

---

# INTRODUCTION TO WEB APPLICATION PENETRATION TESTING

*By: Rana Khalil*

# WHO AM I?

- University of Ottawa student – Master of Computer Science
- Thesis: Comparative Analysis of Open-source Web Application Vulnerability Scanners
- Previous work experience include: Software developer, Tester, Ransomware researcher, Security Analyst.
- Aspiring Ethical Hacker!
- Removed my wisdom tooth a week ago 😞

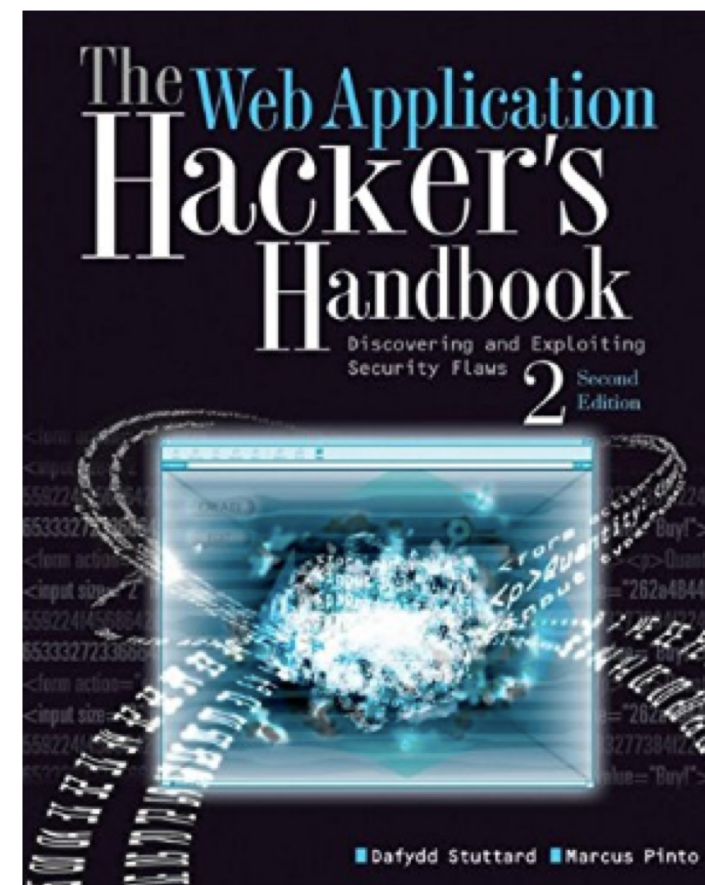


# OUTLINE

- Introduction to web application penetration testing
- Software setup
- Mapping and analyzing the application
- Bypassing client-side controls
- Attacking authentication
- Attacking session management
- Attacking data stores

# THE WEB APPLICATION HACKER'S HANDBOOK

- Bible of web application penetration testing.
- Good mix of theory and practice.
- Most of the examples in the slides are from this book.



# TOOLS NEEDED FOR WORKSHOP

- Burp Community Edition
- OWASP ZAP
- VirtualBox
- OWASP BWA Project
- FoxyProxy Standard
- Modern browser (preferably Firefox)



**Note:** You should have these tools already installed. An email was sent to registered participants prior to workshop.

---

# INTRODUCTION

SECTION I

# WHY THE WEB?

- Let's look at some statistics:
  - Over 3.9 billion internet users in the world
  - Over 1.9 billion websites online
- We use websites for everything: e-commerce, online banking to social networking, social media, etc.
- Web security has become a major concern for businesses.
- Recent example: Equifax. The breach exposed the personal information of 143 million US users and an estimated 100,000 Canadian users.
- According to Trustwave's 2018 Global Security Report:
  - 100% of the web applications scanned by Trustwave displayed at least one vulnerability.
  - Median number of 11 vulnerabilities detected per application.

# HOW TO SECURE A WEB APPLICATION?

Combination of techniques are used:

- Secure coding practices
- Web application firewalls
- Static code analysis
- **Web application penetration testing** ←
- Etc.



