# 网络安全实验报告

## 漏洞一：getenv("SCRIPT\_NAME")

### 寻找过程

搜索程序中容易造成缓冲区溢出的函数getenv()，结果如图1所示。

-> http.c 360 dir\_join(name, getenv("SCRIPT\_NAME"), indices[i]);

函数dir\_join的作用是将dirname复制到dst，再把filename添加到dst末尾。由于getenv函数不会检查环境变量的长度，而name数组的长度为1024bytes，所以getenv(“SCRIPT\_NAME”)的大小可能大于1024 bytes，造成缓冲区溢出。

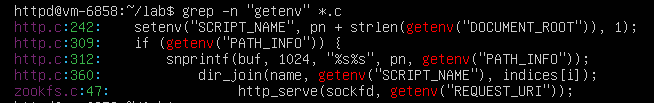


图1 程序中所有的getenv函数位置

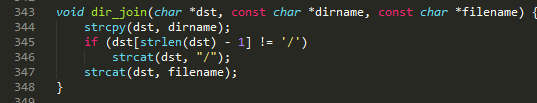


图2 函数dir\_join的实现

搜索环境变量“SCRIPT\_NAME”的设置位置

-> http.c 242 setenv("SCRIPT\_NAME", pn + strlen(getenv("DOCUMENT\_ROOT")), 1);

发现SCRIPT\_NAME的值由pn和DOCUMENT\_ROOT的大小组成，位于函数split\_path中

在函数split\_path中搜索局部变量pn

-> http.c 196 void split\_path(char \*pn)

发现pn是函数split\_path的形参

搜索split\_path的调用位置

-> http.c 283 split\_path(pn);

位于函数http\_serve中

在函数http\_serve中搜索局部变量pn

-> http.c 279 getcwd(pn, sizeof(pn));

-> http.c 282 strcat(pn, name);

发现pn是由当前路径和name拼接而成

在函数http\_serve中搜索局部变量name

-> http.c 273 void http\_serve(int fd, const char \*name)

是函数http\_server的形参

搜索http\_server的调用位置

-> zookfs.c 47 http\_serve(sockfd, getenv("REQUEST\_URI"));

发现name的实参是环境变量REQUEST\_URI

搜索环境变量“REQUEST\_URI”的设置位置

-> http.c 107 envp += sprintf(envp, "REQUEST\_URI=%s", reqpath) + 1;

发现REQUEST\_URI是reqpath设置的

在函数http\_request\_line中搜索局部变量reqpath

-> http.c 64 const char \*http\_request\_line(int fd, char \*reqpath, char \*env, size\_t \*env\_len)

是函数http\_request\_line的传入形参

搜索函数http\_request\_line的调用位置

-> zookd.c 70 if ((errmsg = http\_request\_line(fd, reqpath, env, &env\_len)))

搜索reqpath的定义位置

-> zookd.c 70 char reqpath[2048];

reqpath最大长度为2048，大于1024，所以当reqpath大小大于1024时，会触发缓冲区溢出

### 漏洞触发

reqpath是由客户端传入的请求解析获得的请求路径，所以需要修改客户端请求的路径，使其大于1024 bytes即可造成环境变量SCRIPT\_NAME的大小大于name的大小，从而造成缓冲区溢出，所以修改req = "GET /” +’A’\*1024 “ HTTP/1.0\r\n\r\n"。

## 漏洞二：sprintf(envvar, "HTTP\_%s", buf);

### 寻找过程

搜索程序中的sprintf出现的位置，如图3所示。

-> http.c 165 sprintf(envvar, "HTTP\_%s", buf);

由于envvar大小为512 bytes，而buf大小为8192 bytes，所以当buf实际大小大于512 bytes就会造成缓冲区溢出。

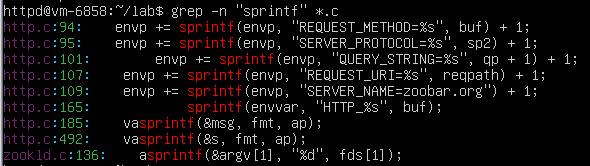


图3 sprintf出现位置

搜索局部变量buf的赋值位置

-> http.c 129 if (http\_read\_line(fd, buf, sizeof(buf)) < 0)

是函数http\_read\_line进行赋值的

搜索函数http\_read\_line的实现代码

-> http.c 36 int cc = read(fd, &buf[i], 1);

发现buf是从文件句柄fd读取的数据

搜索文件句柄fd

-> http.c 116 const char \*http\_request\_headers(int fd)

发现fd为函数http\_request\_line的形参

搜索函数http\_request\_line的调用位置

-> zookd.c 44 if ((errmsg = http\_request\_headers(sockfd)))

### 漏洞触发

envvar存储HTHP头部选项信息，大小为512bytes，所以只需要保证Http头部选项大小大于512bytes且保持正确的头部格式，即可造成缓冲区溢出，因此可以将req改为：

req ="GET / HTTP/1.0\r\nA: “ + ‘A’\*532 + ”\r\n\r\n"

## 实验结果

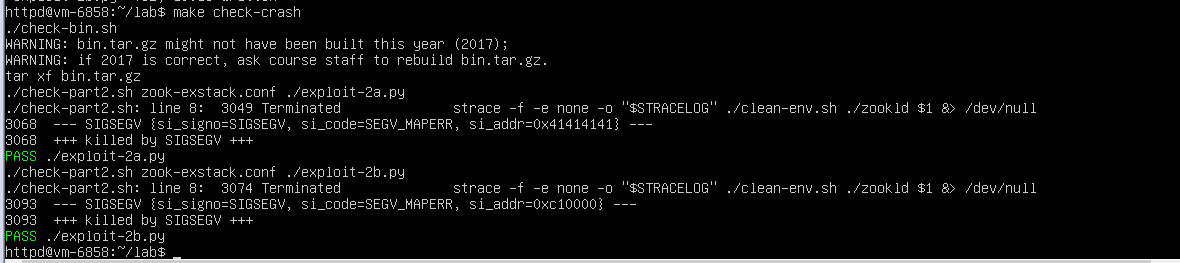


图4 实验结果