基于goahead 的固件程序分析

前言

本文由 本人 首发于 先知安全技术社区： https://xz.aliyun.com/u/5274

最近在分析 `dlink` 的一个固件时遇到了用 `goahead` 开发的 `web` 服务。本文以一个 `github` 上的 开源项目为例简单介绍下对基于 `goahead` 的程序的分析。

https://github.com/Grant999/goahead-1

这里用的 `goahead` 程序的版本为 `2.5`

正文

编译运行

把源码下载下来，然后使用 `make` 编译即可。

```
$ make
.............................................
.............................................
.............................................
gcc -m32 -g -O0 -Wall -DWITH_NONAMESPACES -o webs -Os \
    -DWEBS -DOS="LINUX" -DLINUX -DUSER_MANAGEMENT_SUPPORT -DDIGEST_ACCESS_SUPPORT   -I.  -g -O0 -Wall  \
    main.o libwebs.a
tempnam' is dangerous, better use `mkstemp'
```

编译完成后当前目录下会生成一个 `webs` 的文件， 这个就是 `goahead` 编译完成生成的二进制文件。

然后运行它，就会在 `80` 端口起一个 `http` 服务（监听 `80` 端口需要权限， 所以用 `root` 运行程序）。

```
$ sudo ./webs
webdir: ./www
```

然后用 浏览器去访问

分析

要测试一个东西，首先需要尽可能的去了解它（信息搜集）。

goahead 开发 api

我们首先了解一下 `goahead` 的工作机制。

`GoAhead` 自身实现了一个 `web` 服务器所需提供的基本功能，此外它提供了多种方法供用户扩展服务器的功能， 其中包括 `asp`过程、`GoForms` 过程，`embedded JavaScript`以及外部 `cgi` 程序等， 用户可以根据这些接口开发出各种各样的功能。

对于 `goahead` 本身， 这个项目时间也非常就久了，安全性也得到了检验，所以我们分析的重点不是`goahead` 本身的代码，而应该是用户自定义的那些代码。

相关的 `api` 如下

websUrlHandlerDefine

```
websUrlHandlerDefine(T("/goform"), NULL, 0, websFormHandler, 0);
```

表示 对 `/goform` 的请求都交给 `websFormHandler` 函数处理。函数的参数列表如下

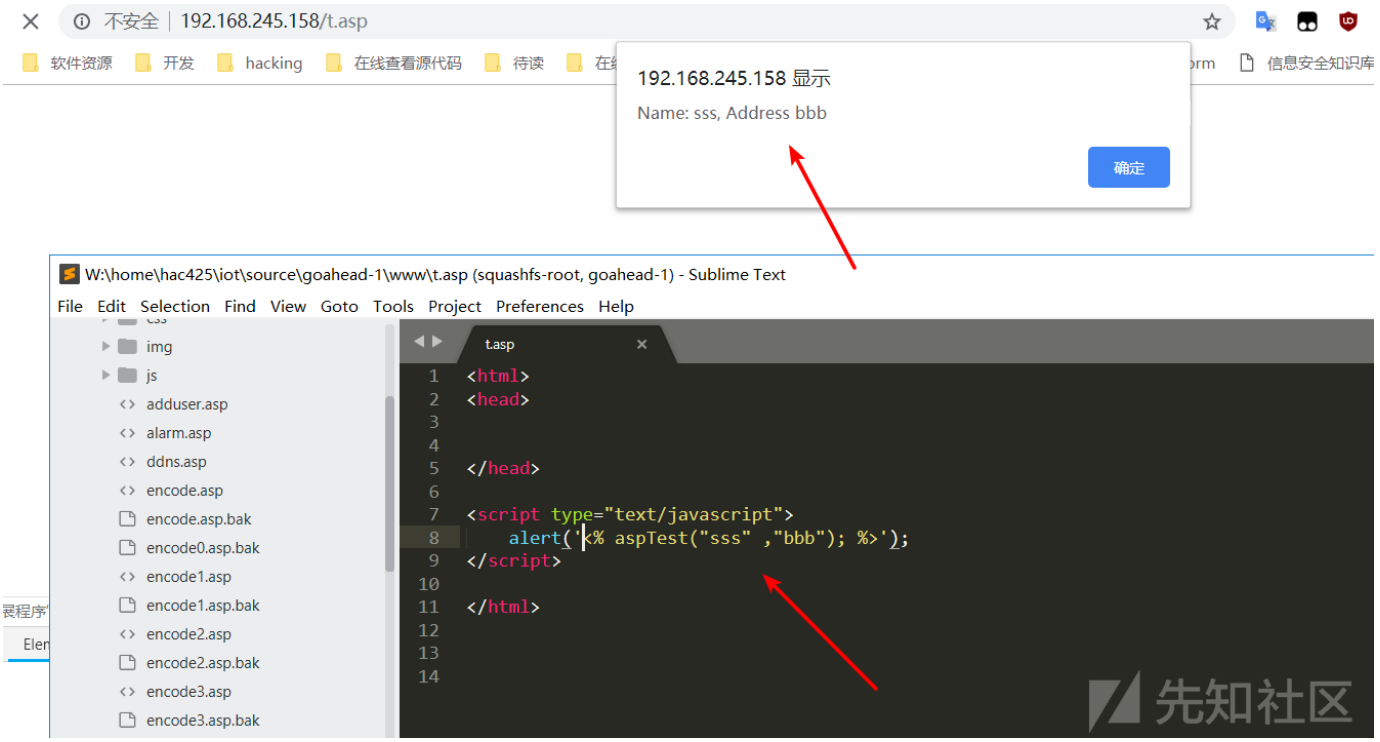
```
int websFormHandler(webs_t wp, char_t *urlPrefix, char_t *webDir, int arg,
    char_t *url, char_t *path, char_t *query)
```

