

梭子鱼WAF测试报告

****

**Barracuda Networks (Shanghai) Co.,Ltd.**

**2009-5-24**

目 录

[第一章 测试方案 3](#_Toc231081971)

[1.1 测试环境与部署方式 3](#_Toc231081972)

[1.1.1 单臂模式 3](#_Toc231081973)

[1.1.2 双臂模式 4](#_Toc231081975)

[1.1.3 桥模式（透明模式） 5](#_Toc231081976)

[1.2 测试过程 7](#_Toc231081978)

[第二章 攻击过程 9](#_Toc231081979)

[2.1探测攻击 9](#_Toc231081980)

[2.2 跨站脚本攻击 12](#_Toc231081981)

[2.3 SQL注入攻击 15](#_Toc231081982)

[2.4 Cookie篡改 19](#_Toc231081983)

[2.5 页面防篡改 26](#_Toc231081984)

[第三章 Web应用防护 29](#_Toc231081985)

[3.1 探测攻击防护 29](#_Toc231081986)

[3.2 跨站脚本攻击防护 32](#_Toc231081987)

[3.3 SQL注入攻击防护 34](#_Toc231081988)

[3.4 Cookie窃取防护 36](#_Toc231081989)

[3.5 页面篡改防护 39](#_Toc231081990)

[3.6 深度内容防护 42](#_Toc231081991)

[3.7 应用层DDOS防护（CC攻击） 44](#_Toc231081992)

[第四章 测试总结 46](#_Toc231081993)

# 第一章 测试方案

## 1.1 测试环境与部署方式

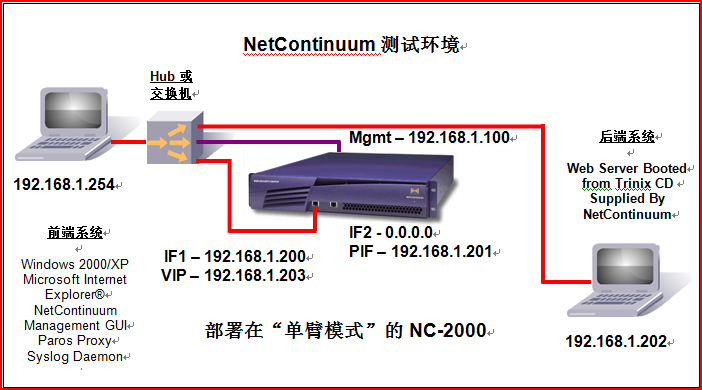
本次梭子鱼WAF的测试在手动搭建的测试环境中进行，测试的设备包含有梭子鱼WAF、WEB服务器以及一些基础的网络设备，例如交换机、路由器。梭子鱼提供了WEB服务器上的网站，名为www.badstore.net，此网站在开发时由于程序员的失误，遗留下了很多的漏洞，黑客可以轻松的攻陷此网站。

在测试部署中，共有三种部署方式可供选择：单臂模式、双臂模式、桥模式**（推荐采用桥模式）。**

### 1.1.1 单臂模式

单臂模式部署方式比较简单，将NC-Gateway的WAN口接在测试环境中的交换机中，保证与测试客户端和Web服务器在同一个网段。具体接口的IP地址如下表所示：

|  |  |
| --- | --- |
| **Client** | **192.168.1.254** |
| **Web Server** | **192.168.1.202** |
| **NC-Gateway Mgmt** | **192.168.1.100** |
| **NC-Gateway WAN(IF1)** | **192.168.1.200** |
| **NC-Gateway VIP** | **192.168.1.203** |



**优点 缺点**

比其他模式简单，无需改变物理网络结构 需要改变DNS解析

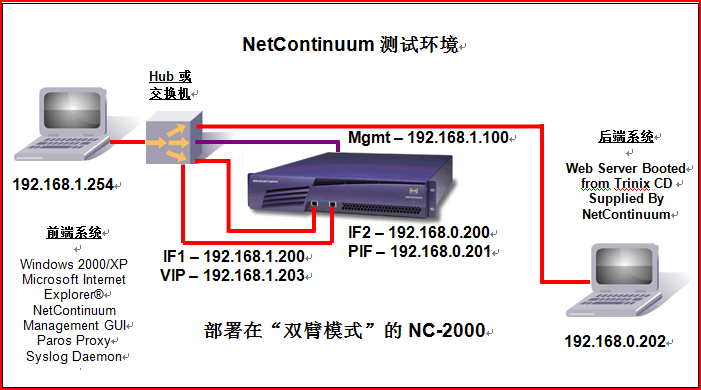
无多路径访问服务器（旁路部署） 吞吐量下降

不易于与企业当前的负载均衡整合安全性不及反向代理模式，因为能够直接访问真实服务器

### 1.1.2 双臂模式

双臂模式的部署方式是将NC-Gateway串接在网络中，并且将WAN口和LAN口接在与服务器相连的交换机上。但是WAN口和LAN口的IP地址配置为不同的网段，具体IP地址如下表所示：

|  |  |
| --- | --- |
| **Client** | **192.168.1.254** |
| **Web Server** | **192.168.0.202** |
| **NC-Gateway Mgmt** | **192.168.1.100** |
| **NC-Gateway WAN(IF1)** | **192.168.1.200** |
| **NC-Gateway LAN(IF2)** | **192.168.0.200** |
| **NC-Gateway VIP** | **192.168.1.203** |



**优点 缺点**

功能全面，包括负载均衡和SSL加速 需要改动服务器IP地址以及DNS解析

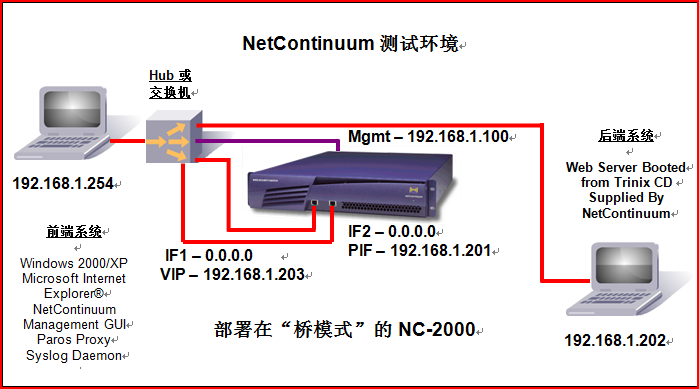
服务器被完全隔离，安全性最高 恢复时需要恢复所有的改动

快速高可用性 部署时需要中断业务

### 1.1.3 桥模式（透明模式）

桥模式的部署方式与双臂模式的部署方式基本相同，都是将NC-Gateway设备串联到网络中。但是在NC-Gateway上的WAN口和LAN口上不用配置IP地址，只需要配置虚拟站点的IP地址。具体IP地址的配置如下表所示：

|  |  |
| --- | --- |
| **Client** | **192.168.1.254** |
| **Web Server** | **192.168.1.202** |
| **NC-Gateway Mgmt** | **192.168.1.100** |
| **NC-Gateway VIP** | **192.168.1.203** |



**优点 缺点**

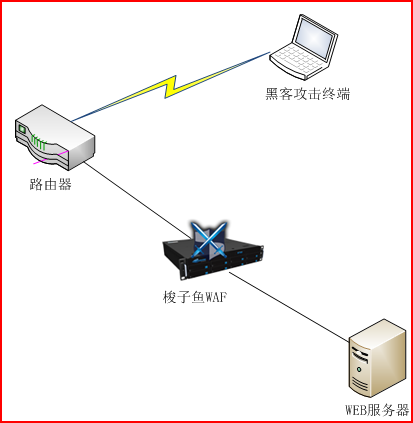
无需改动网络设置 对广播风暴和生成树环路敏感

无需改变真是服务器IP地址 对错误配置的弹性较差

不不具备负载均衡、SSL加速和TCP连接复用、不支持网络防火墙、带外检测、服务器组、主主冗余部署等功能

经过与移动技术人员的讨论分析，决定采用桥模式来进行测试。因为桥模式的部署方式最为方便，并且可以很完整的测试梭子鱼WAF对Web应用的防护功能。

本次测试的拓扑如下图所示：



## 1.2 测试过程

本次测试的主要过程为：

1. 首先用黑客终端来模拟黑客的访问，对WEB服务器进行各种的攻击，攻击的种类包含探测攻击、跨站脚本攻击、强制浏览攻击、SQL注入攻击、Cookie篡改攻击、和页面篡改攻击等攻击。通过各种攻击来观察一个WEB服务是如何被黑客攻陷的过程。
2. 然后通过在梭子鱼WAF上配置各种策略，来阻挡黑客采用的上述攻击，其中包含探测攻击防护、跨站脚本攻击防护、强制浏览攻击防护、SQL注入攻击防护、Cookie篡改攻击防护、以及WEB页面篡改防护。
3. 最后再次的模拟黑客的攻击手段来观察WEB服务器是否任然被攻破。

