# Project 2 Web Security

## 实验环境搭建

Project 2 主要探究对web的攻击，本次试验共有6个部分。Project 2中攻击的是一个提供电子货币服务网站--bitbar（使用ruby 2.4和rails 5.0.2实现）。

接下来介绍bitbar网站的搭建。

* 1. 服务器环境搭建
* 安装Ruby2.4和rails 5.0.2

<http://gorails.com/setup/ubuntu/16.04>

按照上面网址的布置安装Ruby 2.4和rail 5.0.2（安装正确的版本非常重要）。可以跳过MySQL和PostgreSQL部分。

* 下载实验提供的project 2源码
* 重定位到/bitbar目录下，执行bundle install
* 开启服务器 （rails server）

以上的步骤执行完后，你可以在<http://localhost:3000>上访问bitbar。如果要关闭服务器，可以在终端上执行Ctrl+C。在攻击的过程中，你不允许对网站的源码进行修改

## 实验内容介绍

Attack 1： Warn-up exercise: Cookie Theft

* 开始网址

<http://localhost:3000/profile?username=>

* 评分员将提前以user1的身份登录bitbar，然后打开以上的开始网址
* 你的目标是偷取user1的会话cookie并且将cookie发送到

<http://localhost:3000/steal_cookie?cookie=...cookie>\_data\_here...

* 你可以在以下网址上查看最近被偷取的cookie

<http://localhost:3000/view_stolen_cookie>

* 请将你的答案写在warmup.txt中
* 提示：尝试添加一些随机字符串到开始网址后，观察这些随机字符会如何影响网页

Attack 2: Session hijacking with Cookies

* 在本次试验中，你将会获得attacker的身份：用户名attacker，密码attacker。你的目的是伪装成用户user1登录系统
* 你的答案是一个脚本。当这个脚本在JavaScript console中执行时，bitbar将误认为你是以user1。请将这个脚本写到a.sh中
* 本次试验中，你可以使用Mechanize。Mechanize是一个Ruby的库函数， 它被用于与web应用实现自动化交互。在本次试验中，你必须要保存服务器发送的所有的cookie值。
* 提示：网站是如何保存会话的？网站是如何验证用户当前是否登录？网站是如何验证cookie的真实性的？

Attack 3: Cross-site Request Forgery

* 你的答案是一个名字为b.html的html文件。评分人将用浏览器打开b.html
* 在打开b.html前，评分人将提前使用user1的身份登录到bitbar
* 打开b.html后，10个bitbar将从user1的账户转到attacker的账户，当转账结束时，页面重定向到[www.baidu.com](http://www.baidu.com)。
* 你可以在<http://localhost:3000/view_users> 查看用户列表以及每个用户拥有的bitbar
* 在攻击的过程中，浏览器的网址中不能出现localhost:3000

Attack 4: Cross-site request forgery with user assistance

* 你的答案是一个或者两个html页面，命名为bp.html，bp2.html(可选)。评分员会在浏览器中打开bp.html
* 在打开bp.html前，评分员已经用user1的身份登录到系统中
* 评分员将于bp.html页面进行交互，因此bp.html的回应要合理。也就是说，如果在页面上有一个表格或者有一个按钮，并且在页面上有一些提示要求评分员进行一些操作，评分员将会依照这些提示执行。
* 在评分员与bp.html页面进行交互后，10 bitbars将会从评分员的账户转到attacker的账户。当这个转账操作执行完成后，页面将重定向到[www.baidu.com](http://www.baidu.com)
* 你的攻击必须要在于用户互动的前提下执行（不要再一次进行一次CSRF攻击）。特别的要注意的是，你的攻击要针对的网址是<http://localhost:3000/super_secure_transfer>或者 <http://localhost:3000/super_secure_post_transfer>。这两个网址做了一些CSRF攻击的防护。在攻击的过程中，你不能直接与<http://localhost:3000/transfer>或者<http://localhost:3000/post_transfer>进行交互。
* 在你的攻击过程中，需要隐藏你的页面正从<http://localhost:3000>上下载内容的事实。

Attack 5: Little Bobby Tables (aka SQL Injection)

* 你的答案是一个恶意的用户名。这个恶意的用户名允许你删除一个你不具有访问权限账户。
* 评分员将使用你提供的恶意用户名新建一个账户。并在“close”页面上确认删除该账户
* 作为结果，新建的账户以及user3的账户将会被删除。其他的账户不变
* 你可以在<http://localhost:3000/view_users>页面上查看所用的用户
* 如果数据库在测试攻击的过程中被破坏了，你可以停止Rails然后使用rake db:reset命令是数据库复原。
* 将你的最终答案写在d.txt中
* 提示：SQL注入；WHERE子句

Attack 6: Profile Worm

* 你的答案是一个用户的profile（简况）。当其他用户阅读这个profile时，1个bitbar将会从当前账户转到attacker的账户，并且将当前用户的profile修改成该profile。因此，如果attacker将他的profile修改成你的答案，以下情况会发生：
  + 如果user1浏览了attacker的profile，那么1 bitbar将从user1的账户转到attacker的账户，user1的profile修改成你答案中的profile
  + 之后，如果user2浏览了user1的profile，那么1 bitbar将从user2的账户转到attacker的账户，user2的profile也被替换成你答案中profile

因此，你的profile worm将会很快扩散到全部的用户账户中

* 将你的恶意的profile写在d.txt中
* 评分过程：评分员将会将你提供的恶意profile复制到attacker的profile上。然后，评分者将使用多个账户浏览attacker的profile。检查是否正常进行转账以及profile的复制
* 转账和profile复制的过程应该具有合理的速度。在这个过程中，评分员不会点击任何地方。
* 在转账和profile的赋值过程中，浏览器的地址栏需要始终停留在<http://localhost:3000/profile?username=x> ，其中x是profile被浏览的用户名。
* 不会出现当前账户没有钱可以转的情况
* 提示：MySpace vulnerability