第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·安全技术· 文章编号：1000—3428(2006)23—0140—03 文献标识码：A 中图分类号：TP393.08

CGI模式中基于环境变量认证的安全性分析

钱鹏江，王士同
（江南大学信息工程学院，无锡 214000）

摘 要：分析了CGI模式中常见的基于环境变量的认证算法所存在的安全隐患，给出了根源是过分信赖服务端环境变量所致的结论。指出了基于随机带无规则的背景图片码认证的客户端提交是一种可行的、较安全的替换算法。
关键词：通用网关接口；认证；环境变量；图片码

Security Analysis of Authentication Based on Environment Variable in CGI Mode

QIAN Pengjiang, WANG Shitong
(School of Information Engineering, Southern Yangtze University, Wuxi 214000)

【Abstract】This paper describes the security trouble of authentication algorithm based on environment variable in CGI mode and points out it's caused by excessively trust the environment variable of server. Moreover, this paper also introduces a replaced algorithm based on validation of random image code with irregular background in client.
【Key words】CGI; Authentication; Environment variable; Image code

在CGI应用中合法性认证通常是不可或缺的，人们习惯于借助Server端的环境变量进行相关验证。但是这种验证习惯通常存在安全隐患，因为它所依赖的环境变量的值不一定真实有效。本文结合Winsock提交证实了这种隐患。据此，本文末尾给出了一种较安全的替代算法，即基于随机带无规则背景的图片码的认证算法。

1 CGI与环境变量

众所周知，CGI(通用网关接口0使得WWW环境具有了交互性，CGI的发展使WWW深受广大使用者喜爱，很多现在流行的网络技术(如ASP、JSP、PHP等0只是对CGI的改进，原理与CGI相同。

在CGI环境下，客户端传送一些信息给Web服务器，Web服务器把接收到的有关信息放入环境变量，然后再去启动所指定的CGI脚本以完成特定的工作。CGI脚本从环境变量中获取相关信息来运行，最后以HTML格式输出相应的执行结果返回给浏览器端，如图1所示。CGI脚本所需的输入参数来自于环境变量，而环境变量的值由Web服务器从客户端提交的内容来设置。

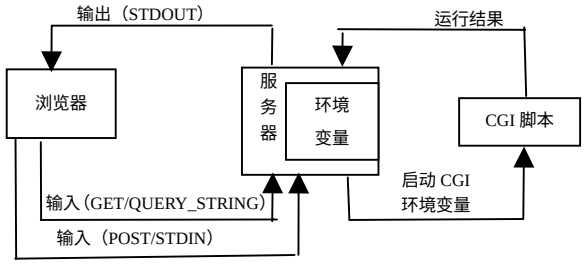

图1 CGI模式数据流程

2 基于环境变量的合法性认证

现今很多Web应用中的合法性认证算法就是基于环境变量进行的，如本文要研究的交互式网站常用的通过“HTTP_REFERER”环境变量进行“外部提交”判断的算法。

2.1 基于环境变量HTTP_REFERER的外部提交验证算法

算法如下：

```
Private Function CheckPostOutside()
  Dim server_v1,server_v2
  CheckPostOutside=False
  '用户若从 A 页链接到本页，则 HTTP_REFERER 变量保存了 A 页的 URL，如 http://www.uokoo.com/home/index.html
  server_v1=Cstr(Request.ServerVariables("HTTP_REFERER"))
  'SERVER_NAME 变量保存了本页所在的服务器名，如 www.uokoo.com
  server_v2=Cstr(Request.ServerVariables("SERVER_NAME"))
  '若 server_v1 从第 8 个字符开始的 len(server_v2)个字符与 server_v2 不同，则表明 A 页与本页不同属一个服务器，即外部提交
  If Mid(server_v1,8,len(server_v2))<>server_v2 Then
     CheckPostOutside=True
  End If
End Function
```