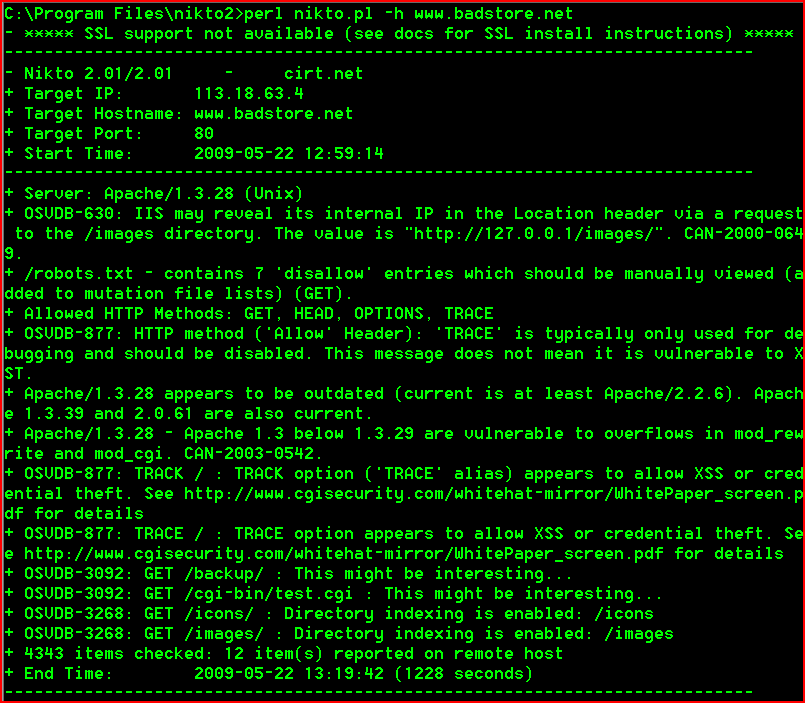
# 第二章 攻击过程

## 2.1探测攻击

探测攻击是黑客攻击一个Web应用之前做的准备工作，这一部分的攻击不会对网站的正常工作造成任何的影响。但是黑客可以通过探测攻击，收集到该网站的漏洞信息、版本信息、服务器信息、端口信息等。通过这些信息的收集，黑客可以有计划的对网站发起后续的攻击。

这里用Nikto软件来模拟黑客进行扫描网站的攻击，Nikto是一款开放源代码的、功能强大的WEB扫描评估软件，能对web服务器多种安全项目进行测试的扫描软件，能在230多种服务器上扫描出2600多种有潜在危险的文件、CGI及其他问题，它可以扫描指定主机的WEB类型、主机名、特定目录、COOKIE、特定CGI漏洞、返回主机允许的http模式等等。它也使用LibWhiske库，但通常比Whisker更新的更为频繁。

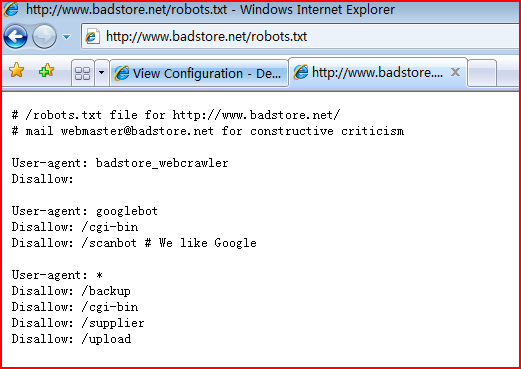
在没有使用NC2000进行防护之前，下图是使用nikto对测试网站www.badstore.net进行扫描的结果。



通过使用nikto软件的扫描，黑客可以很轻易的发现如下信息：

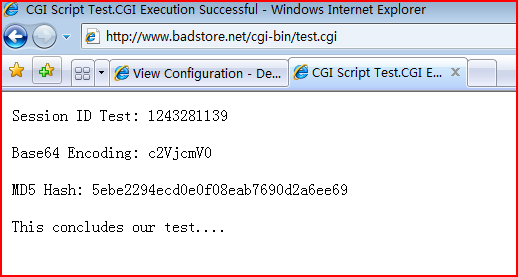
1. 此网站的WEB应用是用Apache 1.3.28软件开发的。
2. 在/（根目录）下存在文件robots.txt文件。
3. 在/（根目录）下存在backup、icons、images、cgi-bin目录，并且在cgi-bin目录下还存在test.cgi文件。

robots.txt文件为提供蜘蛛爬行程序是否能够访问目录的权限，里面记录了哪些目录可以被蜘蛛程序爬行，哪些却不可以。如果在URL地址栏中输入www.badstore.net/robots.txt，可以在页面上看到如下返回结果：



可以发现backup、cgi-bin、supplier、upload文件夹是不允许被蜘蛛程序爬行的。

然后再输入www.badstore.net/cgi-bin/test.cgi，可以发现如下信息：



其中我们可以发现比较重要的一点是，此网站采用的是Base64进行编码的。

同时我们还可以发现Backup可能为管理员备份网站时的文件夹。Supplier文件夹中存放的可能为供应商的一些机密文件。Cgi-bin文件夹中可能存放的是与网站有关的脚本。Upload可能存放的是上传文件。

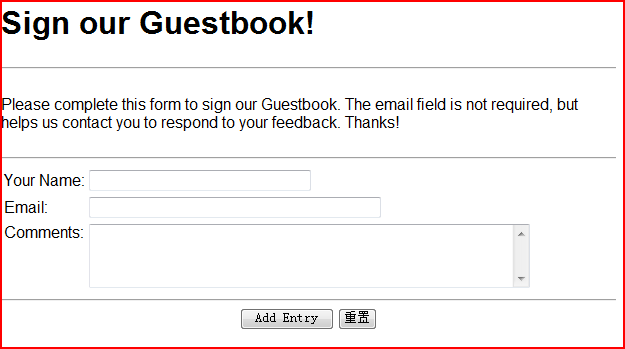
黑客可以记录下这些文件夹，然后方便后续攻击时对上述文件夹进行攻击。

## 2.2 跨站脚本攻击

跨站脚本攻击是通过在网页中加入恶意代码，当访问者浏览网页时恶意代码会被执行或者通过给管理员发信息的方式诱使管理员浏览，从而获得管理员权限，控制整个网站。攻击者利用跨站请求伪造能够轻松地强迫用户的浏览器发出非故意的HTTP请求，如诈骗性的电汇请求、修改口令和下载非法的内容等请求。

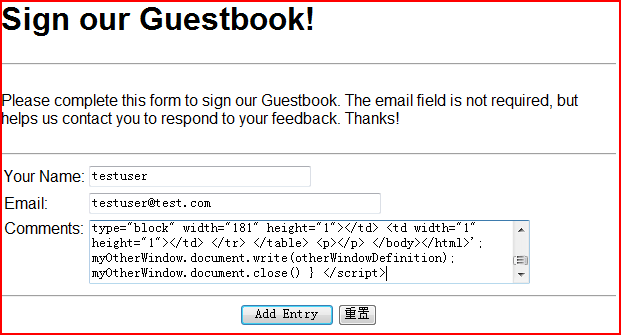
在目前所有的安全威胁中，Web安全无疑是被攻防双方观注程度最高，同时也是参与人数最多、安全漏洞与安全事件发生频率与数量最多、受害面最广的安全事故高发区，而在这个高居安全事件之首的事故频发区中，据多家国际权威机构统计，到目前为止，跨站脚本攻击是Web安全中最为常用，攻击成功率最高的攻击手段。因此，如果用一句话来总结跨站脚本攻击的危害，那就是：“跨站脚本攻击是到目前为止最受关注的、威胁最高的攻击手段”。

http://www.badstore.net/cgi-bin/badstore.cgi?action=guestbook页面是badstore进行订阅杂志的网页，与其他网站一样，这里需要输入用户的邮箱信息，以便发送电子杂志时使用。

