Web 安全实战演练 1-4

--- 环境搭建+密码字典破解测试+SQL 注入测试+XSS 测试

作者：@_U2_

Web 安全实战演练（1）- 学习环境搭建

入侵网站在任何国家都是违法行为，因此学习 Web 安全，不能以任何第三方开展正常业务的网站为测试目标。

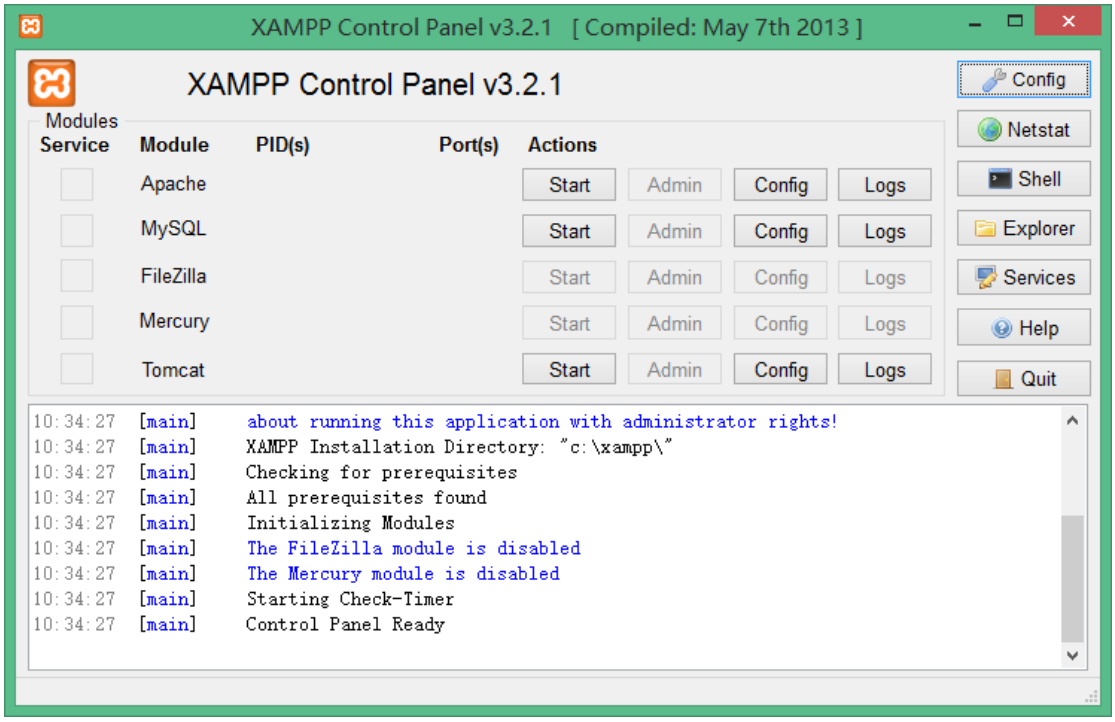
为了解决这个问题，首先需要自行搭建一套测试环境。

如果您不擅长自行搭建服务器，下面推荐一种简单、快速的测试环境（XAMPP）。

操作系统：Windows 系列均可（Windows 7, 8）

所需软件：XAMPP for Windows，下载地址： https://www.apachefriends.org/zh_cn/index.html

软件说明：XAMPP for Windows 内置了 Apache、PHP、Tomcat、MySQL 等组件，能够满足常见的 Web 安全测试需求。安装目录在 c:\xampp（建议不要修改）。



安装后通过 XAMPP Control Panel 管理各组件的启动，启动后可以看到组件列表（其中 FileZilla 用于 FTP 服务器，Mercury 用于邮件服务器，本文暂不涉及）。

以常见的 PHP+MySQL 应用（如 DVWA）为例，启动上图中的 Apache 和 MySQL 即可（点击"Start"按钮）。通过 http://127.0.0.1 访问。