# WHAT IS WEB APP PEN TESTING?

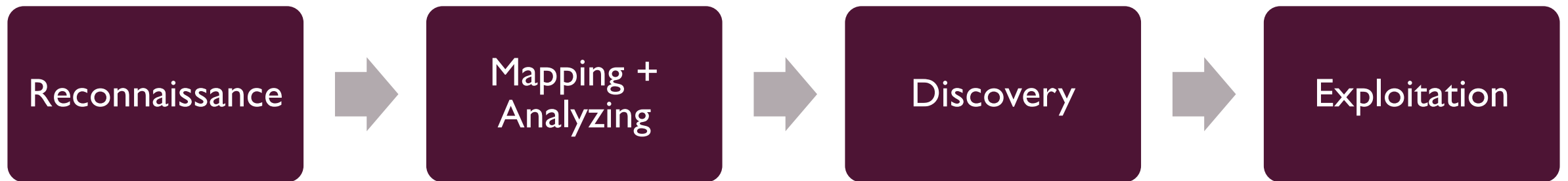
- Combination of manual and automated tests to identify vulnerabilities, security flaws and/or threats in a web application.
- Categorized into three types:
  - White box: Tester has complete access and in-depth knowledge of the system. Access to source code is usually given.
  - Black box: Tester is given little to no information about the system. Just the URL of the application is usually given.
  - Grey box: Combination of white box and black box penetration testing. Limited information and access is given to the tester.
- Several methodologies and guidelines: OWASP, PTES, PCI DSS, etc.
- Most important thing to keep in mind: you need permission to perform a security test!

# WHAT IS OWASP?

- Stands for Open Web Application Security Project
- International open source community “dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted”.
- Contains widely used and popular tools such as the ZAP.
- OWASP TOP 10 Project



# ABSTRACT PEN TESTING METHODOLOGY





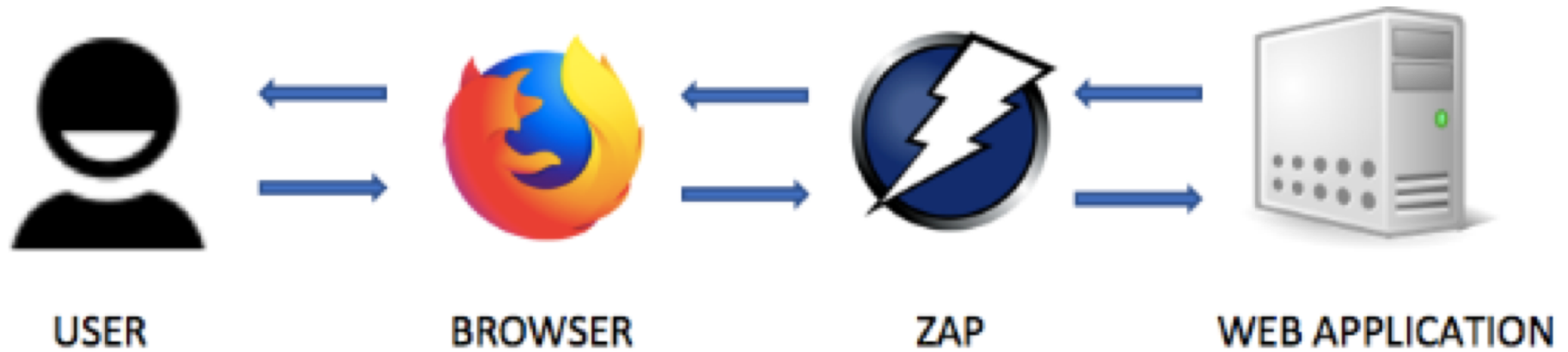
# SOFTWARE SETUP

SECTION 2

# SOFTWARE SETUP

1. Download VirtualBox and OWASP BWA Project virtual machine.
2. Install OWASP BWA Project vm on VirtualBox.
3. Download Firefox and FoxyProxy add-on.
4. Download Burp Suite Community Edition and OWASP ZAP.
5. Configure Burp and ZAP in FoxyProxy.