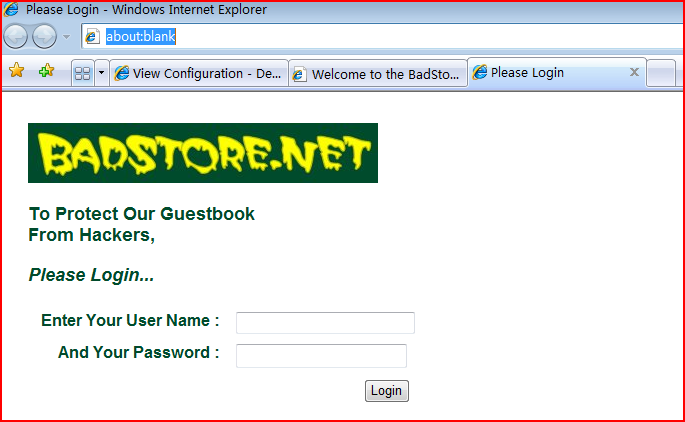


这里，我们填入用户的邮箱信息和注释信息，但是在注释信息中，可以加入如下的一段跨站脚本攻击的代码。

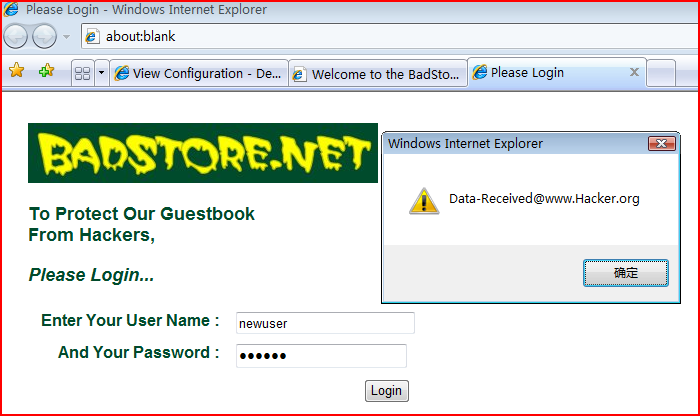
<script> myOtherWindow = open("", "secondWindow", "toolbar=no, location=no, directories=no, status=no, menubar=no, scrollbars=no, resizable=yes,width=575, height=375"); if (myOtherWindow != null); { var otherWindowDefinition = '<html><head><meta http-equiv="content-type" content="text/html;charset=iso-8859-1"><meta name="generator" content="Adobe GoLive 6"> <title>Please Login</title> </head><body bgcolor="#ffffff"> <table width="534" border="0" cellspacing="0" cellpadding="0" cool gridx="16" gridy="16" height="326" showgridx showgridy usegridx usegridy> <tr height="16"> <td width="16" height="325" rowspan="6"></td> <td width="517" height="16" colspan="5"></td> <td width="1" height="16"><spacer type="block" width="1" height="16"></td> </tr> <tr height="80"> <td width="517" height="80" colspan="5" valign="top" align="left" xpos="16"><img src="/images/BadStore.jpg" alt="" height="60" width="350" border="0"></td> <td width="1" height="80"><spacer type="block" width="1" height="80"></td> </tr> <tr height="108"> <td content csheight="96" width="256" height="108" colspan="3" valign="top" xpos="16"><font size="4" color="#004b2c" face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><b>To Protect Our Guestbook From Hackers, </b></font> <p><font size="4" color="#004b2c" face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><b><i>Please Login...</i></b></font></p> </td> <td width="261" height="108" colspan="2"></td> <td width="1" height="108"><spacer type="block" width="1" height="108"></td> </tr> <tr height="32"> <td content csheight="20" width="192" height="32" valign="top" xpos="16"> <div align="right"> <font color="#004b2c" face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><b>Enter Your User Name :</b></font></div> </td> <td width="16" height="121" rowspan="3"></td> <td width="309" height="32" colspan="3" valign="top" align="left" xpos="224"><input type="text" name="pp\_user\_id" size="24" tabindex="1" border="0"></td> <td width="1" height="32"><spacer type="block" width="1" height="32"></td> </tr> <tr height="36"> <td content csheight="20" width="192" height="36" valign="top" xpos="16"> <div align="right"> <font color="#004b2c" face="Arial,Helvetica,Geneva,Swiss,SunSans-Regular"><b>And Your Password :</b></font></div> </td> <td width="309" height="36" colspan="3" valign="top" align="left" xpos="224"><input type="password" name="pp\_password" size="24" tabindex="2" border="0"></td> <td width="1" height="36"><spacer type="block" width="1" height="36"></td> </tr> <tr height="53"> <td width="192" height="53"></td> <td width="128" height="53" colspan="2"></td> <td width="181" height="53" valign="top" align="left" xpos="352"><Input type="button" name="Sendit" VALUE="Login" onClick=alert("Data-Received@www.Hacker.org") ></td> <td width="1" height="53"><spacer type="block" width="1" height="53"></td> </tr> <tr height="1" cntrlrow> <td width="16" height="1"><spacer type="block" width="16" height="1"></td> <td width="192" height="1"><spacer type="block" width="192" height="1"></td> <td width="16" height="1"><spacer type="block" width="16" height="1"></td> <td width="48" height="1"><spacer type="block" width="48" height="1"></td> <td width="80" height="1"><spacer type="block" width="80" height="1"></td> <td width="181" height="1"><spacer type="block" width="181" height="1"></td> <td width="1" height="1"></td> </tr> </table> <p></p> </body></html>'; myOtherWindow.document.write(otherWindowDefinition); myOtherWindow.document.close() } </script>



在点击了“Add Entry”以后，可以正常的提交注册信息。但是当使用另外一个用户注册时，可以发现网站会自动的弹出一个新的对话框：



在新用户不知情的情况下，再次输入用户注册的信息。



此时，新用户甚至是管理员的用户名和密码就会被发送到黑客的邮箱中，此时该网站的控制权已经被黑客获取。

## 2.3 SQL注入攻击

随着B/S模式被广泛的应用，用这种模式编写应用程序的程序员也越来越多，但由于开发人员的水平和经验参差不齐，相当一部分的开发人员在编写代码的时候，没有对用户的输入数据或者是页面中所携带的信息（如Cookie）进行必要的合法性判断，导致了攻击者可以提交一段数据库查询代码，根据程序返回的结果，获得一些他想得到的数据。

SQL注入利用的是正常的HTTP服务端口，表面上看来和正常的web访问没有区别，隐蔽性极强，不易被发现。

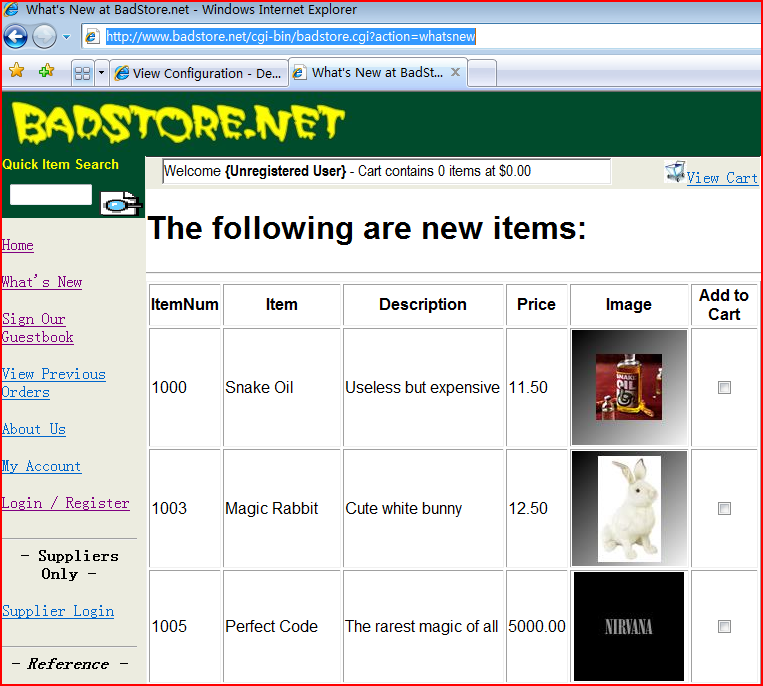
攻击过程简单，目前互联网上流行众多的SQL注入攻击工具，攻击者借助这些工具可很快对目标WEB系统实施攻击和破坏。

危害大，由于WEB编程语言自身的缺陷以及具有安全编程能力的开发人员少之又少，大多数WEB业务系统均具有被SQL注入攻击的可能。而攻击者一旦攻击成功，可以对控制整个WEB业务系统，对数据做任意的修改，破坏力达到及至。

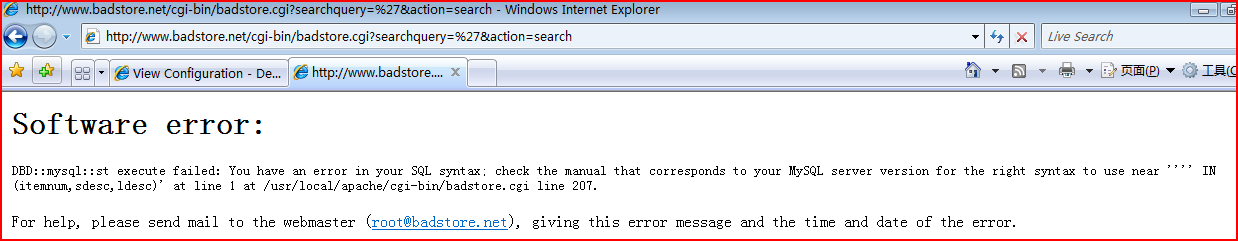
SQL注入的主要危害包括：

* 未经授权状况下操作数据库中的数据
* 恶意篡改网页内容
* 私自添加系统帐号或者是数据库使用者帐号
* 网页挂木马

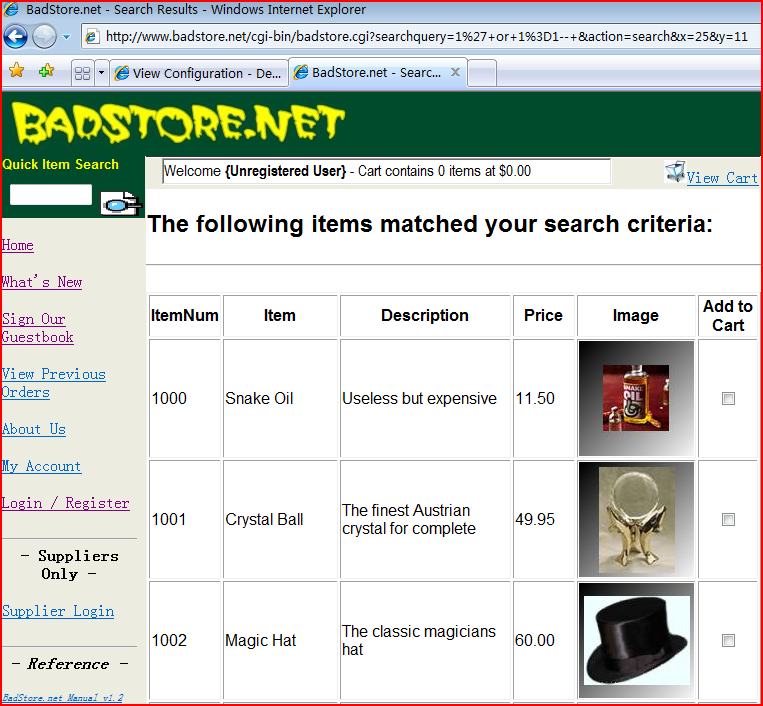
http://www.badstore.net/cgi-bin/badstore.cgi?action=whatsnew页面是提供商品列表的页面，在该页面的左上角有一个提供查询的工具栏：



