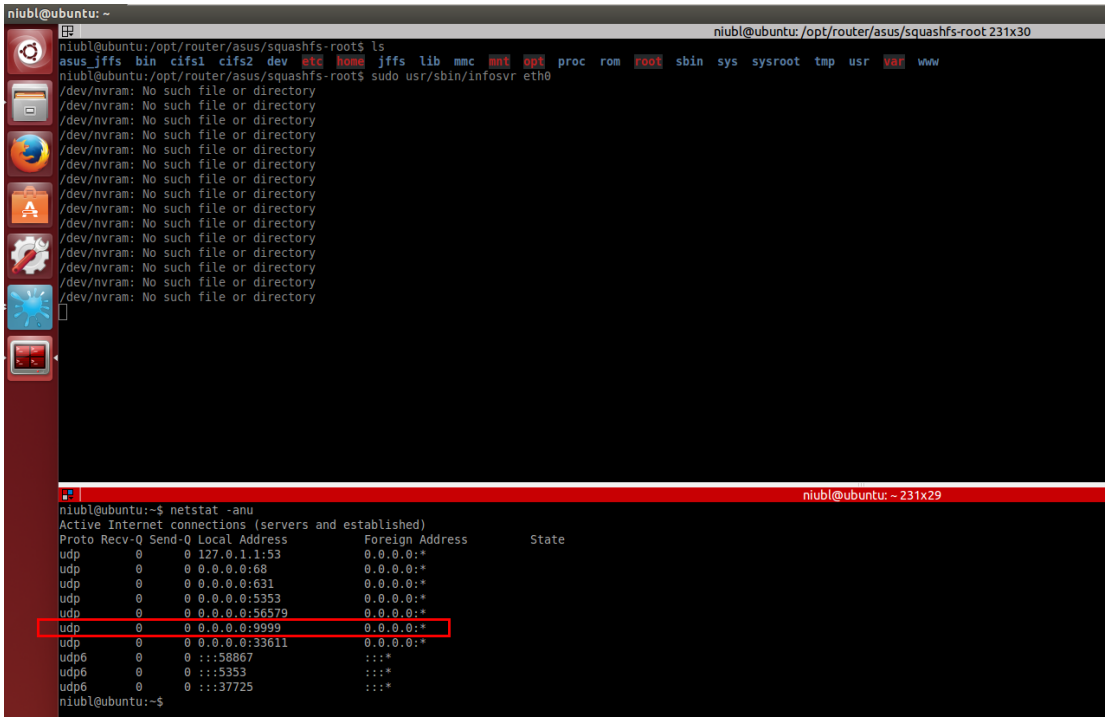
http://dlsvr04.asus.com/pub/ASUS/wireless/RT-AC66U/FW_RT_AC66U_30043763626.zip