# HOW DOES A PROXY WORK?



# SOFTWARE SETUP

*Setup Demonstration*



# MAPPING AND ANALYZING THE APPLICATION

SECTION 3



# MAPPING THE APPLICATION

- Explore the visible content
- Review public resources
- Identify any hidden content

# ANALYZING THE APPLICATION

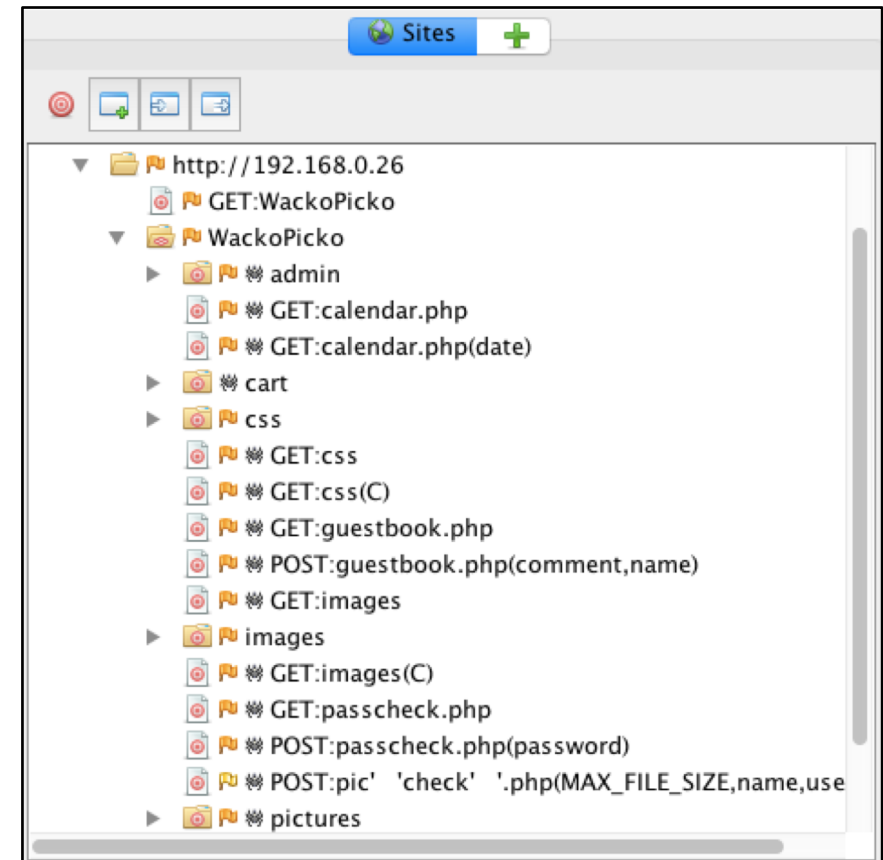
- Identify functionality
- Identify data entry points
- Identify the technologies used
- Map the attack surface

# EXERCISE #1: WACKOPICKO

## Web Spidering

Your goal is to use ZAP spider to crawl the application.

1. Configure browser to work with ZAP
2. Load the application on configured browser
3. Include root application URL in context.
4. Right click on the root URL in the Sites tree map > select Attack > select Spider.
5. Right click on the root URL in the Sites tree map > select Attack > select Ajax Spider.