在这个提供查询的工具栏中，如果随意的输入一个符号，例如一个单引号“‘”，页面会返回错误信息。

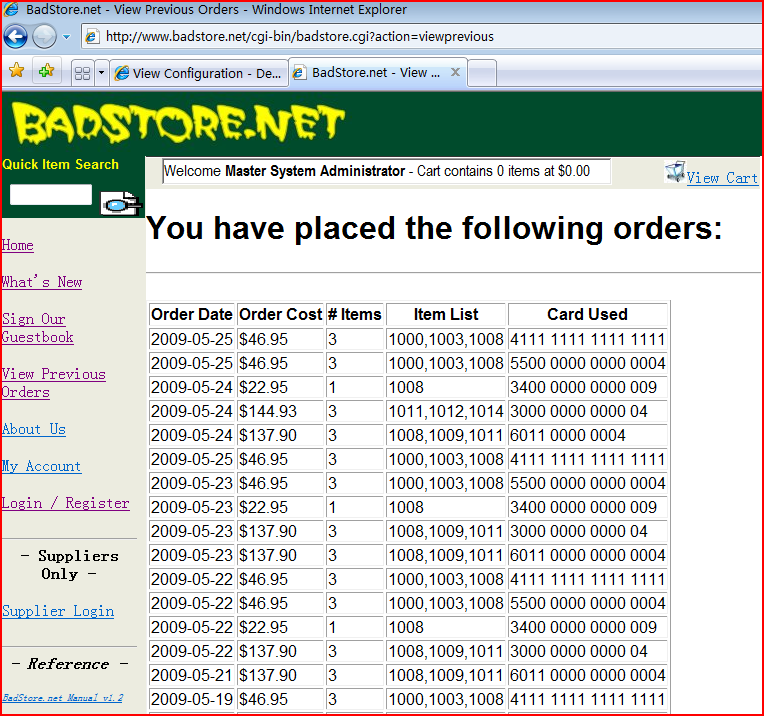


在这些信息中，我们不难发现该网站使用的数据库是MySQL，所以，黑客可以依据MySQL的语法规则，再次的在查询的工具栏中输入如下内容“1’ or 1=1-- ”，该内容是一个MySQL的语法规则，其意义为返回的结果为真，我们来看一下输入了上述内容以后，会发生什么：



可以发现，我们搜索出了数据库中所有的商品，其中包含了发布在网站上的商品，和还没有甚至是网站管理员不想发布在网站上的商品内容。

http://www.badstore.net/cgi-bin/badstore.cgi?action=loginregister页面是提供用户登陆的页面，我们利用上述的方式再次输入到对话框中，然后可以发现，此时可以成功的登录网站，并且登陆的身份为“**Master System Administrator**”。然后点击左边的“**View Previous Orders**”，可以发现，此时页面中显示的是所有用户的订单信息。

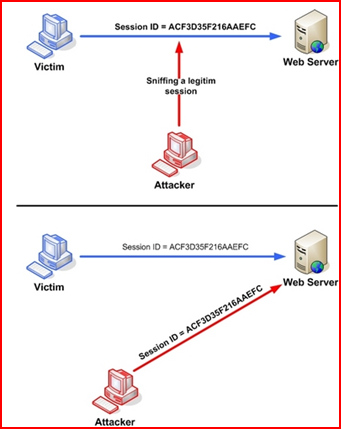


上述过程就简单的进行了一次SQL注入攻击的过程，虽然这里的SQL代码简单，并且没有对网站的实际运营造成影响，但是对于有经验的黑客来说，插入一段具有危害的攻击性代码并非难事。所以SQL注入的攻击危害极其严重。

## 2.4 Cookie篡改

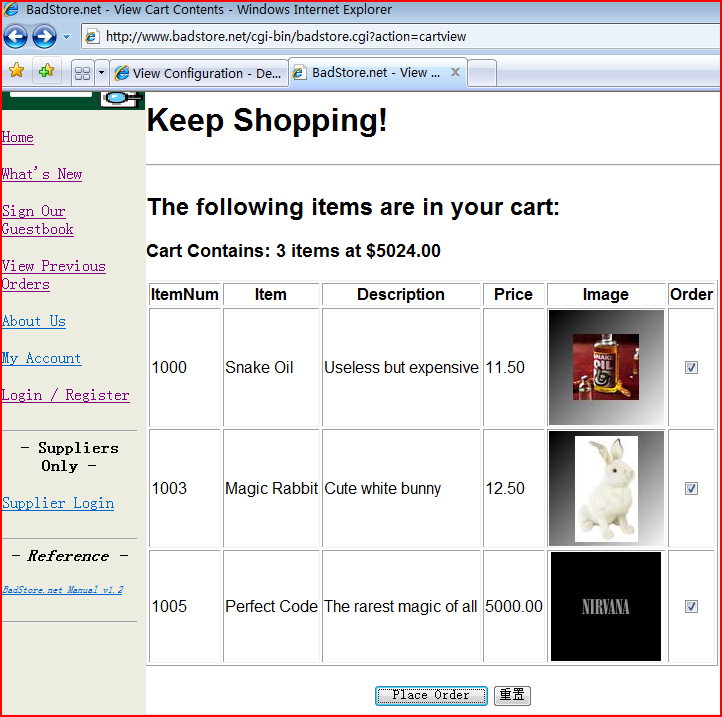
Cookie就是服务器暂存放在你的电脑里的资料（.txt格式的文本文件），好让服务器用来辨认你的计算机。当你在浏览网站的时候，Web服务器会先送一小小资料放在你的计算机上，Cookie 会帮你在网站上所打的文字或是一些选择都记录下来。当下次你再访问同一个网站，Web服务器会先看看有没有它上次留下的Cookie资料，有的话，就会依据Cookie里的内容来判断使用者，送出特定的网页内容给你。

Cookie的内容尤其重要，黑客可以利用截取的Cookie，来假冒真实的用户来与Web服务器进行交互。如果在移动的一些增值业务中出现了类似的Cookie劫持攻击，并且没有相应的防护措施进行防护，所造成的后果将不堪设想。下图为Cookie劫持攻击的过程演示：



为了实现黑客截取Cookie的动作，我们可以使用Paros，Paros是一款代理软件产品，这里利用Paros来获取访问页面时的Cookie。

首先进入登录页面，输入用户名和密码。然后选择想要购买的产品。

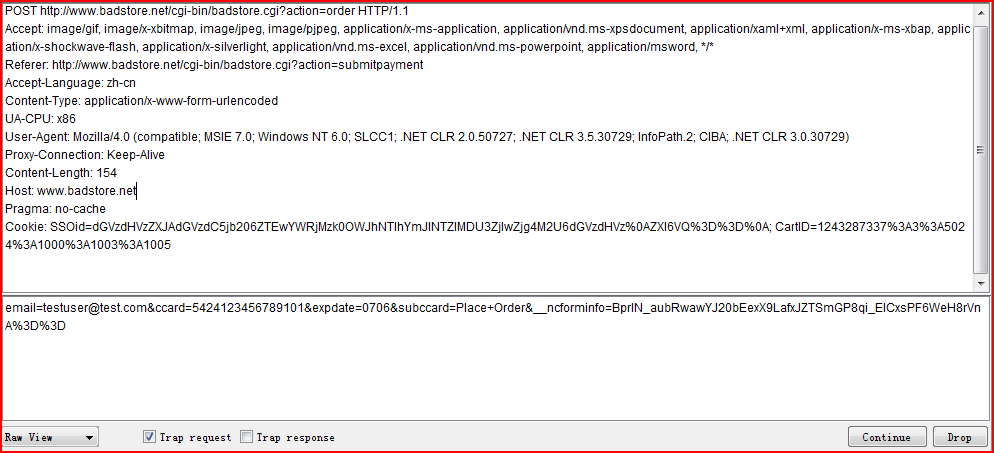


这里一共选择购买了3个商品，总金额为5024美元，在填写完信用卡信息以后，这笔交易的Cookie会发送到Web服务器的数据库中。

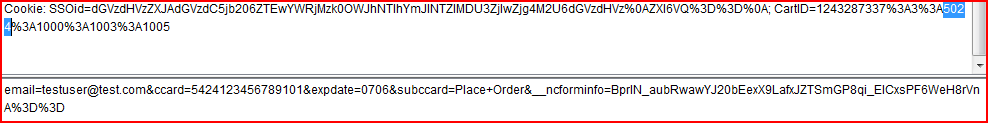
聪明的黑客会在Cookie提交到服务器之前，将Cookie截取下来，然后修改Cookie中的内容，再将修改以后的Cookie提交到服务器的数据库中，以下是操作的具体步骤：

1. 劫持Cookie

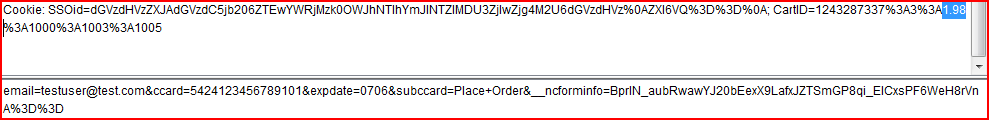
在Cookie提交到服务器之前，将Cookie截取下来。



1. 修改Cookie中交易的金额，将5024美元修改为1.98美元：

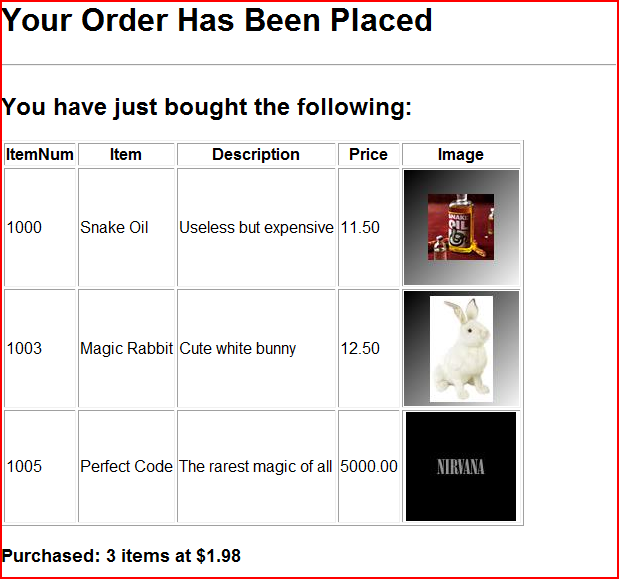


**修改以前为5024美元**



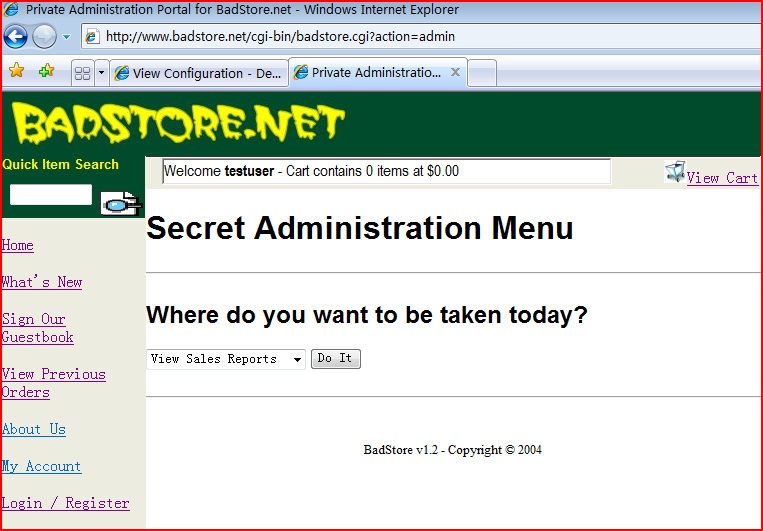
**修改以后为1.98美元**

在修改完Cookie以后，将Cookie提交到服务器的数据库中，这时，页面上返回的结果为这笔交易的金额为1.98美元。

****

于此同时，黑客发现了Cookie是未加密的，所以将进一步的对网站数据库发起攻击。

在注册信息的时候，黑客发现了注册完毕的URL为http://www.badstore.net/cgi-bin/badstore.cgi?action=register，如果将register改为admin，将会发生什么呢？