binwalk 解压文件：

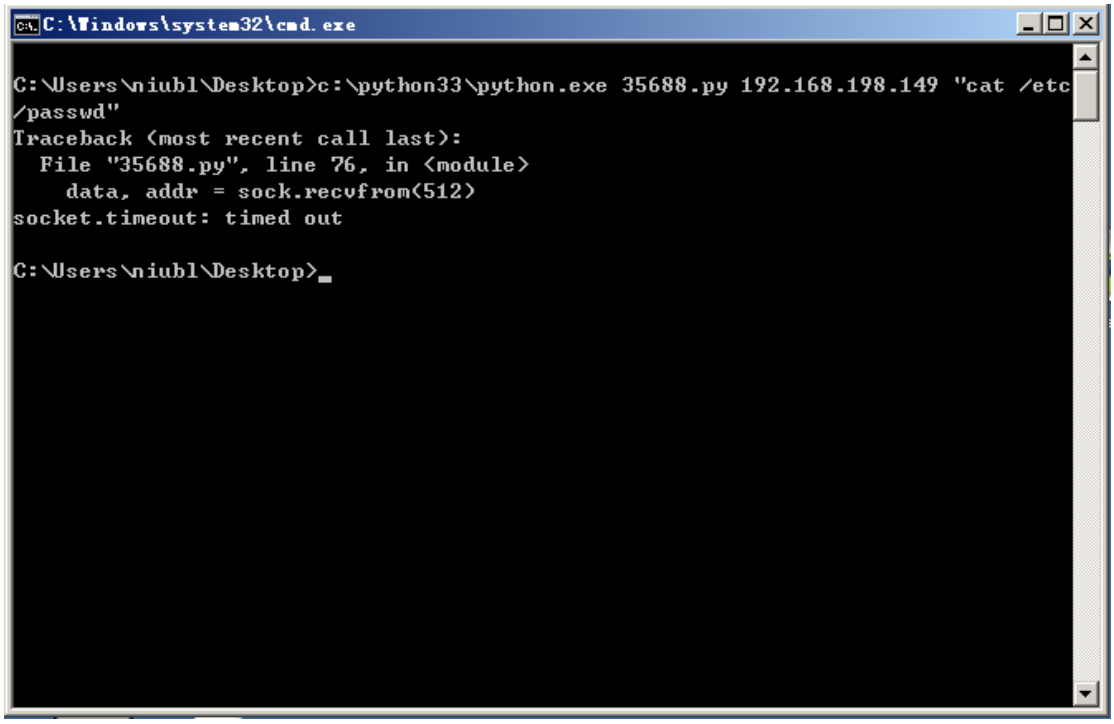
```
root@localhost:~/router/asus# ls
FW_RT_AC66U_30043763626.zip
root@localhost:~/router/asus# unzip FW_RT_AC66U_30043763626.zip
Archive:  FW_RT_AC66U_30043763626.zip
 extracting: FW_RT_AC66U_30043763626.trx
root@localhost:~/router/asus# ls
FW_RT_AC66U_30043763626.trx  FW_RT_AC66U_30043763626.zip
root@localhost:~/router/asus# binwalk -Me FW_RT_AC66U_30043763626.trx

Scan Time:     2015-01-13 12:06:58
Target File:   FW_RT_AC66U_30043763626.trx
MD5 Checksum:  e692e2339f7f268cee778f8e89a99d32
Signatures:    285

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             TRX firmware header, little endian, header size: 28 bytes, image size: 26386432 bytes, CRC32: 0xA3484294 flags: 0x0, version: 1
28            0x1C            LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size: 3392380 bytes
```

```
root@localhost:~/router/asus# ls
FW_RT_AC66U_30043763626.trx  _FW_RT_AC66U_30043763626.trx.extracted  FW_RT_AC66U_30043763626.zip
root@localhost:~/router/asus# ls _FW_RT_AC66U_30043763626.trx.extracted/
134EBC.squashfs  1C  1C.7z  _1C.extracted  squashfs-root
root@localhost:~/router/asus# CD _FW_RT_AC66U_30043763626.trx.extracted/squashfs-root/
CD: command not found
root@localhost:~/router/asus# cd _FW_RT_AC66U_30043763626.trx.extracted/squashfs-root/
root@localhost:~/router/asus/_FW_RT_AC66U_30043763626.trx.extracted/squashfs-root# ls
asus_jffs  bin  cifs1  cifs2  dev  etc  home  jffs  lib  mnt  opt  proc  rom  root  sbin  sys  sysroot  tmp  usr  var  www
root@localhost:~/router/asus/_FW_RT_AC66U_30043763626.trx.extracted/squashfs-root#
```