如果是 JSP+MySQL 应用（如 WAVSEP），启动 Tomcat 和 MySQL 即可，通过 http://127.0.0.1:8080 访问。

启动后，默认 MySQL 是空口令，可以通过

C:\xampp\mysql\bin\ mysqladmin -u root password 123456

命令将口令修改为 123456 （此口令为弱密码，不推荐采用，仅为示例，请修改为自己可以记住的复杂密码）。

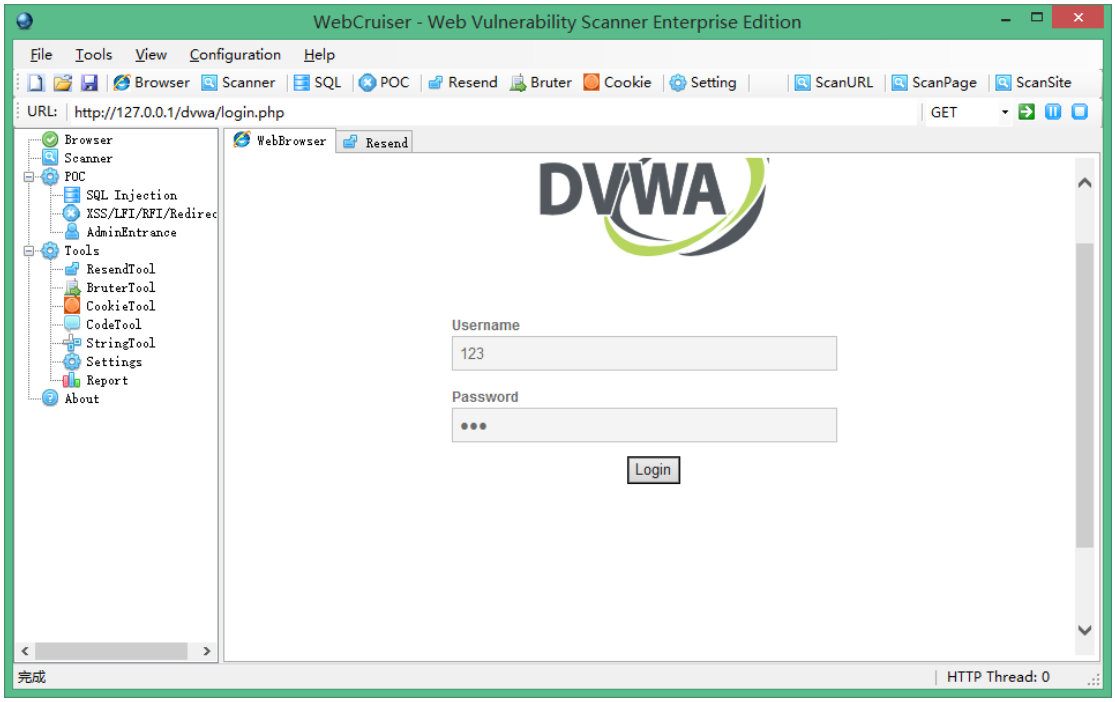
安装安全演练测试环境：DVWA

从 https://github.com/RandomStorm/DVWA 下载 zip 格式的压缩包，解压到

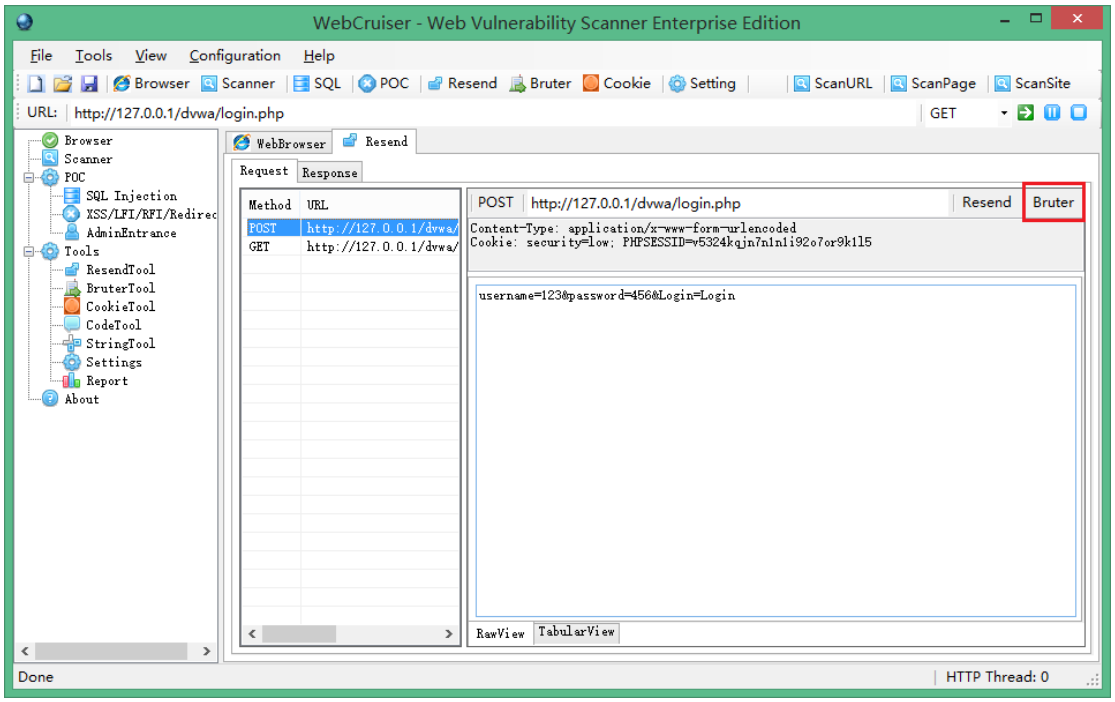
C:\xampp\htdocs\dvwa ，
修改配置文件 C:\xampp\htdocs\dvwa\config\config.inc.php:
$_DVWA['db_server'] = 'localhost';
$_DVWA['db_database'] = 'dvwa';
$_DVWA['db_user'] = 'root';
$_DVWA['db_password'] = '123456';
$_DVWA['default_security_level'] = "low";
打开 http://127.0.0.1/dvwa/setup.php ，
点击"Create/Reset Database"，即可安装完成。
这时进入首页，自动转入 http://127.0.0.1/DVWA/login.php

这时可以开始研究怎么进入这个系统了。

Web 安全实战演练（2）- DVWA 登录爆破
第一关，考察的是怎么登录进入？
一般来说，有这样几种方式：
第一种，通过网络搜索，获知其默认用户名和口令，直接登入；
第二种，可尝试 SQL 注入（如尝试用户名 admin';-- 或修改 input 为 textarea 然后多行注入，密码随便输入）；
这里采用第三种，就是挂密码字典进行爆破。
俗话说，预先善其事，必先利其器，需要找到一款好用的工具。
这里我们采用 WebCruiser Web Vulnerability Scanner 3 （http://www.janusec.com/ ）：
首先，在 WebCruiser 界面随便输入用户名和密码，提交。