用户的身份将变成admin，此时我们再次的利用Paros来劫持提交的Cookie。由于在前面的探测攻击中，黑客已经发现此网站是用Based64来进行编码的。所以黑客可以利用Based64的编码规则来修改用户的身份，从而来完成另外一次的Cookie劫持攻击。

这是黑客劫持到的Cookie: SSOid=dGVzdHVzZXJAdGVzdC5jb206ZTEwYWRjMzk0OWJhNTlhYmJlNTZlMDU3ZjIwZjg4M2U6dGVzdHVz%0AZXI6VQ%3D%3D%0A

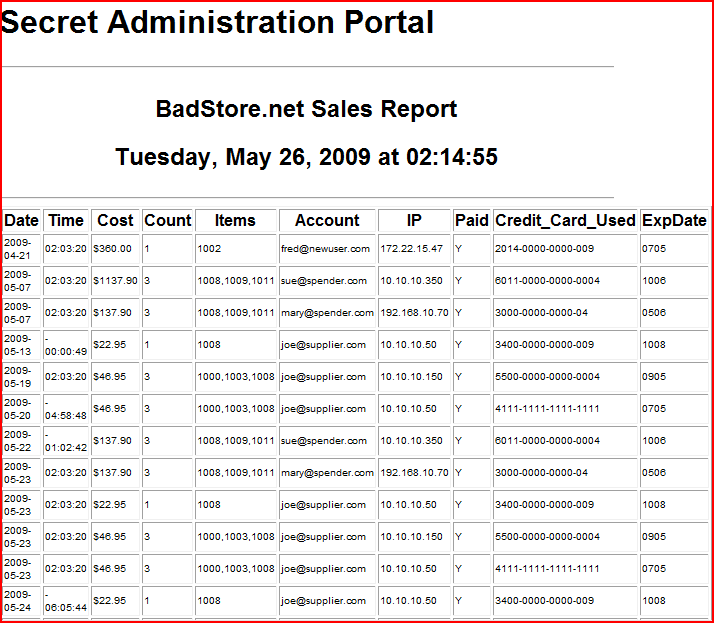
将cookie中的“dGVzdHVzZXJAdGVzdC5jb206ZTEwYWRjMzk0OWJhNTlhYmJlNTZlMDU3ZjIwZjg4M2U6dGVzdHVz%0AZXI6VQ%3D%3D”进行Based64的反编码后，可以将用户的身份改成admin，然后将修改以后的Cookie进行Based64的编码，最后将编码以后的Cookie提交到服务器端。

编码以前的Cookie：SSOid=dGVzdHVzZXJAdGVzdC5jb206ZTEwYWRjMzk0OWJhNTlhYmJlNTZlMDU3ZjIwZjg4M2U6dGVzdHVz%0AZXI6VQ%3D%3D%0A

编码以后的Cookie：

Cookie: SSOid=Z3dAYW9sLmNvbTplMTBhZGMzOTQ5YmE1OWFiYmU1NmUwNTdmMjBmODgzZTpHdWVzcyBXaG86QQ%3D%3D%0A

在提交了修改以后的Cookie以后，页面上可以显示出以admin权限查看到的所有用户的交易记录。

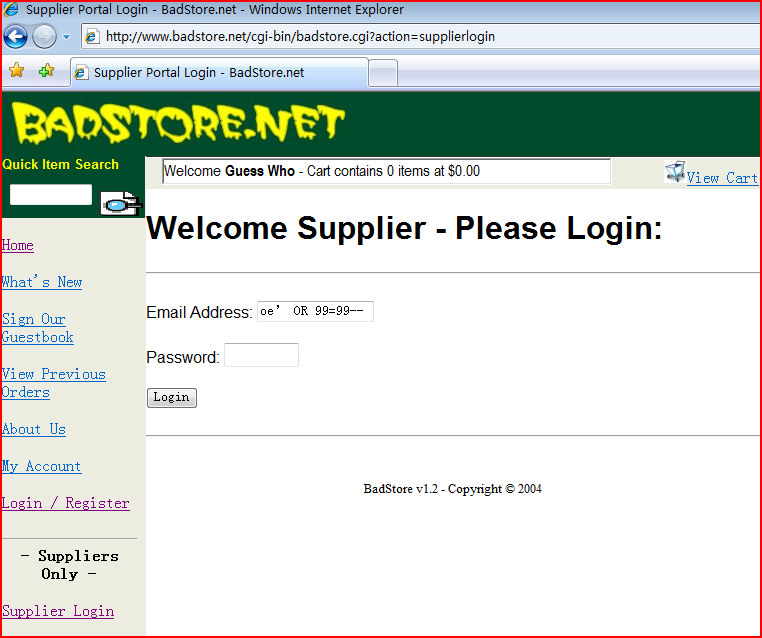


这样就完成了一次Cookie劫持的攻击，可以发现这种攻击的结果将是毁灭性的，在这个网站上购买任何产品，金额都将由黑客自己来决定，这是由于Cookie未加密所导致的。

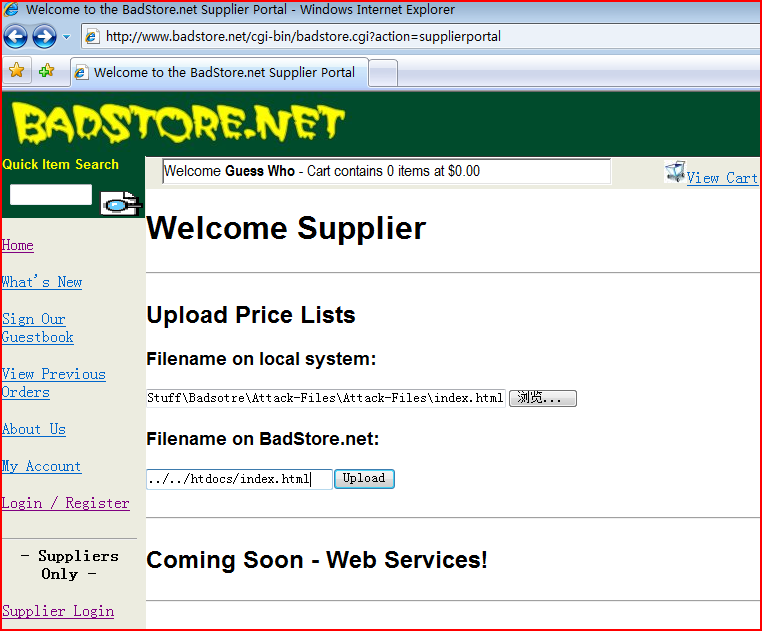
## 2.5 页面防篡改

页面篡改是黑客通过网站的漏洞，通过各种方法修改网页上的内容。黑客可以通过获取网站管理员的账户来修改网页，或者通过上传网页或者图片等方式来修改网页的内容。

在“Supplier Login”页面中，输入“Joe’ or 99=99-- ”（SQL注入攻击），来获取供应商用户的权限：



在获取了Supplier的权限以后，可以输入需要上传的文件，并且可以指定上传到服务器的具体路径：



由于没有对用户的上传权限做限定，所以这里用户指定了上传的主页文件到服务器的“../../htdocs/index.html”，此路径为Apache的宿主目录，而文件index.html为网站的主页。所以在上传好文件以后，再次访问www.badstore.net时，主页面已经被替换为黑客指定的网页了。



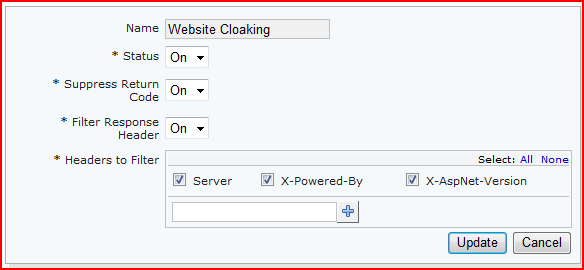
# 第三章 Web应用防护

上一章节主要介绍了作为一名黑客是如何攻击一个网站的：从最初的网站扫描，到试探性的攻击，到最后利用SQL漏洞获取了管理员的权限，最后将网站的主页修改为黑客自己定义的主页。