2.2 算法分析

显然这种验证算法的关键是HTTP_REFERER环境变量。如果HTTP_REFERER能准确无误地标识提交页(如算法中的A页0，则算法没问题，反之则不然。

分析发现这种算法存在严重的安全隐患。

在如下的环境中进行验证分析：

网站：www.uokoo.com，IP：211.90.124.3，网站目录/new/administrator/中包含addcity.asp和addcitydeal.asp两个文件。前者为新城市的添加页面，后者为提交处理页面。外

作者简介：钱鹏江(1979－)，男，讲师、博士生，主研方向：网络安全，人工智能；王士同，教授、博导
收稿日期：2006-08-09 **E-mail：**qianpjiang@126.com

加网络数据包拦截软件(本实验采用 Visual Sniffer 软件)。

其中 addcitydeal.asp 开始处包含如下外部提交判断：

```
if CheckPostOutside() then
  response.redirect "addcity.asp"
end if
```

首先打开 Visual Sniffer，以拦截和采样网络数据包信息用于数据分析。然后打开 www.uokoo.com/new/administrator/addcity.asp 页面，输入如图 2 所示内容后提交。最后关闭网络拦截。

图 2 addcity.asp 输入界面

查看 Visual Sniffer，得到拦截内容如图 3 所示。

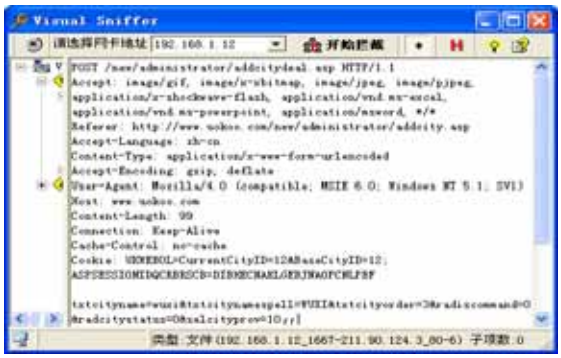

图 3 Visual Sniffer 拦截数据包内容

用户输入如图 2 所示内容后，addcity.asp 页面发送了如图 3 所示的数据包给 Web 服务器，然后 Web 服务器提取信息并填充相应的环境变量，再调用 addcitydeal.asp 页面(相当于 CGI 脚本 0 进行数据逻辑处理并更新数据库。

算法中环境变量“HTTP_REFERER”的值即来自于如图 3 所示数据包中的“ Referer ”字段。该“ Referer ”字段的值能否由用户自行设定？若能，就可能通过 addcitydeal.asp 中的外部提交检测，从而安全隐患就必然存在。

2.3 安全性验证

下面采用 VC++、Winsock API 进行验证。

主要算法如下：

```
…
char ip[50]="211.90.124.3"; //设定 Web 服务器的 IP
USHORT port=80;   //设定 Socket 绑定端口为 80
char value[1024]=
{"txtcityname=wuxi&txtcitynamespell=WUXI&txtcityorder=3&radiscommand=0&radcitystatus=0&selcityprov=10"};
//模拟 post 方式提交的表单内容
SOCKET sock;
struct sockaddr_in sin;
char sendbuf[1024*15]={0};
…
/* 设置 sockaddr_in 地址*/
sin.sin_family=AF_INET;
if(inet_addr(ip)!=INADDR_NONE)
  sin.sin_addr.s_addr=inet_addr(ip);
sin.sin_port=htons(port);
/*下面这段代码很重要，模仿图示内容设置发送数据包的各字段*/
char tempbuf[1024]={0};
strcat(sendbuf,"POST /new/administrator/addcitydeal.asp HTTP/1.1\n");
strcat(sendbuf,"Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*\n");
//用户自行制定 Referer 的值
strcat(sendbuf,"Referer: http://www.uokoo.com/new/administrator/addcity.asp\n");
strcat(sendbuf,"Accept-Language: zh-cn\n");
strcat(sendbuf,"Content-Type:application/x-www-form-urlencoded\n");
strcat(sendbuf,"Accept-Encoding: gzip, deflate\n");
strcat(sendbuf,"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)\n");
strcat(sendbuf,"Host: www.uokoo.com\n");
memset(tempbuf,0,sizeof(tempbuf));
sprintf(tempbuf,"Content-Length: %d\n",strlen(value));
strcat(sendbuf,tempbuf);
strcat(sendbuf,"Connection: Keep-Alive\n");
strcat(sendbuf,"Cache-Control: no-cache\n");
strcat(sendbuf,"Cookie: UKWEBOL=CurrentCityID=12&BaseCityID=12; ASPSESSIONIDSARARSCB=BAODFGAAAFOHFEOHPBBGCPID\n");
strcat(sendbuf,"\n");
strcat(sendbuf,value);
…
/*创建进行数据包发送的线程*/
CreateThread(NULL,0,senddata,&i,0,NULL);
…
DWORD WINAPI senddata(LPVOID lp)
{// 由 socket 函数创建套接字 sock
  SOCKET sock=socket(AF_INET,SOCK_STREAM,0);
  …
  int ret;
  //提出与服务器端的Socket建立连接的申请
  ret=connect(sock,(struct sockaddr*)&sin,sizeof(sin));
  …
  //发送 sendbuf 串内容，串末包含“\0”所以长度加 1
  ret=send(sock,sendbuf,strlen(sendbuf)+1,0);
  if(ret>0)
        printf("Send success!\n");
  else
        printf("Send failue!\n");
  char recvbuf[1024*10]={0};
  ret=recv(sock,recvbuf,sizeof(recvbuf),0);
  if(strstr(recvbuf,"100")||strstr(recvbuf,"200"))
        printf("提交成功!\n");
  else
        printf("提交失败!\n");
        closesocket(sock);
  return 1;
}
…
```