切换到重放（Resend）界面：

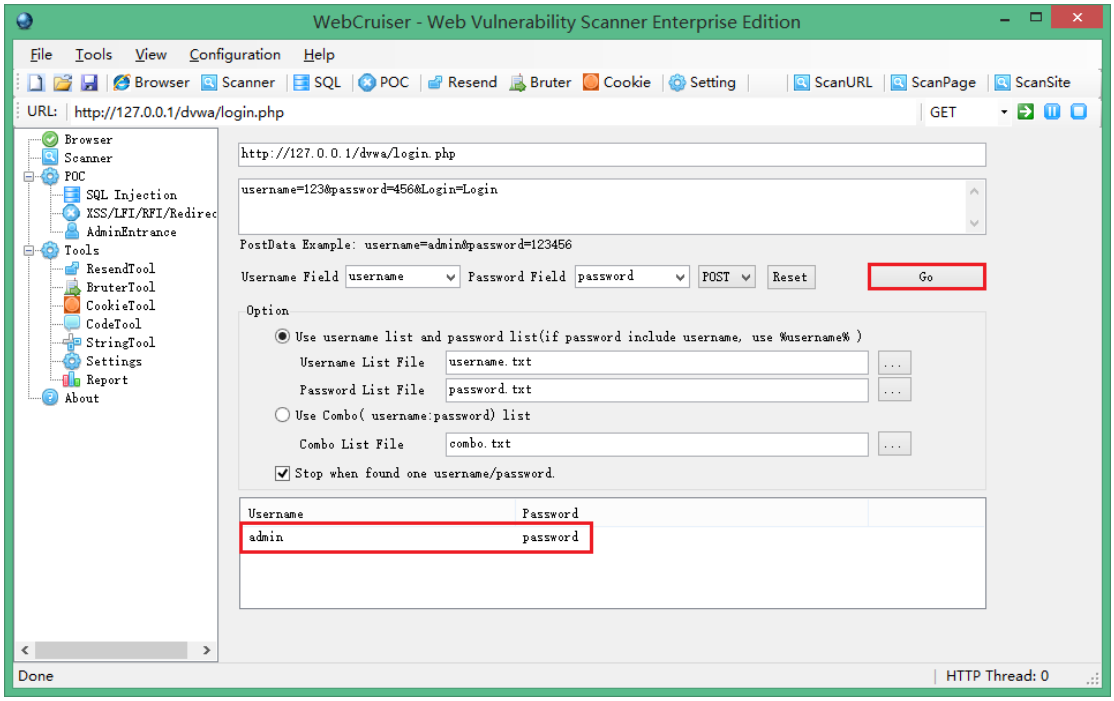


可以看到刚才提交的请求内容，注意 URL 的右边有个“Bruter”按钮，点击它，会切换到 Bruter 界面。

可以看到，WebCruiser 已经自动识别出了用户名和密码字段。

字典文件位于 WebCruiser 的同级目录下，可以打开查看并进行自定义修改。

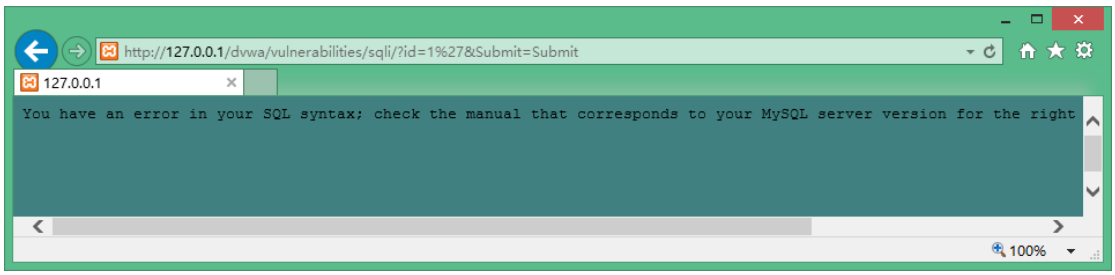
其中 username.txt 和 password.txt 一起使用，combo.txt 单独使用。



点击“Go”启动猜解，在下方的结果窗口可以看到，已猜出密码。
使用该密码登录。

Web 安全实战演练（3）- SQL 注入
DVWA 菜单栏点击“SQL Injection”切换到 SQL 注入测试。
先随便输入一个 1，测试：

再输入 1’ 尝试：



因为单引号不匹配，导致数据库抛出异常。

至此，初步判断这里可能存在 SQL 注入漏洞。

继续尝试 1 and 1=1 和 1 and 1=2

发现跟预期的结果不一样，整型 SQL 注入排除。

继续用 1' and '1'='1 和 1' and '1'='2 尝试字符型

输入 1’ and ‘1’=’2 时没有查询数据返回。

至此，可以判定：此处存在字符型 SQL 注入。

【总结】SQL 注入漏洞判定准则：

假设原始请求返回为 Response0，

追加真逻辑的返回为 Response1，

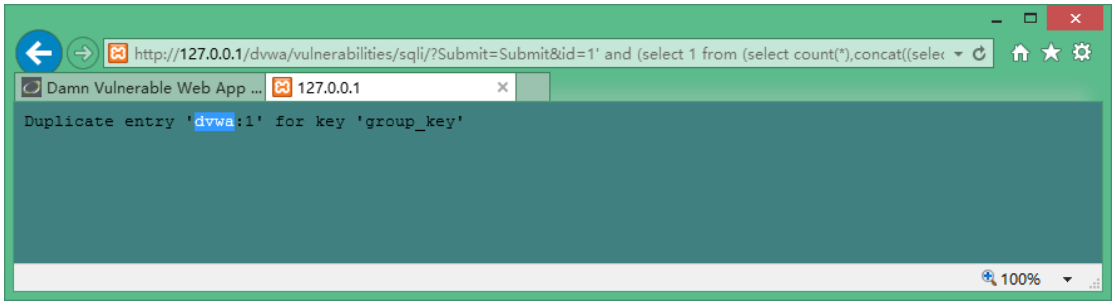
追加假逻辑的返回为 Response2，

如果 Response0 等于 Response1，并且 Response1 不等于 Response2，则判定漏洞存在。

必然有人开始质疑，何以见得？能否拿个数据看看？

提交：http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select 1 from

(select count(*),concat((select database()),0x3a,floor(rand(0)*2)) x from information_schema.tables group by x)a)%23