# EXERCISE #1: WACKOPICKO

*Solution demonstration using ZAP and Burp Suite.*

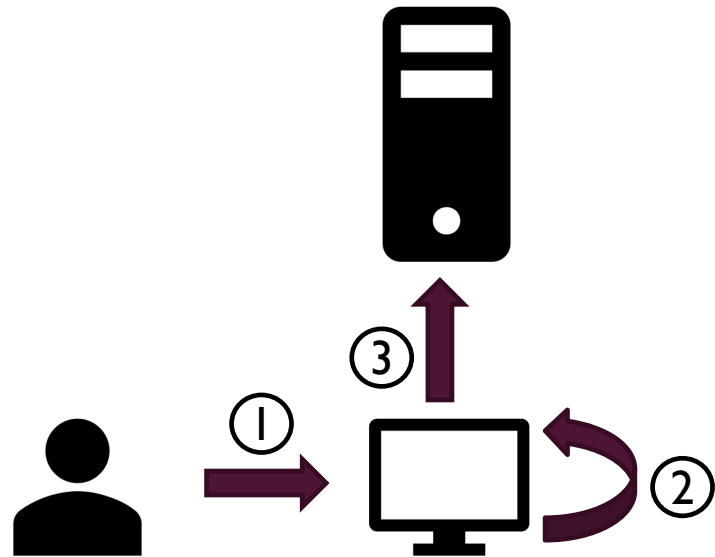
---

# BYPASSING CLIENT-SIDE CONTROLS

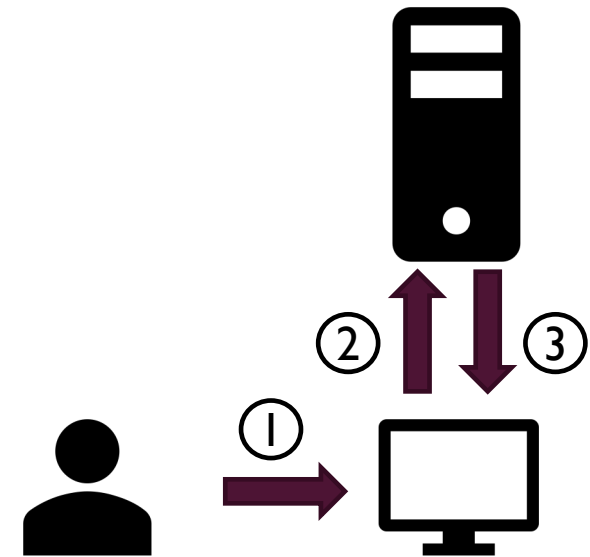
SECTION 4

# CLIENT-SIDE VS SERVER-SIDE VALIDATION

## Client-side



## Server-side



# BYPASSING CLIENT-SIDE CONTROLS

- Allowing clients to submit arbitrary input is a core security problem in web applications.
  - Users have full control of everything submitted from the client.
  - Can cause a range of problems including corrupting data stores, allowing unauthorized access to users and buffer overflows.
- In general, there are two ways client-side controls are used to restrict user input:
  - Transmitting data via the client using mechanisms that “prevent” user interaction. Examples include hidden form fields, disabled elements, referrer header, URL parameters, etc.
  - Controlling user input using measures that “restrict” user input. Examples include HTML form features, client-side scripts, etc.

# TRANSMITTING DATA VIA THE CLIENT

## ➤ Example #1: Hidden Form Field

Please enter the required quantity:

**Product:** iPhone Ultimate  
**Price:** 449  
**Quantity:**  (Maximum quantity is 50)

Code

```
<form method="post" action="Shop.aspx?prod=1">
Product: iPhone 5 <br/>
Price: 449 <br/>
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)
<br/>
<input type="hidden" name="price" value="449">
<input type="submit" value="Buy">
</form>
```

Request

```
POST /shop/28/Shop.aspx?prod=1 HTTP/1.1
Host: mdsec.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
```

```
quantity=1&price=449
```



# TRANSMITTING DATA VIA THE CLIENT

## ➤ Example #2: HTTP Cookies

### ■ Response:

```
HTTP/1.1 200 OK
Set-Cookie: DiscountAgreed=25
Content-Length: 1530
...
```

### Request:

```
POST /shop/92/Shop.aspx?prod=3 HTTP/1.1
Host: mdsec.net
Cookie: DiscountAgreed=25
Content-Length: 10

quantity=1
```

# EXERCISE #2: WEBGOAT

## Exploit Hidden Fields

Try to purchase the HDTV for less than the purchase price, if you have not done so already.

### Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
56 inch HDTV (model KTV-551)	2999.99	<input type="text" value="1"/>	\$2999.99

The total charged to your credit card:

\$2999.99

# EXERCISE #2: WEBGOAT

*Solution Demonstration*

# RESTRICTING USER DATA

## ➤ Example #3: Length Limits

Please enter the required quantity:

**Product:** iPhone Ultimate  
**Price:** 449  
**Quantity:**

```
<form method="post" action="Shop.aspx?prod=1">  
Product: iPhone 5 <br/>  
Price: 449 <br/>  
Quantity: <input type="text" name="quantity" maxlength="1"> <br/>  
<input type="hidden" name="price" value="449">  
<input type="submit" value="Buy">  
</form>
```

# RESTRICTING USER DATA

## ➤ Example #4: Disabled Elements

Please enter the required quantity:

**Product:** Blackberry Rude

**Price:**

**Quantity:**  (Maximum quantity is 50)

```
<form method="post" action="Shop.aspx?prod=5">  
Product: Blackberry Rude <br/>  
Price: <input type="text" disabled="true" name="price" value="299">  
<br/>  
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)  
<br/>  
<input type="submit" value="Buy">  
</form>
```

# EXERCISE #3: WEBGOAT

## Bypass HTML Field Restrictions

You must submit invalid values for all six fields in one form submission.

Select field with two possible values:

Radio button with two possible values:

 foo bar

Checkbox:

 checkbox

Input field restricted to 5 characters:

Disabled input field:

Submit button:

# EXERCISE #3: WEBGOAT

*Solution Demonstration*

# RESTRICTING USER DATA

## ➤ Example #5: Script-Based Validation

```
<form method="post" action="Shop.aspx?prod=2" onsubmit="return
validateForm(this)">
Product: Samsung Multiverse <br/>
Price: 399 <br/>
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)
<br/>
<input type="submit" value="Buy">
</form>
```

```
<script>function validateForm(theForm)
{
    var isInteger = /^\d+$/;
    var valid = isInteger.test(quantity) &&
        quantity > 0 && quantity <= 50;
    if (!valid)
        alert('Please enter a valid quantity');
    return valid;
}
</script>
```



# EXERCISE #4: WEBGOAT

## Bypass Client Side JavaScript Validation

You must break all 7 validators at the same time.

Field1: exactly three lowercase characters(`^[a-z]{3}$`)

Field2: exactly three digits(`^[0-9]{3}$`)

Field3: letters, numbers, and space only(`^[a-zA-Z0-9 ]*$`)

Field4: enumeration of numbers (`^(one|two|three|four|five|six|seven|eight|nine)$`)

Field5: simple zip code (`^\d{5}$`)

Field6: zip with optional dash four (`^\d{5}(-\d{4})?$`)

Field7: US phone number with or without dashes (`^[2-9]\d{2}-?\d{3}-?\d{4}$`)

# EXERCISE #4: WEBGOAT

*Solution Demonstration*



# ATTACKING AUTHENTICATION



SECTION 5

# AUTHENTICATION

- Authentication is a mechanism for validating a user.
- There are many authentication technologies:
  - HTML forms-based authentication
  - Multifactor authentication
  - Client SSL certificates and/or smartcards
  - etc
- In general, there are two factors that result in insecure authentication:
  - Design flaws in authentication mechanisms
  - Implementation flaws in authentication

# DESIGN FLAWS IN AUTHENTICATION MECHANISMS

- Bad passwords\*
- Brute-forcible logins
- Verbose Failure messages
- Vulnerable transmission of credentials
- Weaknesses in password change functionality
- Weaknesses in forgotten password functionality\*
- etc.

# BAD PASSWORDS

- Very short or blank passwords
- Common dictionary words or names
- The same as the username
- Still set to the default value
  
- Check the strength of your password:

<https://password.kaspersky.com/>

# EXERCISE #5: WEBGOAT

## Forgot Password

The goal is to retrieve the password of another user.

**Webgoat Password Recovery**  
Please input your username. See the OWASP admin if you do not have an account.

\*Required Fields

\*User Name:

# EXERCISE #5: WEBGOAT

*Solution Demonstration*



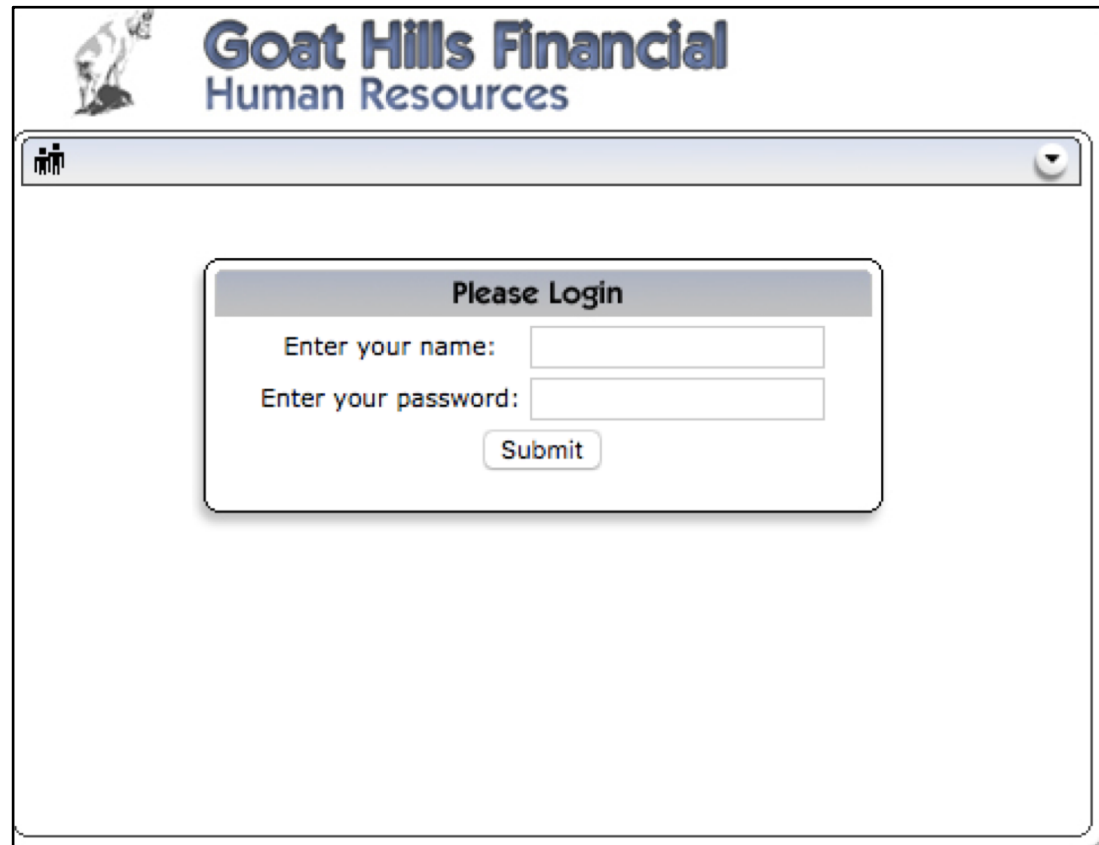
# IMPLEMENTATION FLAWS IN AUTHENTICATION

- Fail-open login mechanisms
- Defects in multistage login mechanisms\*
  - Assumption that access to a later stage means that the user cleared prior stages.
  - Trusting client side data across stages
- Insecure storage of credentials
- etc.

# EXERCISE #6: WEBGOAT

## Multi Level Login 2

Your goal is to log in as Jane.



The screenshot shows a web browser window with the title bar "Goat Hills Financial Human Resources". The page content includes a header with a goat logo and the text "Goat Hills Financial Human Resources". Below the header is a navigation bar with a user icon and a dropdown arrow. The main content area features a "Please Login" form with two input fields: "Enter your name:" and "Enter your password:", followed by a "Submit" button.

# EXERCISE #6: WEBGOAT

*Solution Demonstration*



# ATTACKING SESSION MANAGEMENT

SECTION 6

# ATTACKING SESSION MANAGEMENT

- Understand the mechanism
- Test session tokens for meaning
- Test session tokens for predictability
- Check for session termination

# EXERCISE #7: WACKOPICKO

## Session Management

Try to determine how the session is being calculated for the admin interface. Log in as admin/admin multiple times to solve this exercise.