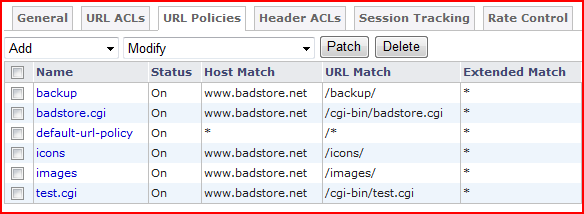
梭子鱼的WAF可以对目前绝大部分的Web应用攻击进行防护。

## 3.1 探测攻击防护

对于互联网上的探测攻击，梭子鱼的防护方式主要有2种，分别是网站隐身（Website Cloaking），和反爬行触发器（Anti Crawl Trigger）。

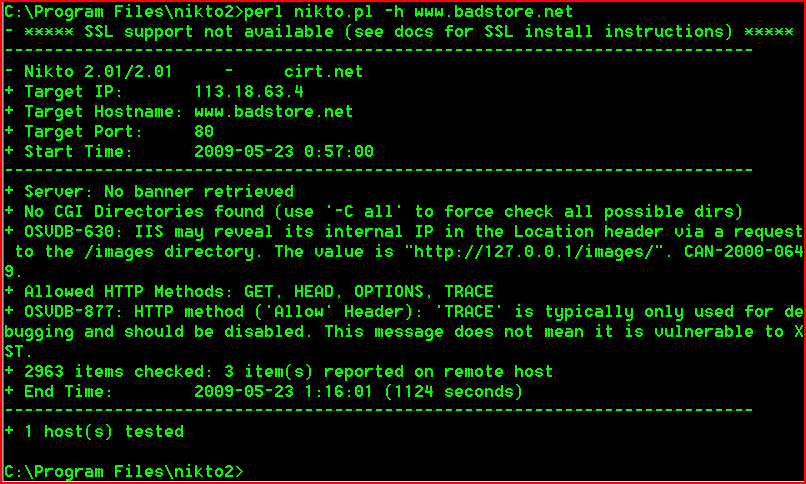


**Website Cloaking**

****

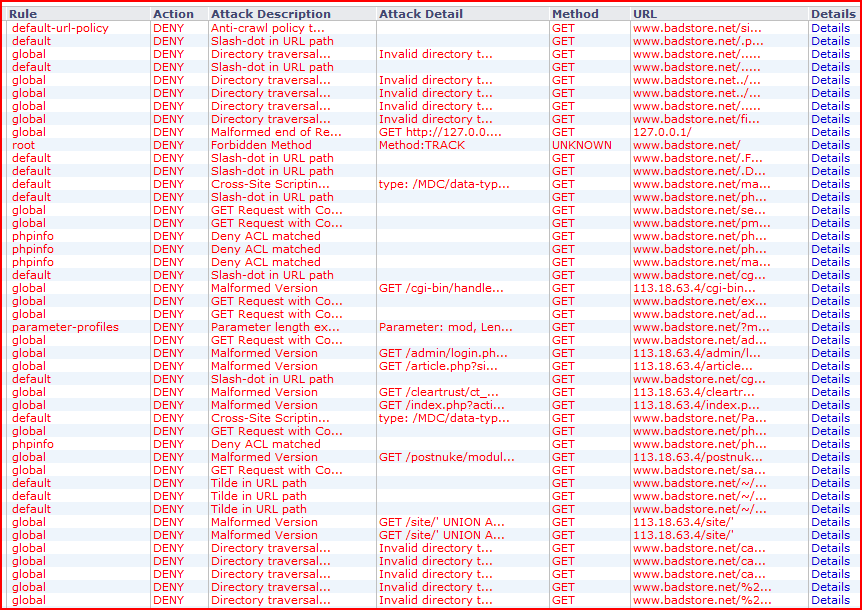
**Anti Crawl Trigger**

当开启了上述这2个功能以后，可以来再次观察一下nikto扫描网站以后的结果。

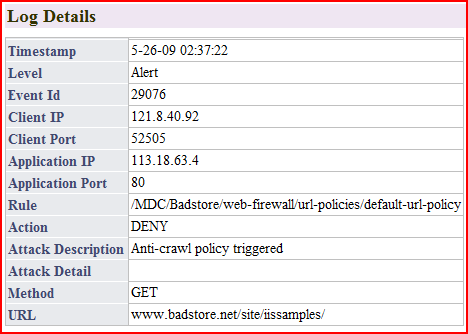
****

从图中我们可以很清楚的发现，与在没有开启反爬行功能和网站隐身功能时相比，所有的敏感信息都被完好的保护起来，黑客无法通过任何扫描软件扫描出网站的结构。

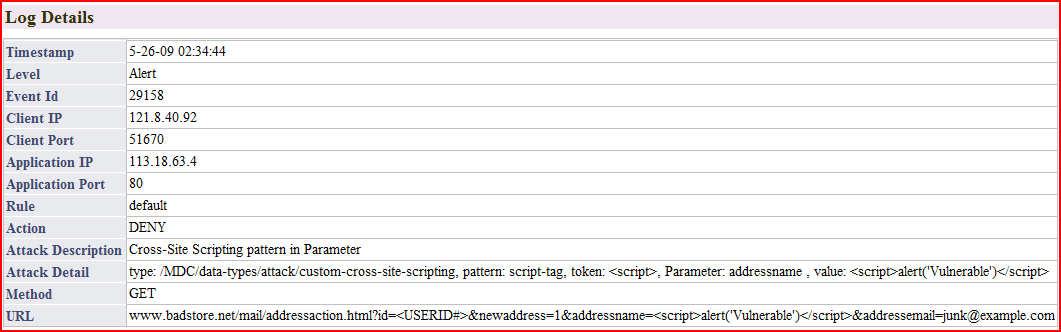
观察梭子鱼WAF上的日志可以发现，在nikto扫描的过程中，nikto对网站做了遍历目录攻击、跨站脚本攻击、爬行攻击、和SQL注入攻击等操作。



**反爬行防御过程中的部分日志**

****

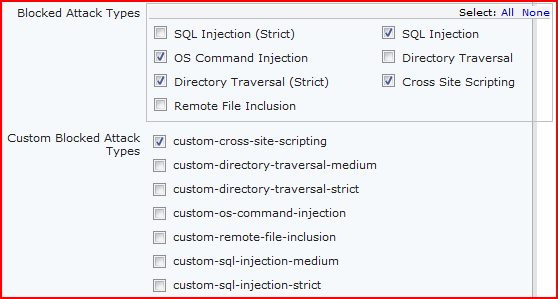
**违反了反爬行策略的详细日志**

****

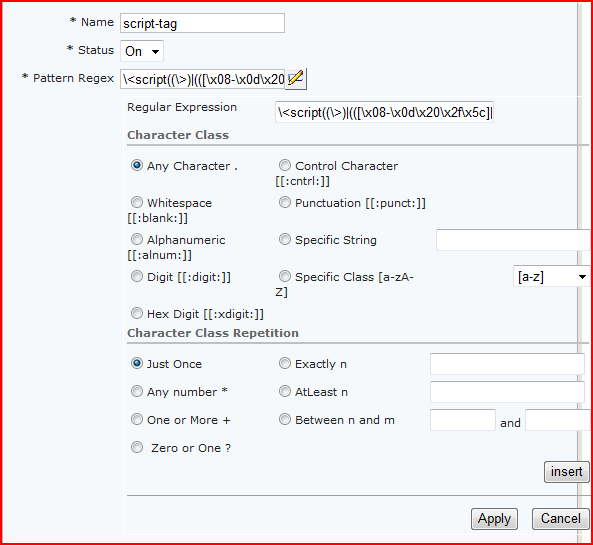
**违反了跨站脚本的详细日志**

## 3.2 跨站脚本攻击防护

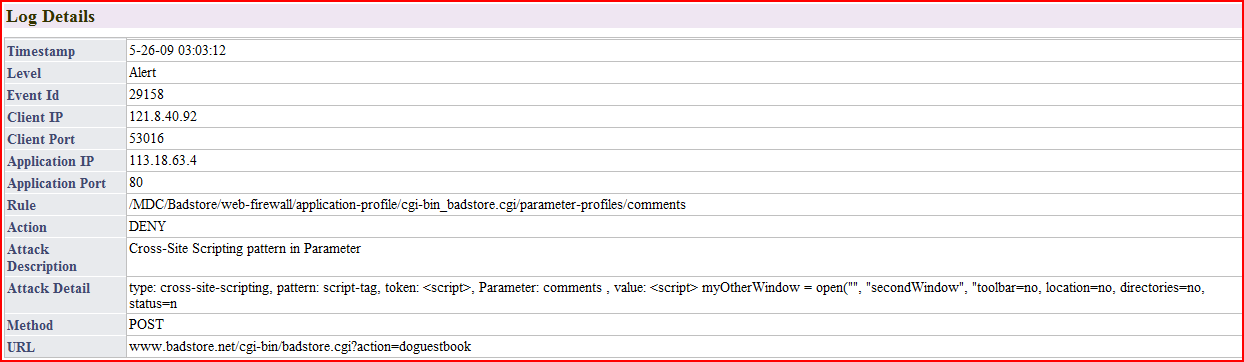
梭子鱼对于跨站脚本攻击的防护主要是对输入参数中的关键字进行设定，对于多个关键字的设定可以采用正则表达式的方式进行逻辑上的组合，从而可以达到逻辑上的全面防护。



在“custom-cross-site-scripting”中，用户可以自定义防护的各种参数以及正则表达式的写法。在默认的“custom-cross-site-scripting”中的正则表达式明确规定，如果输入的信息中带有“\<script>”字样的连接，都将被视为跨站脚本攻击的连接。