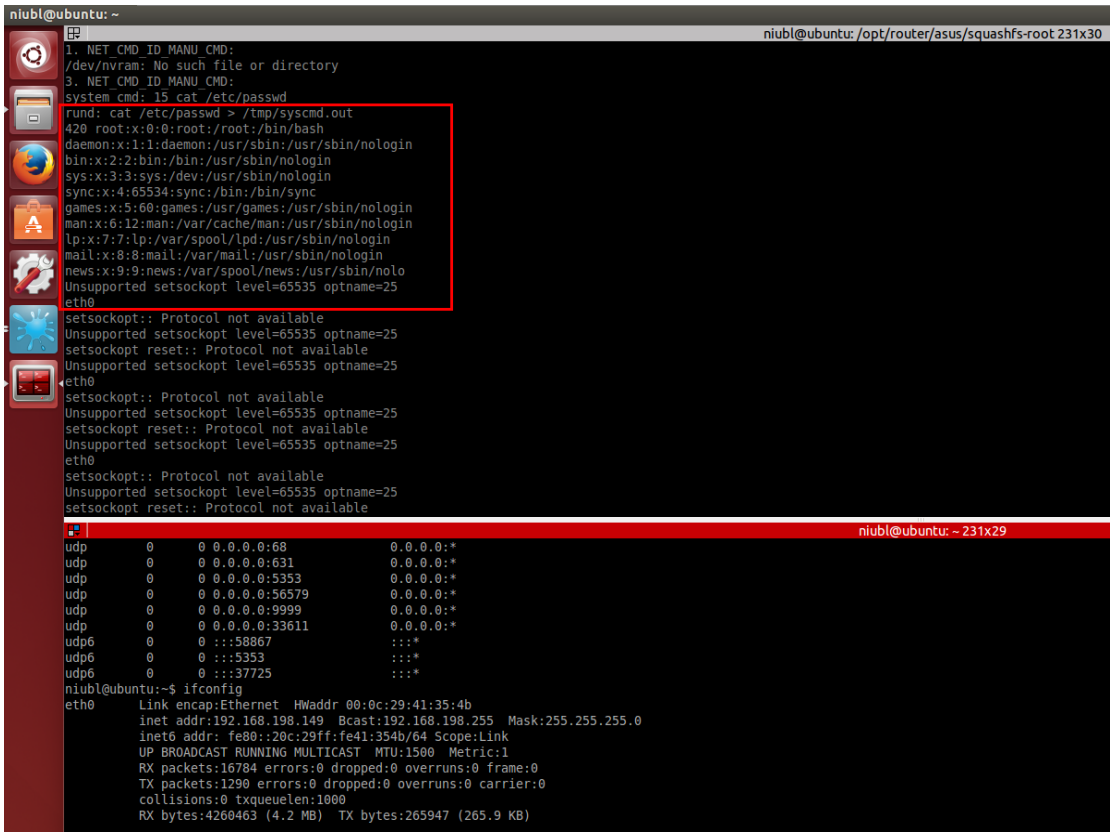
模拟运行，如图：



攻击 infosvr 程序，如图：



命令执行成功，如图：



e) 漏洞修复

开源路由器系统 Asuswrt 已经 2015 年 1 月 10 号下午修复了漏洞，修复漏洞的办法是直接注释掉了触发命令执行漏洞的关键部分，如图：

```
430  #if 0 // Vulenrable code
431                  case NET_CMD_ID_MANU_CMD:
432                  {
433                       #define MAXSYSCMD 256
434                       char cmdstr[MAXSYSCMD];
435                       PKT_SYSCMD *syscmd;
436                       PKT_SYSCMD_RES *syscmd_res;
437                       FILE *fp;
438
439  printf("1. NET_CMD_ID_MANU_CMD:\n");
440  _dprintf("2. NET_CMD_ID_MANU_CMD:\n");
441  fprintf(stderr, "3. NET_CMD_ID_MANU_CMD:\n");
442
443                       syscmd = (PKT_SYSCMD *)(pdubuf+sizeof(IBOX_COMM_PKT_HDR_EX));
444                       syscmd_res = (PKT_SYSCMD_RES *)(pdubuf_res+sizeof(IBOX_COMM_PKT_RES_EX));
445
446                       if (__le16_to_cpu(syscmd->len) >= MAXSYSCMD) syscmd->len = __cpu_to_le16(MAXSYSCMD);
447                       syscmd->cmd[__le16_to_cpu(syscmd->len)]=0;
448                       syscmd->len=__cpu_to_le16(strlen(syscmd->cmd));
449                       fprintf(stderr,"system cmd: %d %s\n", syscmd->len, syscmd->cmd);
450  #if 0
```

华硕官方也推出相应的固件升级，想要修复漏洞的用户可以去下载相关路由器型号的升级固件：

http://www.asus.com.cn/Networking/Wireless_Routers_Products/

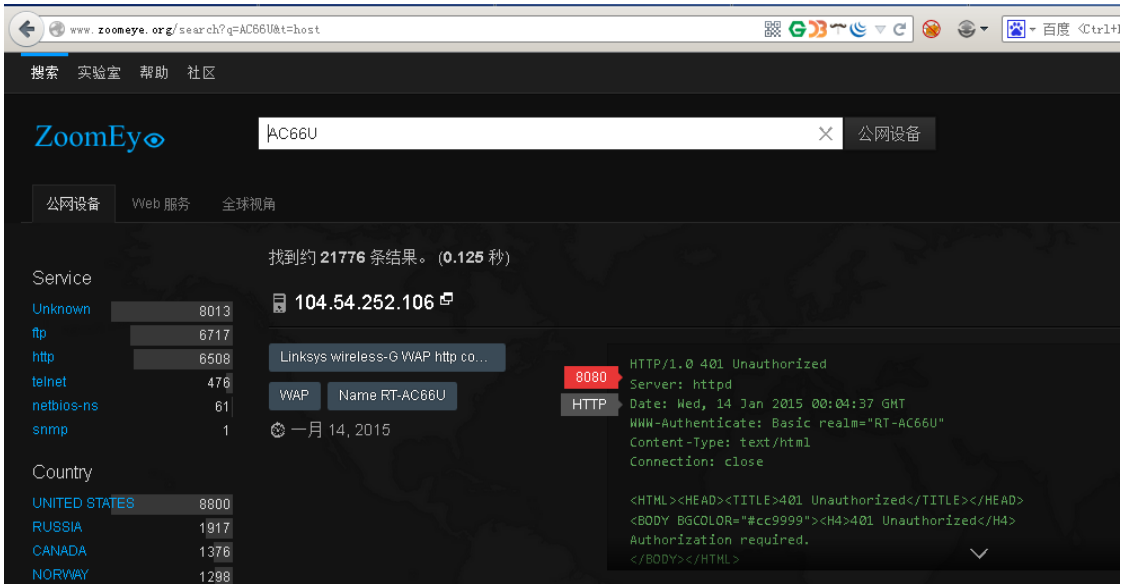
例如 RT-AC66U 型号路由器 2015 年 1 月 12 号推出的升级版本，如图：

