ASP 代码安全

湖南师范大学　田支斌

ASP(Active Server Pages)是微软开发的基于Windows Nt Server和Internet Information Server的服务器脚本运行环境。它是一种高效的动态网页开发技术,便于学习,易于使用。即使微软推出了新的ASP.Net技术,ASP在互联网上仍然被大量使用,足以说明ASP的强大生命力。ASP因为其使用的普遍性及源码的开放性,使得ASP应用的同时面临一系列的安全问题,如数据被非法下载,用户帐号泄密,数据库被非法访问等,ASP Web网站安全是一个复杂的系统问题,本文将从四个方面就代码本身的安全进行较为深入的探讨:

防止文件被非法下载

避免包含文件被下载

在网站设计中,为了便于管理和维护,我们常将程序设计成多个模块,然后用服务器的包含文件命令<!--#include file="" -->将各模块包含在某一程序当中。过去许多程序员喜欢将这些文件以*.inc形式命名,但是ASP脚本环境只能解释以特定后缀名的文件,如*.asp,*.asa,*.htm等,对于*.inc的文件,ASP脚本环境无法对其解释,web服务器则将其直接下载到本地浏览器,从而暴露ASP程序的各种重要信息。为避免这种情况发生,我们将*.inc改名为*.asp,这样即使包含文件的地址暴露,浏览者也无法将其源码下载到本地。

避免Access数据库文件被下载

用Access数据库结合ASP编制小型Web程序,简单实用,深受广大程序员的喜爱,但是用这种方式编程需将Access的数据库文件存于ASP程序目录之下,如同上面的第一点所论述的一样,一旦浏览者获知了数据库文件的具体地址,数据库文件将会被浏览者下载到本地,从而暴露大量的重要信息。但是我们如果将Access数据库的文件*.mdb改名为*.asp或*.asa,则即使浏览者获知了数据库文件的具体地址,当浏览者试图下载这些*.asp文件时,浏览者将会碰到访问错误,数据库在浏览器以乱码显示,同时在数据库的连接字符串,我们将相应的*.mdb改为*.asp或*.asa并不会影响程序对数据库的正常访问。

防止下载地址暴露

有时为了方便使用或其它一些原因,我们想将某些程序或资料提供下载但同时对些文件或资料下载时能进行访问授权控制,因此我们不能暴露我们的下载地址。我们可将文件或资料下载地址存储在数据库中,需要时再动态地从数据库中提取这些地址,最后我们用的地址转移语句(Response.Redirect "")将文件或资料直接下载到客户端,这样客户端将只会显示文件名而不会显示具体的下载地址,从而达到了隐藏下载地址的目的。

数据库安全

设定合适权限的数据库用户

现在较为大型的ASP程序多采用SQL数据库,SQL数据库安装时默认的用户是Sa,许多程序员喜欢将此用户作为Web程序访问数据库的用户,这是非常危险的作法,因为Sa在SQL数据库中具有超级用户的权限,它可以操作任何数据库,并可操作注册表,甚至能执行外部程序,一旦黑客通ASP程序非法入侵SQL数据库,后果将不堪设想。因此我们在使用SQL数据库作为ASP程序的后台数据库时,一定要为ASP程序设定一个只能访问必要数据库的SQL数据库用户,这样系统的安全性将得到极大提高。

过滤变量输入中的非法字符和改造危险的SQL语句

黑客通ASP程序非法入侵数据库,大多是利用ASP程序中的SQL语句设计漏洞构造畸型的SQL语句达到入侵数据库的目的,如很多网站把用户名和密码放到数据库中,在登陆验证时用以下sql:

```
sql="select * from user where username='"&username&"' and pass='"&pass&'"
```

此时我们只要构造一个特殊的用户名:如 ben' or '1'='1,此时实际访问数据库的sql语句将变为:

```
sql="select * from user where username='ben' or '1'='1' and pass='" '"&pass&'"
```

由于'1'='1'永远为真,这个语句将返回真值,所以验证通过。对于这类入侵,我们一方面要改造我们的sql语句,如上述语句可改为:

```
sql="select * from user where username='"&username&"'
```

查询数据库后,取得结果集Rs,再以下方式判别:

```
If Not Rs.Eof Then
   If Rs("password")=password Then
授权进入
Else
    拒绝访问
End If
Else
   拒绝访问
End If
```

在无法判断我们的SQL语句是否安全时，一种更通用的办法是将变量传给SQL语句时，对变量进行过滤，如：

```
Replace(Str,"'","''")      '过滤逗号
Replace(Str,";","")        '过滤分号
Replace(Str,"-","")        '过滤"-"
```