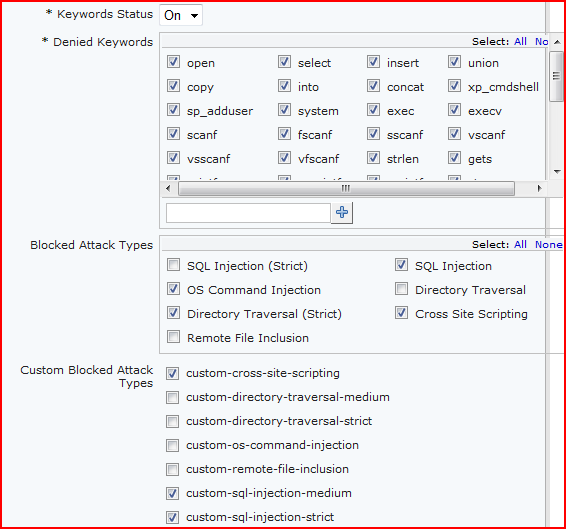
在设置好策略以后，我们再次重复2.2章节中的黑客操作，可以发现此时HTTP的连接被阻断，返回的页面为另外一个报错页面，此页面为梭子鱼默认提供的错误信息页面，这个页面中的内容可以由用户自行定义。

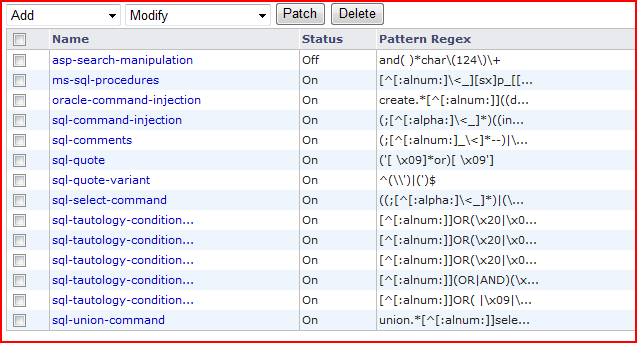


日志中明确了攻击发生的时间、攻击威胁等级、发起攻击的客户端IP地址以及端口、违反的策略规则、攻击描述、攻击的详细信息以及攻击的URL。

## 3.3 SQL注入攻击防护

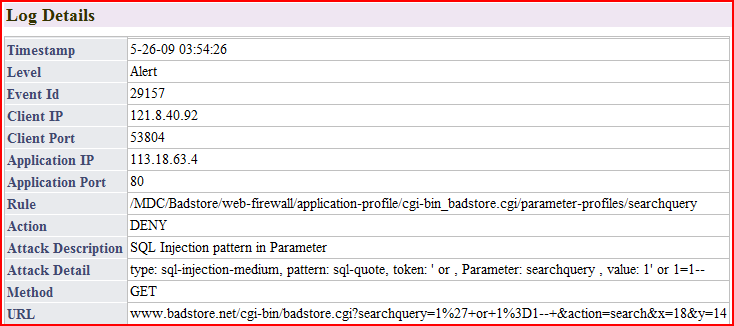
梭子鱼对于SQL攻击的防护主要是对输入参数中的关键字进行判断，对于多个关键字的设定可以采用正则表达式的方式进行逻辑上的组合，从而可以达到逻辑上的全面防护。同时还可以对于SQL语言中一些关键字进行设定，如果在提交的参数内容中包含了带有SQL关键字的连接，此链接将被阻断。



在“custom-sql-injection-medium” 中，用户可以自定义防护的各种参数以及正则表达式的写法。在默认的策略中定义了多种SQL注入攻击的正则表达式，如下图所示：

我们重复2.3中SQL注入攻击的过程，以便获取所有商品的列表。在重复了SQL注入攻击以后，发现当前HTTP的连接被阻断了，并且返回了错误信息页面。

查看梭子鱼的防火墙日志，如下图所示：

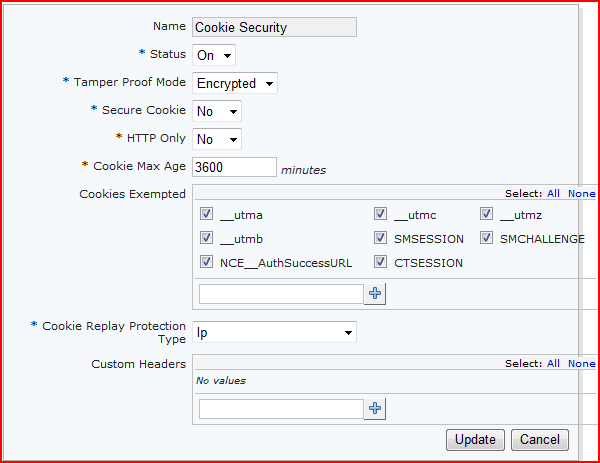


日志中明确了攻击发生的时间、攻击威胁等级、发起攻击的客户端IP地址以及端口、违反的策略规则、攻击描述、攻击的详细信息以及攻击的URL。

这里可以发现，日志中指出攻击的描述为“SQL Injection”，攻击细节是在参数中包含了查询字符串，其值为“1’ or 1=1-- ”。所以此链接违反了“/MDC/Badstore/web-firewall/application-profile/cgi-bin\_badstore.cgi/parameter-profiles/searchquery”规则，所以被阻断。

## 3.4 Cookie窃取防护

梭子鱼对Cookie的防护有2中方法，分别问加密和签名。2种防护的方式都是对Cookie进行重新加密处理，加密以后的内容是黑客无法破解的，只有用梭子鱼的WAF上私有的私钥才能进行解密的操作。



重复2.4中黑客对Cookie进行修改的操作。以下是截取到的Cookie信息：

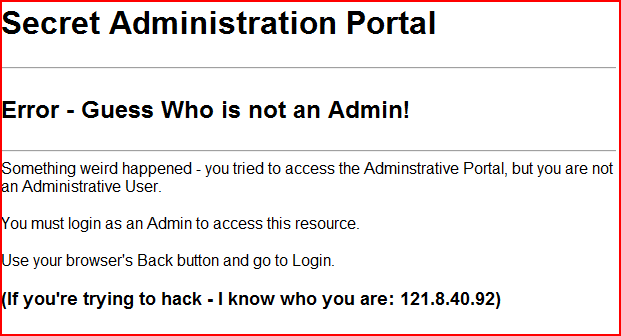
Cookie: NCES\_SSOid=6YHAYppe4jmd3TlPUWpJE2H6QNFQp2D8FqB8cpnABPEfHIlOr6AWOMqt4Qu2UkOVldPMn9n/TLE0ZLu3vjRNb1Ni8Uc8P8W86BksYVU36erXJZntK9sTY6+JB5jExmG6HAbf9nqMPavK6++NHyF5OowdW4WRH8m88zFqOXUgjY4=; NCES\_CartID=k63eCM131CTHW9AZLD7iJ+9ZRII6XBzTEvV/77E1jqiskB0wvIfrXaiQ1gf4C2gtqnL1TEOFwVz4FQho8iU99eJyhGtTn659nSWf8cGcIX5jGExtd2BFG2NeNZxjtleA

红色部分为梭子鱼WAF对Cookie进行签名以后的结果。可以看到，原来的SSOid和CartID都已经被梭子鱼进行了签名处理，变为NCES\_SSOid和NCES\_CartID，其内容都为加密以后的值，原来没有对Cookie进行处理之前的5024的金额，在加密处理以后已经无法在Cookie中找到，从而也就无法对Cookie进行修改。

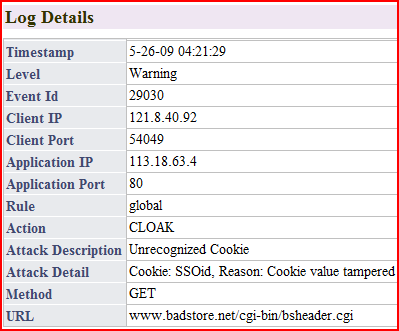
以下Cookie是黑客试图获取管理员权限，并且查看所有交易内容时的Cookie截取信息：

Cookie: NCES\_SSOid=6YHAYppe4jmd3TlPUWpJE2H6QNFQp2D8FqB8cpnABPEfHIlOr6AWOMqt4Qu2UkOVldPMn9n/TLE0ZLu3vjRNb1Ni8Uc8P8W86BksYVU36erXJZntK9sTY6+JB5jExmG6HAbf9nqMPavK6++NHyF5OowdW4WRH8m88zFqOXUgjY4=

可以发现这里的SSOid也被梭子鱼签名为NCES\_SSOid，黑客无法对里面的内容进行Baesed64的反编码操作，也就无法修改用户的身份。如果强行修改Cookie的内容，下图是页面的返回结果：



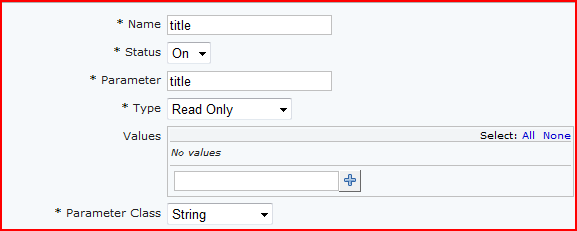
同时在梭子鱼的日志上也可以查看到有Cookie被篡改的警告信息。



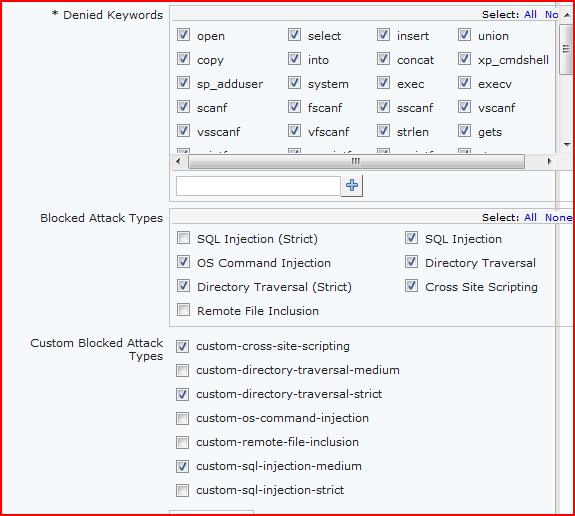
通过上述方法的Cookie防护，梭子鱼可以对Cookie进行全面的防护，能够最大限度的保证Cookie的安全传输。