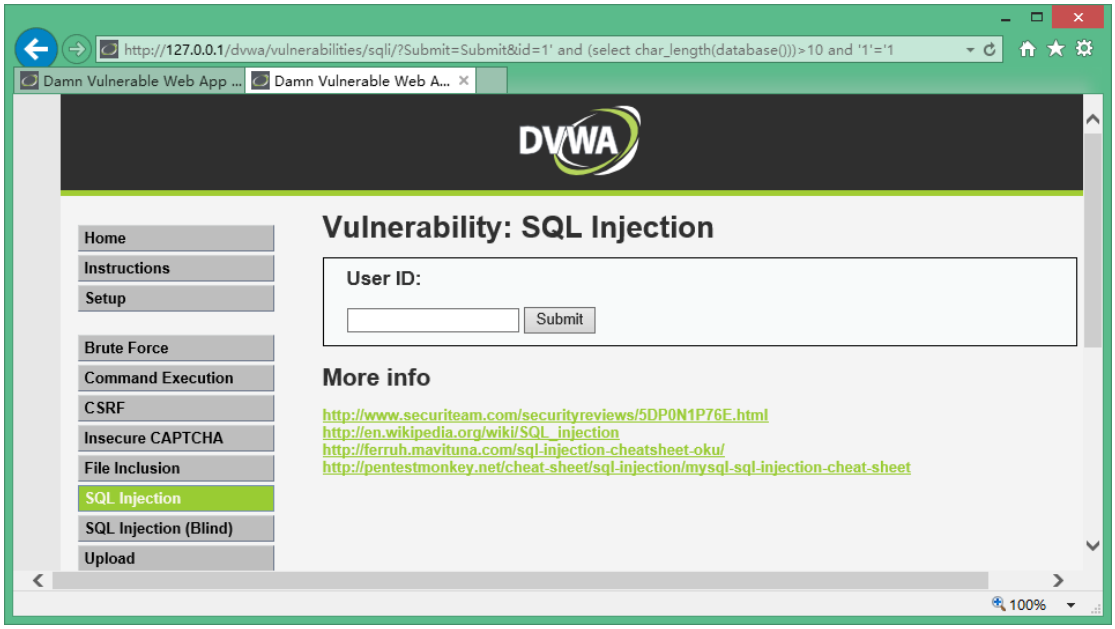
可以看到，数据库名称 dvwa 已经显示在网页上了。

这句语法稍微复杂了点，实际上是故意构造的会造成 MySQL 内部异常的语句（两次 rand 运算），这种直接报错并展示异常信息的方式可以快速的获取数据。

简单一点的，

判断数据库长度是否大于 10：

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select char_length(database()))>10 and '1'='1



不对。判断数据库名称长度小于 10 试试：

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select char_length(database()))<10 and '1'='1

这回对了，继续猜，直到：

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select char_length(database()))=4 and '1'='1

此时，得到数据库名称长度为 4.

继续刚才的办法：

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select ord(substr(database(),1,1)))=100 and %271%27=%271

得到数据库名称第一位的 ASCII 码为 100，对应字母 d 。

以此类推，

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select ord(substr(database(),2,1)))=118 and %271%27=%271 ，得到第二位字母为 v 。

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select ord(substr(database(),3,1)))=119 and %271%27=%271 ，得到第三位字母为 w 。

http://127.0.0.1/dvwa/vulnerabilities/sqli/?Submit=Submit&id=1' and (select ord(substr(database(),4,1)))=97 and %271%27=%271 ，得到第四位字母为 a 。