或判断用户输入数据的类型：

```
If IsNumeric(Request.QueryString("ID")=False Then
Response.Write("非法输入")
Response.End
End If
```

经过变量过滤后，大量SQL入侵将难于成功。

限制站外数据提交和页面的直接访问

有很多时候，SQL入侵是通过在浏览器地址栏中直接构造精心设计的SQL查询语句或通过网页中构造伪造的查询表单向服务器提供虚假数据，达到欺骗服务器的目的，这是一种通过站外提交数据入侵服务器的方法。对于这类入侵，通常有两种方法进行防范。

一种方法是在需要控制访问的页面加入Session变量，一旦浏览者试图直接访问受控制访问的页面时，当程序判定特定的Session变量为空时，则强制将页面转向访问的起始页。这种方法通常对需要提供密码才能访问的页面比较有效。设计者需要注意的是，在合法访问结束后，要注意通过将特定的Session变量赋值为空或用Session.Abandon语句将所有所有Session变量释放。这种方法的缺点有三点：

①当用户的访问量很大时，服务器上的Session变量的资源销耗将很大。

②Session变量的机制在客户端实际上是通过Cookie方式来实现的，客户有时为了安全会将Cookie功能关掉，这将使浏览者无法访问有Session变量控制的网页。

③当程序的特定Session变量还处于活动期间，程序也无法控制浏览者的站外数据提交。

另一种防止浏览非正常访问网页的办法是在程序中使用服务器变量HTTP_REFERER，这个变量保存用户的访问来源，如果直接从浏览器地址栏中输入地址，变量HTTP_REFERER的值将为空，如果从其它服务器通过构造表单向服务器提交数据，变量HTTP_REFERER的值中将不会含有目标服务器的ip地址或域名。如果我们在任何需要进行来源控制的ASP程序的顶部加入如下语句：

```
If Left(Request.Servervariables("http_referrer"),21)<>Ip Then
Response.Redirect Rediurl
End If
```

其中Ip为网站的Ip地址或域名，21为Ip地址或域名的长度，Rediurl为访问的起始页。

通过这个方法将能阻止一切通过地址栏或站外表单向服务提交数据，也能防止站外的盗链接，同时这种方法对服务器的资源开销很小，对客户端也无特别的要求。但这种方法也有一些缺点：

①当我们在程序中使用Response.Redirect 语句时，即使转向地址从正常的服务器提交，变量HTTP_REFERER的值将为空，从而使正常的访问也被阻止。

②当我们使用脚本语言如JavaScript的窗体打开语句Window.open('')打开网页时，即使这个网页是通过网站页面的正常链接打开，变量HTTP_REFERER的值将为空，从而使正常的访问也被阻止。

因此两种方法各有千秋，如果我们在程序中综合使用两种方法，将能很好的起到防止恶意访问目的。

数据库内容加密

一般情况下，用来验证用户合法身份的帐户信息都存储在数据库中，通常情况下程序员都以明文形式将帐户信息存储在数据库中，这种情况使程序存在极大的安全隐患，一旦数据库被黑客意外获取，帐户信息将暴露无疑。如果我们将帐户信息加密后存储于数据库中，将大大增加黑客获帐户信息的难度，从而增加web程序的安全性。

数据库联接字符串加密

为了程序设计的方便，程序员常常将数据库连接字符串作为一个单独的文件保存起来，在这个文件中包含了数据库的地址(远程或本地地址)，访问数据库有效用户名和密码，显然这个文件是Web程序最需要保密的内容。为有效地防止这段ASP源代码泄露，可以对这个文件进行加密。实质上也就是对部分ASP页面进行加密。一是使用组件技术将编程逻辑封装入DLL之中；二是使用微软的Script Encoder对ASP页面进行加密。使用组件技术存在的主要问题是每段代码均需组件化，操作比较繁琐，工作量较大，而使用Encoder对ASP页面进行加密，操作简单、收效良好。Script Encoder的运行程序是SCRENC.EXE，使用方法是：

```
SCRENC [/s] [/f] [/xl] [/l defLanguage ] [/e defExtension] inputfile outputfile
```

(其中：/s 是屏蔽屏幕输出；/f 指定输出文件是否覆盖同名输入文件；/xl 指是否在.asp文件的顶部添加@Language指令；/l defLanguag 指定缺省的脚本语言； /e defExtension 指定待加密文件的扩展名。)

IP 地址控制

有时我们想控制网站的某些资源只能被授权范围内的IP地址访问，一般情况下我们通过在ASP中使用Request.ServerVariable("REMOTE_ADDR") 来取得客户端的IP地址，但Request.ServerVariable("REMOTE_ADDR")无法无法区分获得的IP地址是来自代理服务器还是来自于客户端的