## 3.5 页面篡改防护

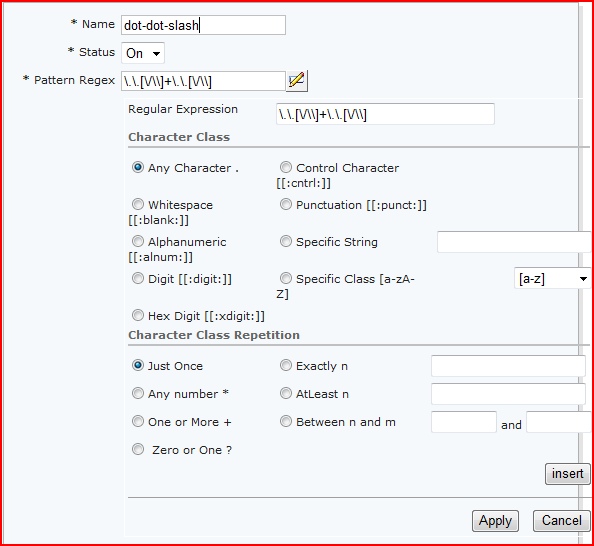
梭子鱼对页面篡改的防护有多种方式，例如可以手动指定参数的性质，例如可以指定页面中的title参数为“Read Only”，则黑客就无法对此参数进行修改的操作。



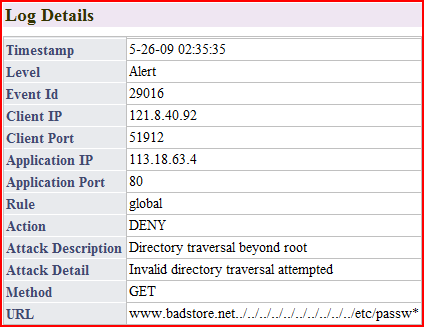
或者指定用户能够访问的目录等级，对于遍历目录的攻击，梭子鱼也可以做到全面防护。



在“custom-directory-traversal-medium”中，用户可以自定义正则表达式。在“dot-dot-slash”规则中，定义了不允许用户访问“../”或者更高级别的目录。



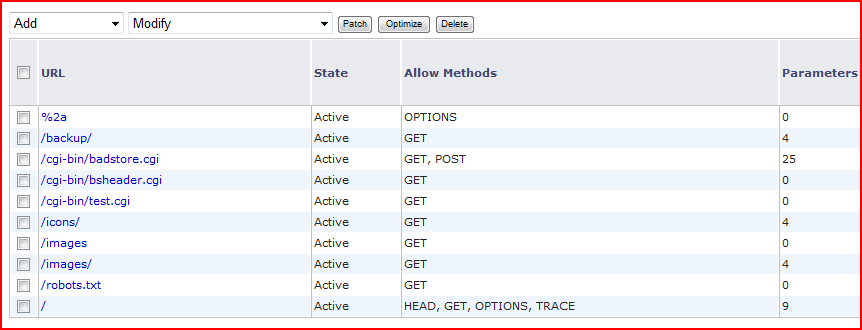
例如在使用nikto扫描网站的时候，会尝试查找存放密码文件的地方。从日志中可以看到，梭子鱼WAF使用“global”规则对此攻击进行了“DENY”的操作。



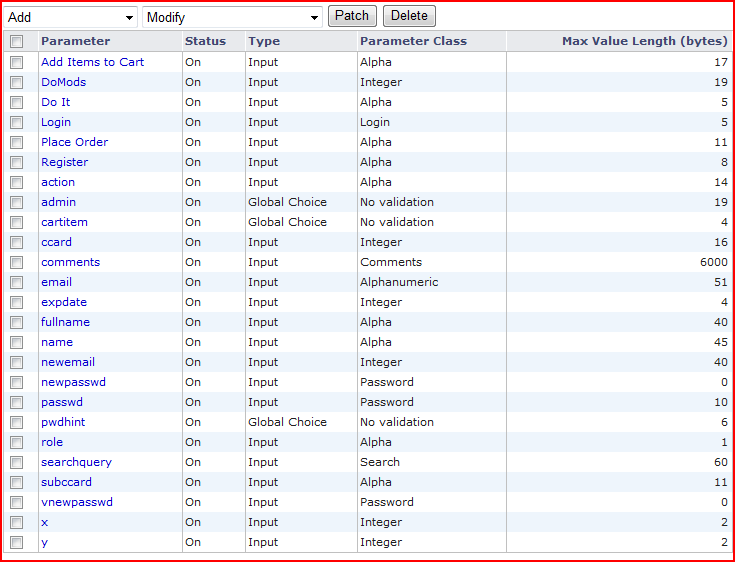
## 3.6 深度内容防护

梭子鱼WAF具有自动建模的功能。此功能开启以后可以将Web服务器上所有用户访问过的URL记录在梭子鱼上，同时记录该URL下所有的参数，以及参数的类型、长度等信息。

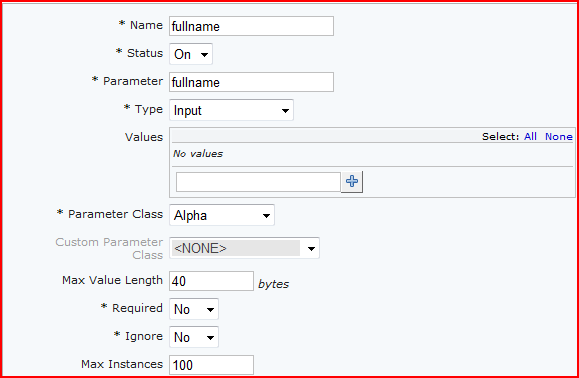
下图为自动建模以后的结果：

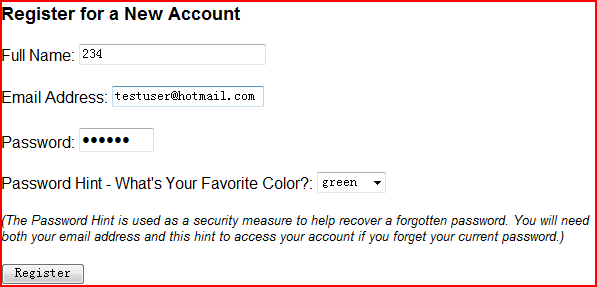


并且可以自动学习每一条URL下所有的参数选项，其中包含参数的类型、参数的分类、以及参数的最大长度：

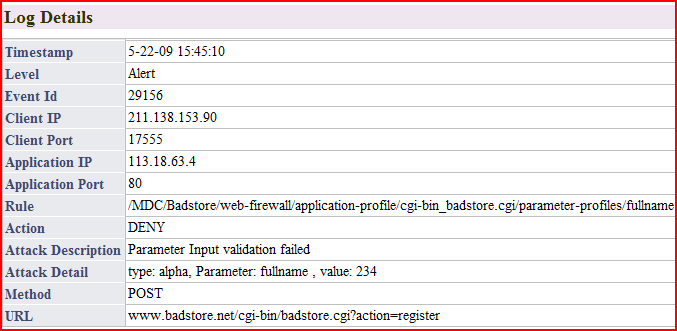


例如在fullname参数中指定了参数的类型为“Input”，并且参数的分类为“Alpha”，如果输入了与fullname格式不相匹配的内容后，可以在日志上查看到如下信息：





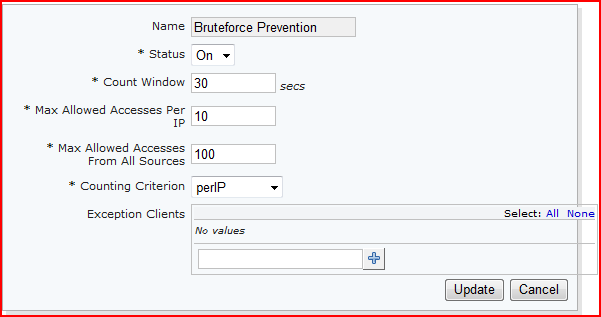
**在Full Name中输入234**

****

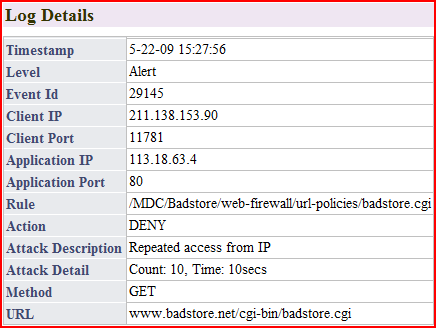
从日志中可以看到，攻击的细节为输入的fullname值为234，但是参数的类型为“alpha”。所以梭子鱼WAF可以自动的判断输入参数的类型，然后来判断这个连接是否为正常连接。

## 3.7 应用层DDOS防护（CC攻击）

应用层的DDOS防护指的是黑客利用大量的HTTP请求来耗尽服务器的资源。梭子鱼可以针对所有的URL或者是个别对数据库服务器操作频繁的URL进行设置。在单位时间内设置能否访问此页面的最大连接次数。



上图的配置为设置www.badstore.net/cgi-bin/badstore.cgi，在30秒内，每个IP地址只能对其进行10次的访问。超过了10此以后，连接将被中断。并且可以在日志上查看到具体的信息。



# 第四章 测试总结

在福建省移动总公司的大力配合下，梭子鱼（中国）公司完成了Web应用层防火墙的测试工作。通过为期2天的测试，移动公司了解了梭子鱼Web应用防火墙的功能，以及防御各种攻击的原理及方法。完成了测试初期制定的测试计划。

梭子鱼Web应用防火墙可以阻挡绝大部分的互联网上的Web应用攻击，可以大大提升福建省移动公司MDC机房的安全等级，并且方便机房管理员同时安全管理多个网站。可以满足福建省移动公司对Web应用层安全的严格要求。