### Admin Area

Username :

Password :

# EXERCISE #7: WACKOPICKO

*Solution Demonstration*

---

# ATTACKING DATA STORES

SECTION 7



# ATTACKING DATA STORES

- Most applications have a data store to manage and store data.
  - User accounts, credentials and personal information.
  - Prices of items
  - Orders
  - Privilege level of a user
- We'll test for SQL injections.
  - Supply unexpected syntax that might cause problems in the application.
  - Identify and analyze any anomalies and error messages received.
  - Attempt to exploit the vulnerability

# EXERCISE #8: WEBGOAT

## String SQL Injection

Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = ?
```

No results matched. Try Again.

# EXERCISE #8: WEBGOAT

*Solution Demonstration*

---

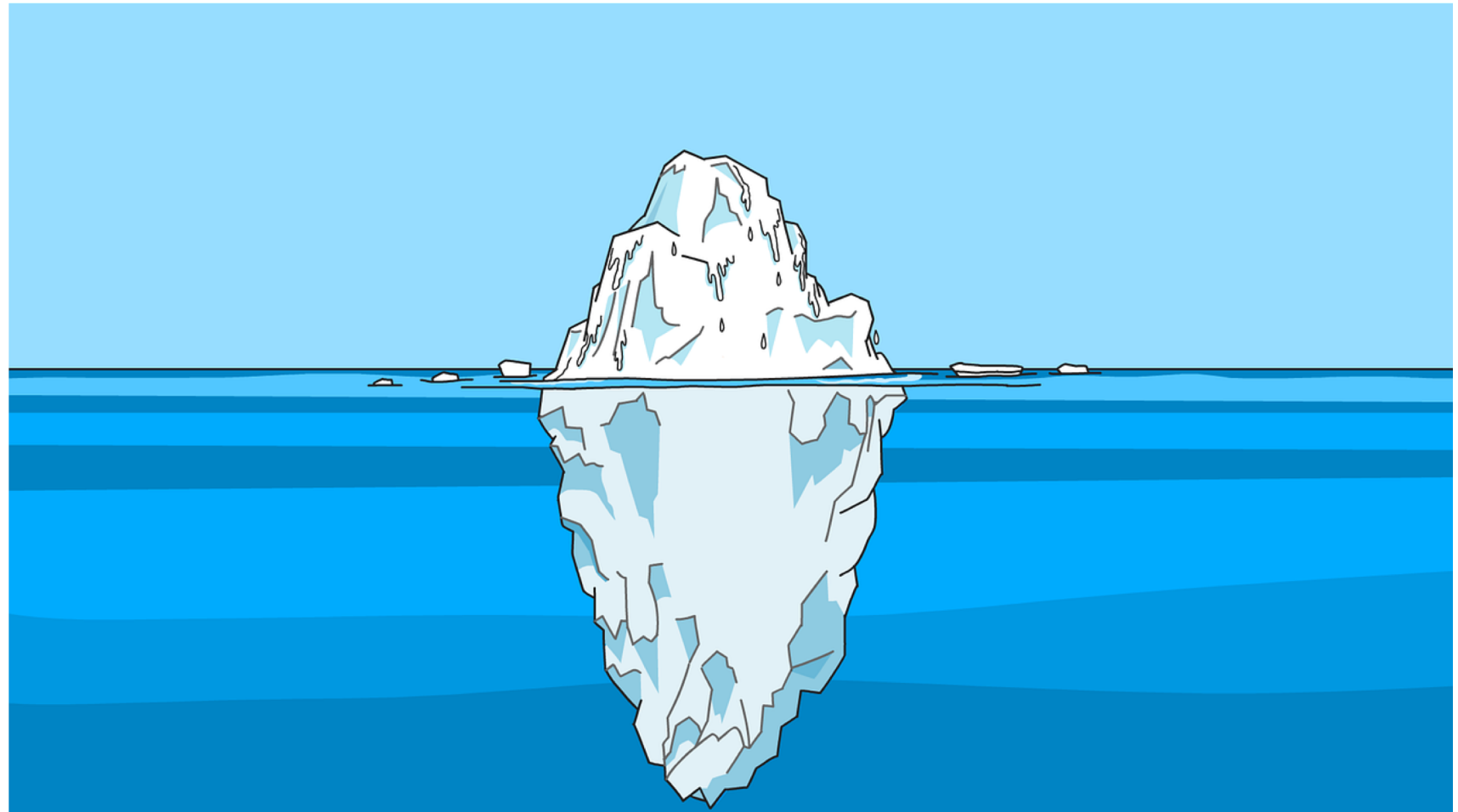
# CONCLUSION

SECTION 8

# WHAT NEXT?

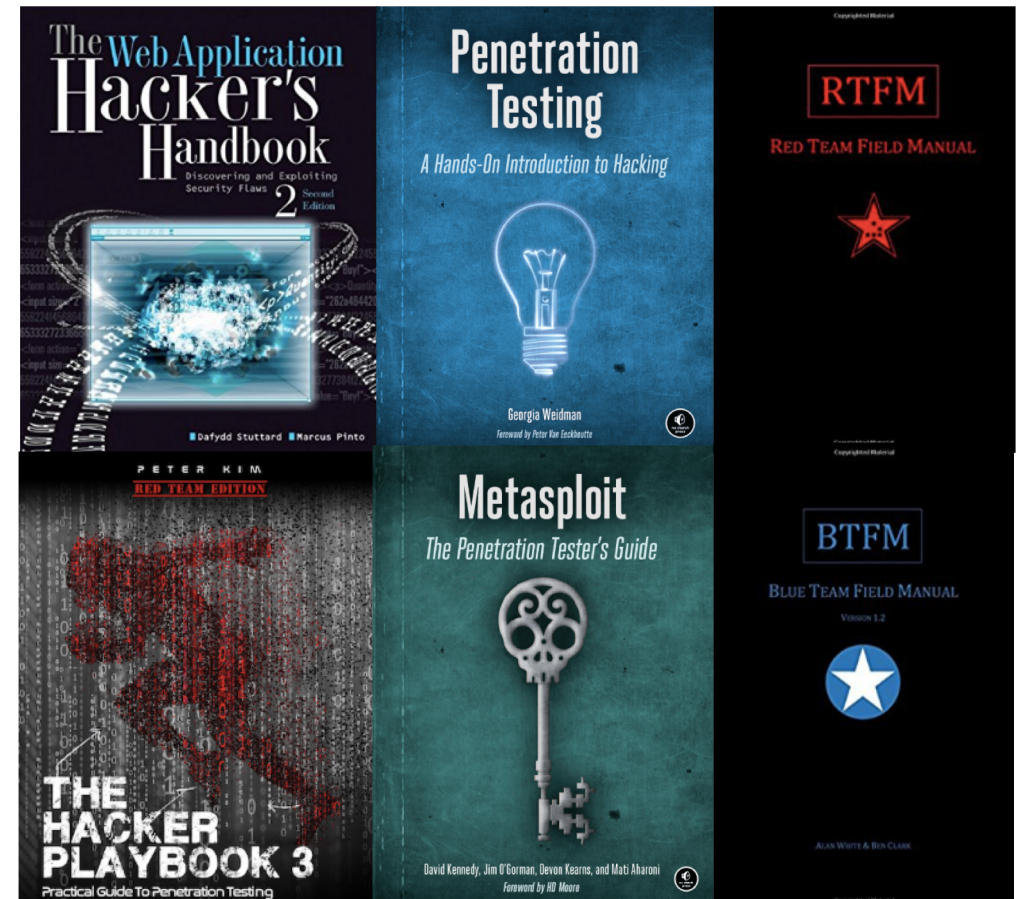
Concepts  
we learned  
today

Remaining  
concepts



# WHAT NEXT?

- Read books
- Get a mentor
- So many free online resource! (Cybrary, coursera, MIT open courseware, etc.)
- Practice on intentionally vulnerable web applications
- Participate in CTF competitions
- Attend conferences
- Contribute to open-source security projects
- Apply to security jobs (Even if you don't have all the qualifications. Most people don't!)
- Take life one step at a time. You can do this!



# GET IN TOUCH!



**/ranakhalil**



**<https://rkhalil01.github.io/>**



**@rana\_\_khalil**



**/rkhalil01**



Laptop stickers  
generously  
donated by  
OWASP!