其中 `wp` 这个参数里面包含了用户 请求的相关信息，比如请求头， 请求数据等。开发者通过 `wp` 这个参数就能获取到 用户请求的信息了。

websAspDefine

```
websAspDefine(T("aspTest"), aspTest);
```

当在 `asp` 文件中调用 `aspTest`, 实际调用的是这里 `aspTest` 这个 `c` 函数



ps:
调用的 `asp` 函数的语句需要用 `<% %>` 包围

websFormDefine

```
websFormDefine(T("privacy"), FormPrivacy);
```

和 `websUrlHandlerDefine` 差不多， 表示往 `/goform/privacy` 的请求由 `FormPrivacy` 这个函数进行处理。

漏洞分析

根据上面提到的 `api` 在源代码里面搜索引用，可以很快的找的注册用户自定义回调函数的位置。

位于 `initWebs` 函数

```
/*
 *  First create the URL handlers. Note: handlers are called in sorted order
 *  with the longest path handler examined first. Here we define the security
 *  handler, forms handler and the default web page handler.
 */
    websUrlHandlerDefine(T(""), NULL, 0, websSecurityHandler,
        WEBS_HANDLER_FIRST);
    websUrlHandlerDefine(T("/goform"), NULL, 0, websFormHandler, 0);
    websUrlHandlerDefine(T("/cgi-bin"), NULL, 0, websCgiHandler, 0);
    websUrlHandlerDefine(T(""), NULL, 0, websDefaultHandler,
        WEBS_HANDLER_LAST);

/*
 *  Now define two test procedures. Replace these with your application
 *  relevant ASP script procedures and form functions.
 */
    websAspDefine(T("aspTest"), aspTest);
    websFormDefine(T("formTest"), formTest);

    websFormDefine(T("osd"), FormOSD);
    websFormDefine(T("privacy"), FormPrivacy);
    websFormDefine(T("system"), FormSystem);
    //  websFormDefine(T("upload"), upldForm);

    formaspDefineLWT();
```