程序编译后执行，结果如图 4 所示。

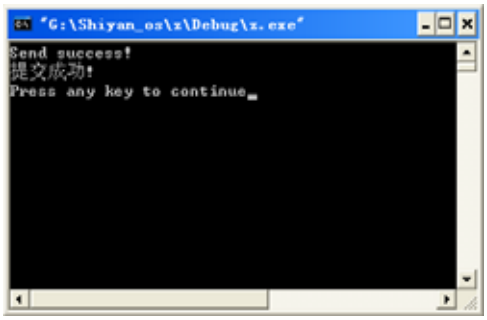

图 4 Winsock 提交结果

打开城市列表页面，“wuxi”城市名已被列出。说明事实提交成功。即客户端通过 Winsock 提交的内容顺利地通过了服务器端 addcitydeal.asp 的外部提交验证并被添加进数据库，而事实上根本未经由 addcity.asp 页提交数据。即外部提交发生了。

至此证实了该种通过 HTTP_REFERER 进行的外部提交认证算法存在严重的安全隐患。问题的原因是过分相信了服务端环境变量所致。

3 基于图片码的验证算法

一种比较安全的改进算法是基于图片码的验证。在输入页面生成随机的含验证码的图片，且图片有无规则的背景，由用户自行识别输入，然后在提交处理页面进行合法验证。这种算法目前是比较安全的，虽然有图片识别引擎,但是目前尚未发现“理想”的“绕道”方法。

产生随机含验证码的图片的算法如下：

(1)定义数组 strHexImage(240,1)，数组第一维用来存放 240 张含验证码及不规则背景的图片的十六进制形式的字符串，第二维则存放对应验证码的字符串值；形如：

strHexImage(0,0) = "47 49 46 38 39 …";strHexImage(0,1) = "LT28"

(2)intRandNum = CInt(Rnd * UBound(strHexImage))，生成 0～240 之间的一个随机整数。

(3)Session("strSecurityCode")=strHexImage(intRandNum, 1)， strHexImage(intRandNum,1)即是用户必须输入的唯一正确的验证码，用 Session 对象保存它，以便在提交处理页面中验证用户输入的验证码是否正确。

(4)sarryHex=split(strHexImage(intRandNum,0), " ")，将图片对应的十六进制字符串按空格分割并生成数组 sarryHex。

(5)使用 Response.BinaryWrite 方法循环输出 sarryHex 数组的内容。代码如下：

```
For i = 0 to CLng(UBound(sarryHex))
    Response.BinaryWrite ChrB(CLng("&H" & sarryHex(i)))
Next
```

4 结束语

本文分析了基于 HTTP_REFERER 环境变量认证存在的安全隐患，并提出了在客户端输入页面采用生成随机的含验证码的无规则背景图片的改进算法。因为图片码的随机性及无规则背景图片码的内容须人工识别和输入，一般软件或工具是很难获取的，所以该方法还是比较安全的。

参考文献

1 王艳平. Windows 网络与通信程序设计[M]. 北京: 人民邮电出版社, 2006-01.