SELinux 安全策略配置

成都电子科技大学　李云雪　苏智睿　王晓斌

随着Internet的发展，分布式应用迅速普及，加强整个分布式系统安全的前提是端系统必须能够基于机密性和完整性的要求实施信息的隔离，而操作系统提供的安全机制是确保信息隔离的基础。目前，未经安全加强的主流操作系统一般采用自主访问控制(DAC)机制，DAC仅仅依赖用户标识和属主权限来决定访问策略，忽略了其他安全相关的信息如用户的角色，数据的敏感性和完整性，可执行程序的功能和可信任度。所以仅使用自主访问控制(DAC)来加强系统的安全性是不充分的。

通过实施强制访问控制(MAC)机制来加强操作系统的安全性可有效解决上述DAC机制的脆弱性。MAC的访问策略基于系统范围统一分配的标签，MAC的安全策略由系统安全策略管理员定义，实施于系统中所有的主体(进程)和客体(文件，网络接口)。MAC机制以软件，硬件或固件形式存在的可信计算基(TCB)为基础，以引用监视器(reference monitor)为不可旁路的监控机制，由系统强制实施安全访问策略，决定系统所有主体对客体的授权访问。

Security-enhanced linux，或简称SELinux是在linux内核的大多数子系统中应用了Flask安全体系结构，包括应用于文件，进程和套接字的强制访问控制。NAS发布了SELinux源码及其研究方案的相关背景资料，目的是为了检验SELinux的安全性并激励对主流操作系统上支持多种不同安全模块的安全体系结构的研究。

SElinux 体系结构

SELinux的安全机制采用了Flask/Fluke安全体系结构，此安全体系结构在安全操作系统研究领域的最主要突破是灵活支持多种强制访问控制策略，支持策略的动态改变。

Flask安全体系结构清晰分离定义安全策略的部件和实施安全策略的部件，安全策略逻辑封装在单独的操作系统组件中，对外提供获得安全策略裁决的良好接口。这个单独的组件称为安全服务器。在Flask安全体系结构中系统实施安全策略的组件称为客体管理器，客体管理器从安全服务器获得安全策略的裁决并通过绑订客体的标识和操控对客体的访问来实施安全策略裁决。客体管理器分布在SELinux内核的大多数子系统上，例如进程管理，文件管理，套接字管理等。应用层的管理子系统也被融入了客体管理器，如Windows X图象处理系统和数据库管理系统。Flask安全体系结构和客体的标识过程如图所示。

首先主体向客体管理器请求创建新的对象，客体管理器向安全服务器请求新客体的安全标识(SID)，请求主体的SID，相关客体的SID和新创建客体的类型作为参数传递；接着安全服务器按策略逻辑中的标识规则来决定客体的安全上下文并向客体管理器返回与安全上下文相关联的SID；最后客

真实IP地址，有时一些内部网的用户通过私设代理服务器，从而使外部网的用户也能访问内部网的资源。如何防止用户透过代理服务器访问内部资源呢？这时通过Request.ServerVariable("HTTP_X_FORWARDED_FOR")我们可透过代理服务器取得客户端的真实IP 地址。不过要注意的事，并不是每个代理服务器都能用Request.ServerVariables("HTTP_X_FORWARDED_FOR") 来读取客户端真实 IP，有时用此方法读取到的仍然是代理服务器的 IP。但是如果客户端没有通过代理服务器来访问，那么用 Request.ServerVariables("HTTP_X_FORWARDED_FOR") 取到的值将是空的。因此，如果要在程序中使用此方法，可以这样处理：

```
userip = Request.ServerVariables("HTTP_X_FORWARDED_FOR")
If userip = "" Then
    拒绝访问
Else
If Request.ServerVariables("REMOTE_ADDR")不在授权范围内 Then
    拒绝访问
End If
End If
合法访问
```

网页恶意脚本过滤

当我们在编制论坛、聊天室等读者可通过网页编辑文本的程序时，使用者可通过网页输入JavaScript恶意代码。如众所周知的聊天室攻击手段：<img src="javascript:n=1;do{window.open('')}while(n==1)" width="1">让别人开无数个窗口，从而致死机。因而我们在编制论坛、聊天室或留言薄之类的程序时，可用使用Server.Htmlencode()将使用使用者输入内容中的Html代码全部屏蔽掉，有时我们因各种原因不能屏蔽所有的 HTML、Javascript 语句，这时可以写一段程序判断客户端的输入，并屏蔽掉危险的 HTML、Javascript 语句，从而防止这种恶意攻击的发生。