向上面4个 `websUrlHandlerDefine` 是 `goahead` 自带的，这里不管它。

通过对下面几个注册的函数的简单浏览，在 `FormPrivacy` 函数内部存在一个栈溢出漏洞。

下面对这个函数做一个简单的分析

```
void FormPrivacy(webs_t wp,char_t *path,char_t *query)
{
    char_t *pszOperate = NULL;

    char_t *pszColorY = NULL;
    char_t *pszColorU    = NULL;
    char_t *pszColorV = NULL;
    char_t *pszPosX      = NULL;
    char_t *pszPosY   = NULL;
    char_t *pszWidth    = NULL;
    char_t *pszHeight   = NULL;

    pszOperate = websGetVar(wp, T("Operate"), T(""));

    printf("Prvacy mask operate: %s\n", pszOperate);

    // get privacy para
    if (0 == gstricmp(pszOperate, T("get"))) { ...
    // set or del privacy mask
    else if (0 == gstricmp(pszOperate, T("set")) || ...
    else if (0 == gstricmp(pszOperate, T("clear"))) { ...
} « end FormPrivacy »
/*************************************************************
```


首先用

`websGetVar(wp, T("Operate"), T(""))`

获取 `Operate` 参数的值， 然后根据值的不同，进行不同的操作。

问题出在了 `set` 这操作的处理逻辑

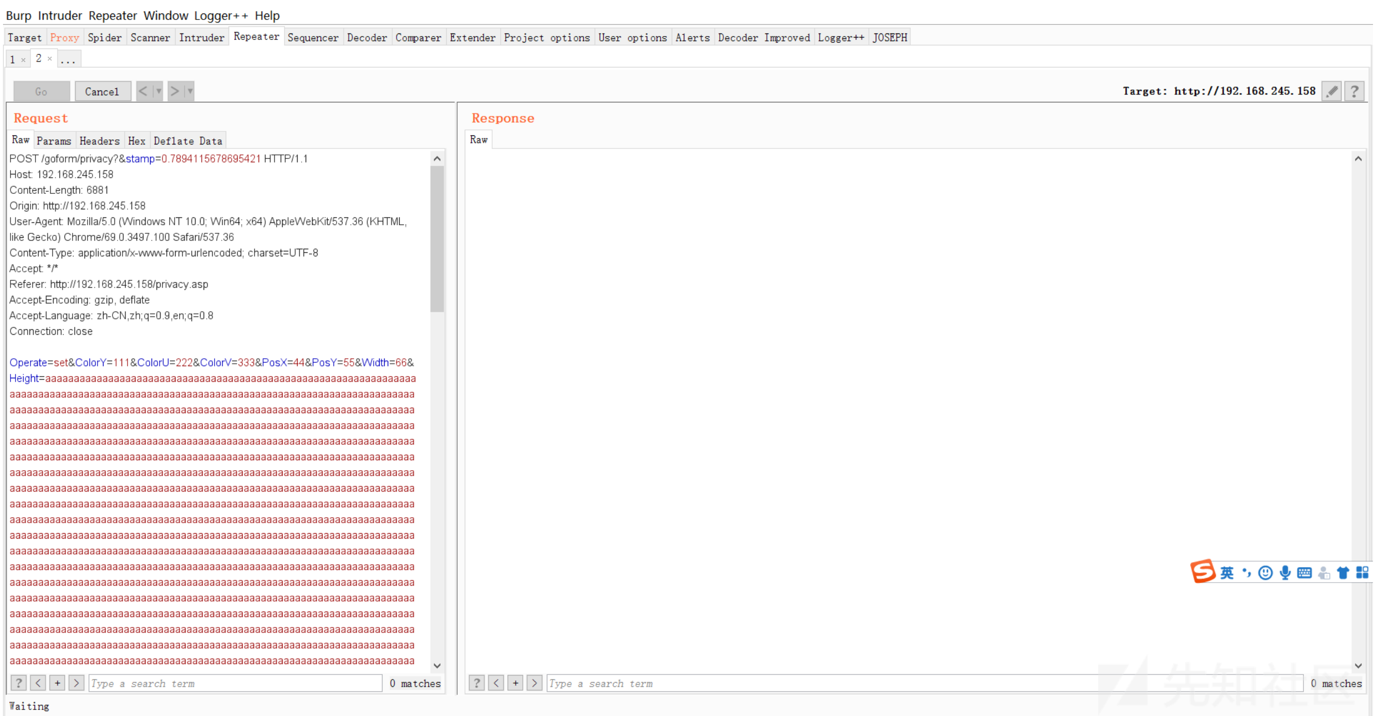
首先取出几个参数， 然后使用 `sprintf` 把参数填到 `szParam` 这个缓冲区 （缓冲区大小为 `20 * 20`)， 这里 `sprintf` 使用的是 `%s` 不会校验字符串的长度，所以当我们传一个很长的字符串作为 `Height` 的参数值，就会触发栈溢出。