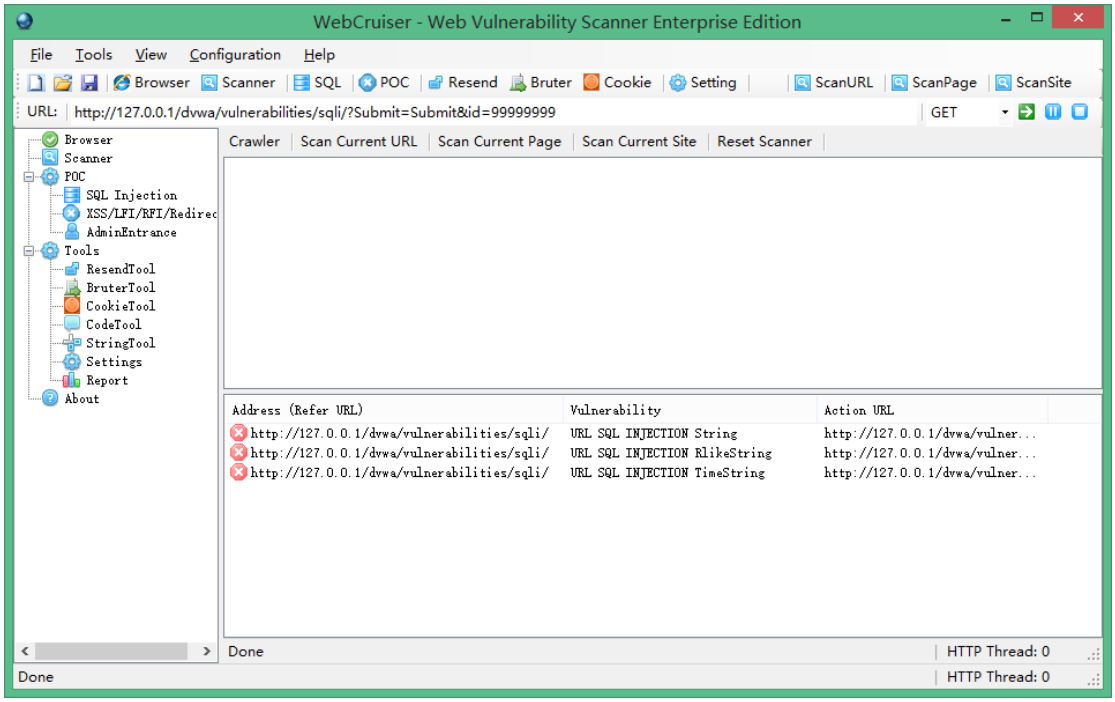
至此，得到完整的数据库名称 dvwa 。

这种逐位猜解的方式，称之为盲注。

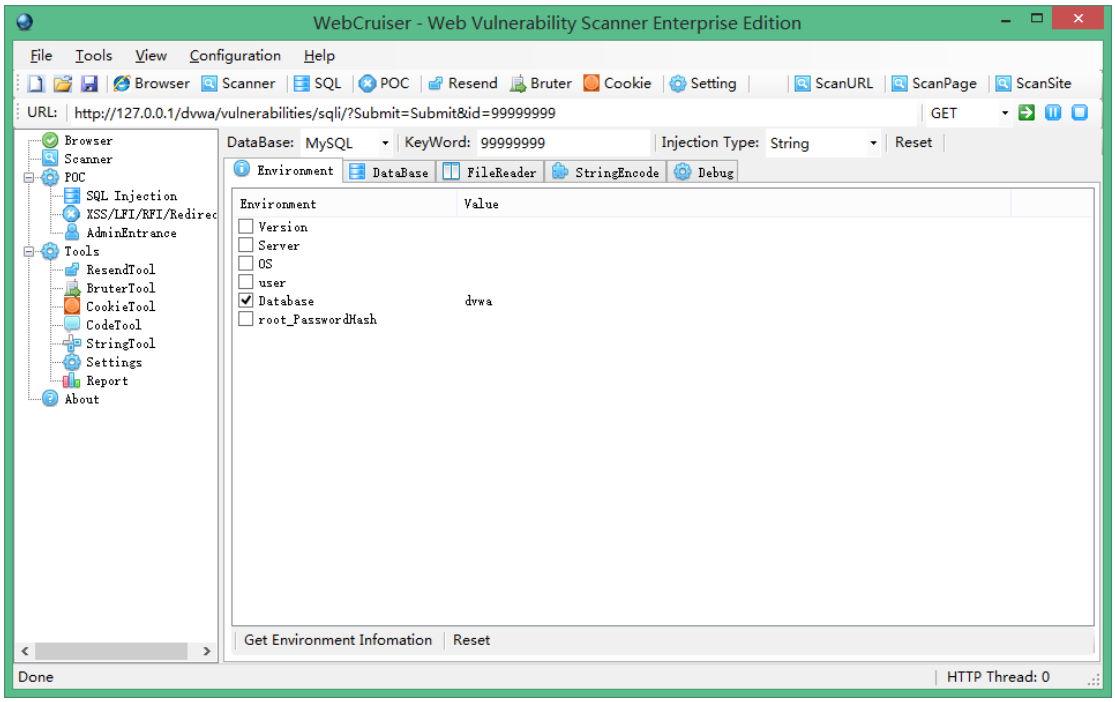
是不是效率太低了?