3. ZoomEye 检测报告

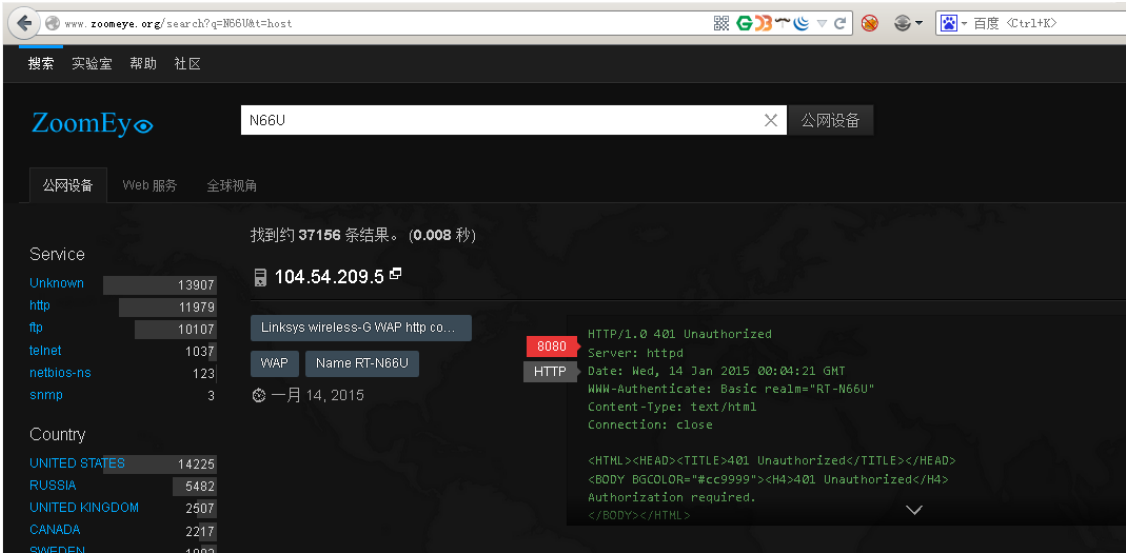
由于此次华硕路由器命令执行漏洞难以使用无损探测方法探测，我们仅根据版本型号去推测漏洞的影响范围。把受影响的华硕路由器型号放在 ZoomEye（http://www.zoomeye.org）中检索，我们得到以下数据：