2 汪晓平, 吴勇强. ASP 网络开发技术[M]. 北京: 人民邮电出版社, 2001-01.

3 CGI 环境变量[Z]. http://www.gaoshou.net/gotop/cgi/1.htm.

（上接第 136 页）

缓存常用(或分组、允许的列表)STA 的哈希链当前验证指针和对应验证公钥，从而实现对 STA 的预验证，抵御拒绝服务攻击，过滤非法消息转发给 AS。

(4)构建 AS 的安全服务目录服务。目录条目包括被认证实体标识、令牌(哈希链)当前指针、令牌认证公钥、加密公钥、证书其他信息。AS 可以在不忙的固定时间(如午夜 3 点)刷新信息库条目状态。撤销证书操作，直接修改目录中对应实体的状态。AS 基于目录服务操作，无需每次验证证书，提高操作效率。

(5)STA的计算效率。STA需要支持对称加/解密操作和公钥加密/解密操作。如果考虑STA为资源受限设备，上述协议还可以进一步优化，可以只要求STA执行公钥加密操作，用于构造$E_{AS_pk}(TS,R1)$，并约定PMK只产生于R1，此时STA无需公钥算法的解密操作。通过配置加密效率高的公钥算法，降低STA计算需求。

(6)AP 计算效率。AP 在认证过程中只需支持对称密码操作，无需执行公钥算法，从而保证其较高的工作效率。

表 2 FWAI 与 WAPI、802.11i 比较

协 议	证 书	交互消息	数字签名/验证			公钥加密/解密			认证模式
			STA	AP	AS	STA	AP	AS	
WAPI	自定义	4	0/1	1/1	1/1	1/1	1/1	0	隐性(滞后)
802.11i/EAP-TLS	X.509	10(12)	1/0	0/0	0/1	1/1	0/0	1/1	显性认证 SAT(可选)/隐性认证 STA/AS
FWAI	自定义	3	0/0	0/0	0/0	1/1	0/0	1/1	显性+隐性(多因子)

表 2 给出了 FWAI 与 WAPI、802.11i 比较。其中 802.11i 中 EAP-TLS 固有的交互会话数为 10，若完成协商产生 PTK，需加上 4 次握手的前 2 个会话，共需 12 个交互会话。

4 结论

本文从 WLAN 网络安全需求特点出发，提出了一种高效 WLAN 实体认证基础架构 FWAI，新机制基于哈希链构造认证令牌实现实体认证。与 802.11i/WAPI 比较，FWAI 认证和密钥协商效率更高，并具有强抗击拒绝服务攻击能力。新机制设计具有完整性，能够全面实现 ESS 结构中实体之间的认证与密钥协商功能；新机制设计追求简洁，结合对称密码算法、公钥密码算法、基于哈希链的实体认证等基础机制，发挥各自的长处，规避它们的负面效应。FWAI 的认证思想可以嵌入到标准 IEEE 802.11i 和 WAPI 中。

参考文献

1 IEEE P802.11i. Medium Access Control (MAC) Security Enhancements[S]. 2004-07.

2 GB 15629.11-2003 信息技术：系统间远程通信和信息交换局域网和城域网特定要求第 11 部分：无线局域网媒体访问控制和物理层规范[S]. 2003-05.

3 Aboba B, Simon D. PPP EAP TLS Authentication Protocol[S]. RFC 2716, 1999-10.

4 张浩军, 陈 莉, 祝跃飞. WAPI 数字证书应用研究[J]. 计算机应用, 2004, 24(12): 67-69.

5 李 勤, 张浩军, 杨 峰等. 无线局域网安全协议的研究和实现[J]. 计算机应用, 2005, 25(1): 160-162.