没错。我们完全可以利用工具来完成这个动作，况且各种类型的注入方式种类繁多，手工方式无法覆盖每一种场景。

以 WebCruiser 为例，浏览到这个页面后，点击 WebCruiser 的右上角“Scan URL”，结果:



发现 SQL 注入漏洞。在漏洞项上面单击右键，选择“SQL INJECTION POC”即可发起 SQL 注入利用工具，继续点击”Get Environment Information”，获取环境信息。

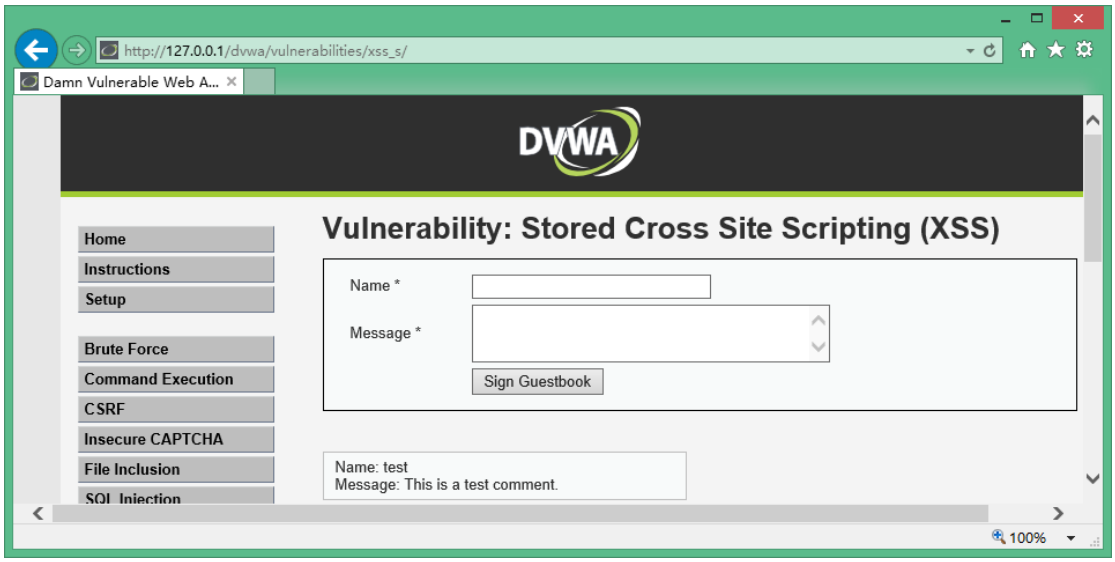


Web 安全实战演练(4)-XSS 测试

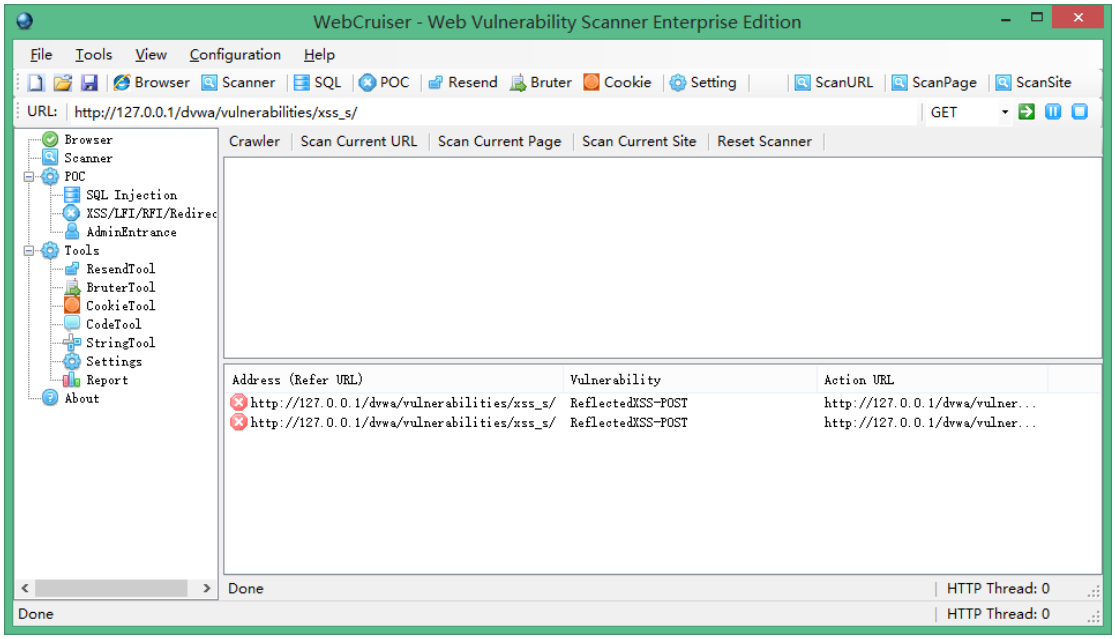
从菜单切换到 XSS

http://127.0.0.1/dvwa/vulnerabilities/xss_s/