触发+简单调试

通过搜索关键字，定位到往这里 发请求的应该是 `privacy.asp`

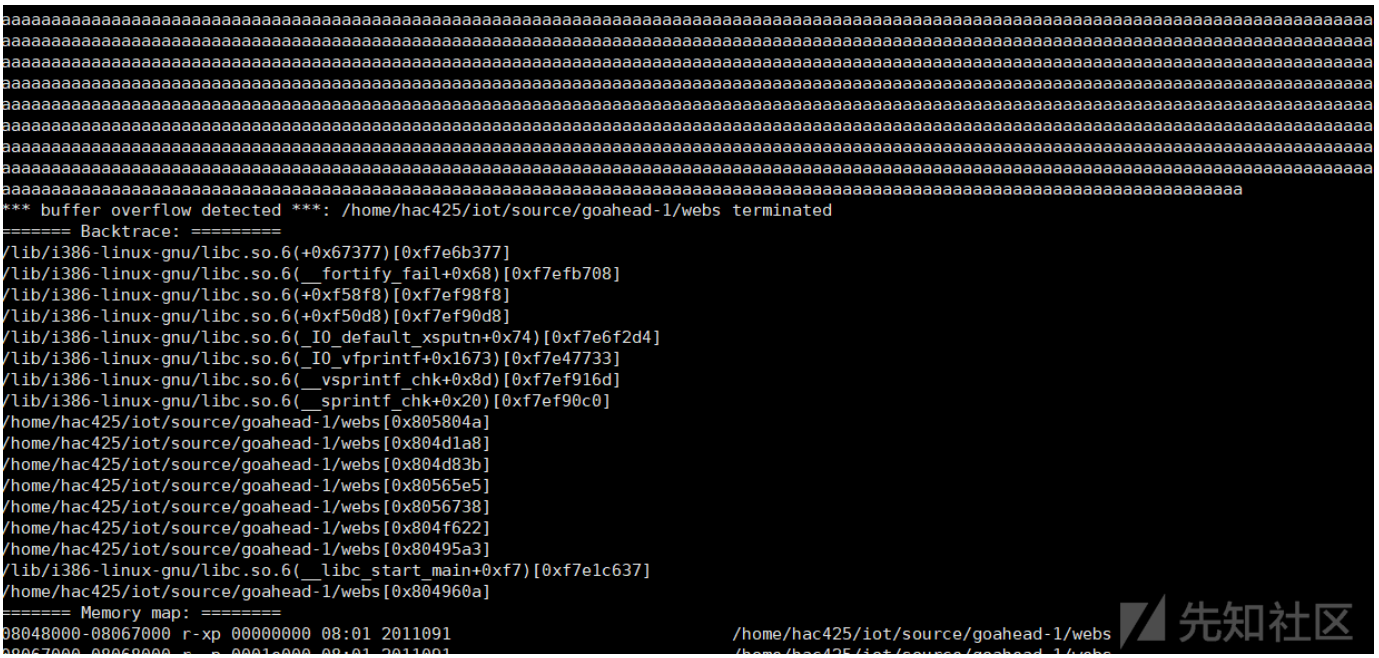
然后访问他

填些参数然后抓包， 修改 `Height` 的值为一个很长的字符串



同时在 `FormPrivacy` 设置一个断点， 发送请求过去，程序会断下来， 我们可以看看参数信息。

可以看到 `wp` 里面保存了此次请求的相关信息， 继续运行可以触发栈溢出



总结

分析 `goahead` 等可供开发者扩展的程序，分析的重点应该在那些自定义的代码上。

来源： https://www.cnblogs.com/hac425/p/9734471.html