型号	数量（个）
RT-AC66U	21776
RT-N66U	37156
RT-AC87U	1314
RT-N56U	23974

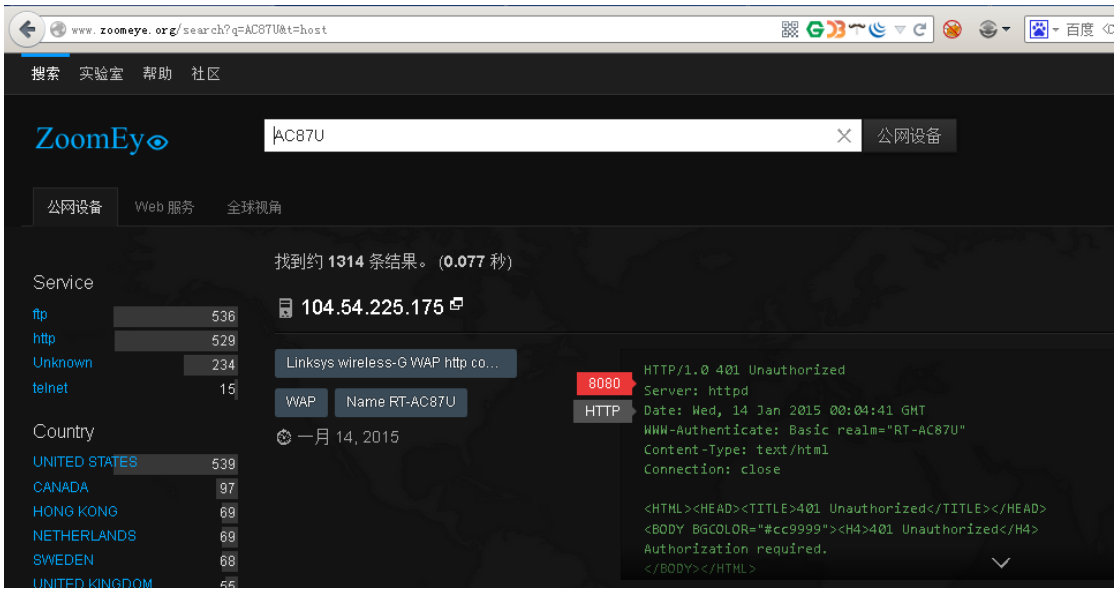
RT-AC66U，21776 个，如图：



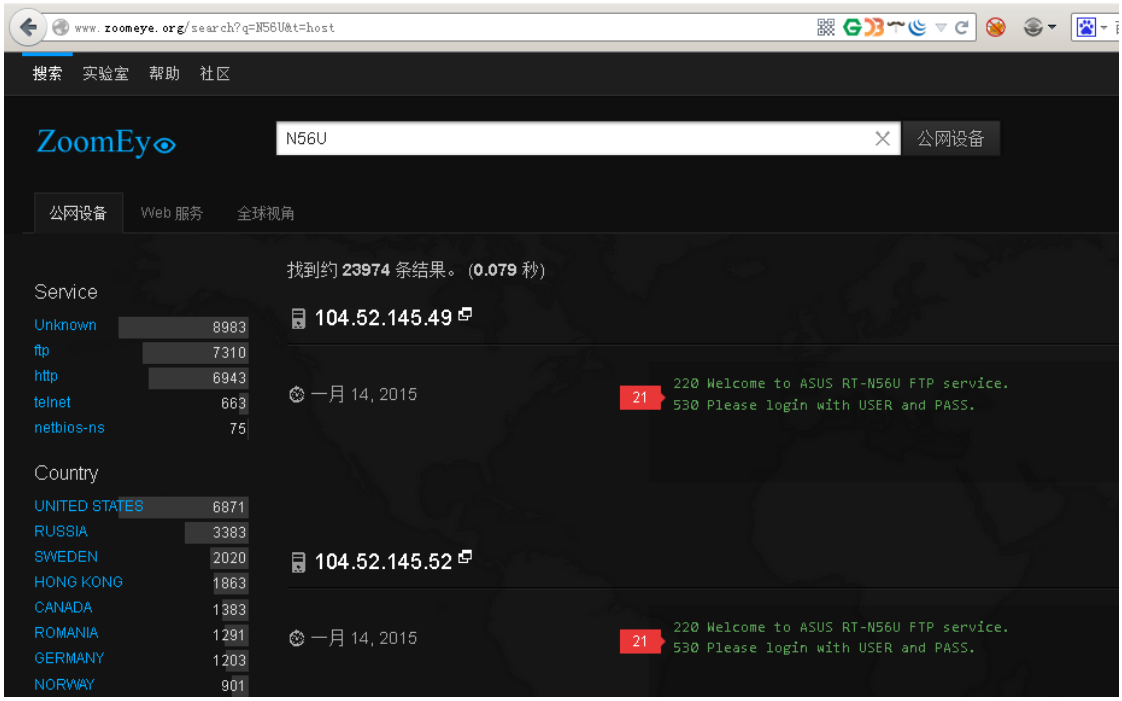
RT-N66U，37156 个，如图：



RT-AC87U，1314 个，如图：



RT-N56U，23974 个，如图：



路由器系统管理端口正常情况下是不会暴漏在公网上的，我们检索到的只是暴漏在公网上开放管理端口路由器设备的，相信还有更多的设备隐藏在背后。所以，建议使用者尽快升级华硕路由器系统。

4. 相关资源链接

1. http://www.freebuf.com/news/56074.html
2. https://github.com/jduck/asus-cmd
3. https://github.com/RMerl/asuswrt-merlin
4. http://www.asus.com.cn/Networking/RTAC68U/HelpDesk_Download/