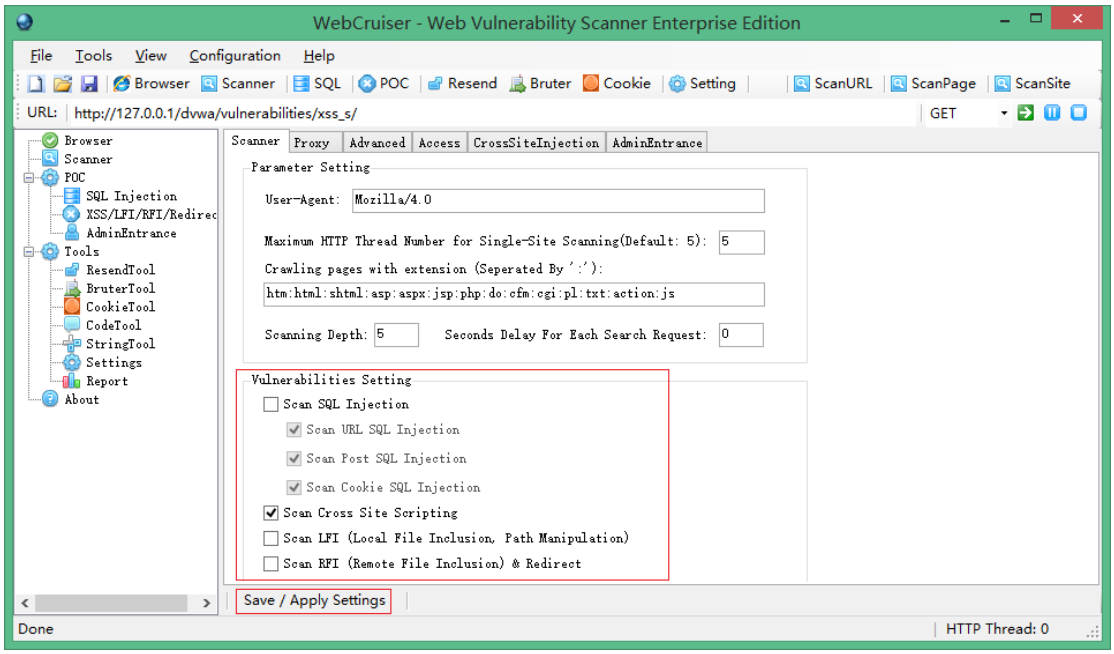
类似这种提交内容或评论的地方，标题或内容均可能构成 XSS 漏洞。



直接输入含有脚本的内容，如 testinput<img src=0 onerror="alert(123456)"> 。
使用扫描器，检出两个 XSS 漏洞。



这里，要提一下，为了提高效率，WebCruiser 可以只扫描指定的漏洞类型（设置中勾选），可以只扫描指定的 URL（ScanURL 按钮发起）。



使用扫描器，可以提高效率，但是深入研究 Web 安全，不能依赖扫描器，需要对每一种漏洞类型的每一个具体的利用形式深入了解(没有捷径可走，只有采取笨办法，遍历所有漏洞，才能获得质的提高，敬请关注后续内容: Web 安全实战演练--遍历高危漏洞)。