

Part 1: Q&A

The URL Cruise Missile

The URL is the gateway to the web, providing the user with unrestricted access to all available online resources. In the wrong hands can be used as a weapon to launch attacks.

Use the graphic below to answer the following questions:

Protocol	Host Name	Path	Parameters
-----	-----	-----	-----
http://	www.buyitnow.tv	/ads.asp	?item=price#1999

Which part of the URL can be manipulated by an attacker to exploit a vulnerable back-end database system?

Answer: **Parameters**

2. Which part of the URL can be manipulated by an attacker to cause a vulnerable web server to dump the /etc/passwd file? Also, name the attack used to exploit this vulnerability.

Answer: **Path , Path Traversal Attack**

3. Name three threat agents that can pose a risk to your organization.

Answer: **Script Kiddies, Organized Cybercriminals, Insider Threats**

4. What kinds of sources can act as an attack vector for injection attacks?

Answer: **Can include malware, web pages, social engineering**

5. Injection attacks exploit which part of the CIA triad?

Answer: **Confidentiality**

6. Which two mitigation methods can be used to thwart injection attacks?

Answer: **Input validation and sanitization**

Web Server Infrastructure

Web application infrastructure includes sub-components and external applications that provide efficiency, scalability, reliability, robustness, and most critically, security.

- The same advancements made in web applications that provide users these conveniences are the same components that criminal hackers use to exploit them. Prudent security administrators need to be aware of how to harden such systems.

Use the graphic below to answer the following questions:

Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Client	Firewall	Web Server	Web Application	Database

1. What stage is the most inner part of the web architecture where data such as, customer names, addresses, account numbers, and credit card info, is stored?

Answer: **Stage 5 | Database**

2. Which stage includes online forms, word processors, shopping carts, video and photo editing, spreadsheets, file scanning, file conversion, and email programs such as Gmail, Yahoo and AOL.

Answer: **Stage 4 | Web Application**

3. What stage is the component that stores files (e.g. HTML documents, images, CSS stylesheets, and JavaScript files) that's connected to the Internet and provides support for physical data interactions between other devices connected to the web?

Answer: **Stage 3 | Web Server**

4. What stage is where the end user interacts with the World Wide Web through the use of a web browser?

Answer: **Stage 1 | Client**

5. Which stage is designed to prevent unauthorized access to and from protected web server resources?

Answer: **Stage 2 | Firewall**

Server Side Attacks

In today's globally connected cyber community, network and OS level attacks are well defended through the proper deployment of technical security controls such as, firewalls, IDS, Data Loss Prevention, EndPoint and security. However, web servers are accessible from anywhere on the web, making them vulnerable to attack.

1. What is the process called that cleans and scrubs user input in order to prevent it from exploiting security holes by proactively modifying user input.

Answer: **Input Sanitation**

2. Name the process that tests user and application-supplied input. The process is designed to prevent malformed data from entering a data information system by verifying user input meets a specific set of criteria (i.e. a string that does not contain standalone single quotation marks).

Answer: **Input Validation**

3. **Secure SDLC** is the process of ensuring security is built into web applications throughout the entire software development life cycle. Name three reasons why organization might fail at producing secure web applications.

Answer: **Garbage managers/management, prices are too expensive, or reliance completely on firewalls**

4. How might an attacker exploit the robots.txt file on a web server?

Answer: **A hacker may modify the URL to access the robot.txt which typically contains sensitive information**

5. What steps can an organization take to obscure or obfuscate their contact information on domain registry web sites?

Answer: **Private domains**

6. True or False: As a network defender, Client-Side validation is preferred over Server-Side validation because it's easier to defend against attacks.
 - Explain why you chose the answer that you did.

Answer: **False, access controls are only on the server-side meaning hackers cannot modify certain information**

Web Application Firewalls

WAFs are designed to defend against different types of HTTP attacks and various query types such as SQLi and XSS.

WAFs are typically present on web sites that use strict transport security mechanisms such as online banking or e-commerce websites.

1. Which layer of the OSI model do WAFs operate at?

Answer: **Layer 7 | Application**

2. A WAF helps protect web applications by filtering and monitoring what?

Answer: **HTTP traffic between web applications and the internet**

3. True or False: A WAF based on the negative security model (Blacklisting) protects against known attacks, and a WAF based on the positive security model (Whitelisting) allows pre-approved traffic to pass.

Answer: **True**

Authentication and Access Controls

Security enhancements designed to require users to present two or more pieces of evidence or credentials when logging into an account is called multi-factor authentication.

- Legislation and regulations such as The Payment Card Industry (PCI) Data Security Standard requires the use of MFAs for all network access to a Card Data Environment (CDE).
- Security administrators should have a comprehensive understanding of the basic underlying principles of how MFA works.

1. Define all four factors of multifactor authentication and give examples of each:

- Factor 1: **Login input. Custom username/password**
- Factor 2: **Physical keys. A smart card, ID with a special chip**
- Factor 3: **Biometrics. Fingerprint, face scan, eye scan**
- Factor 4: **Location. GPS, a callback to a home phone number**

2. True or False: A password and pin is an example of 2-factor authentication.

Answer: **False**

3. True or False: A password and google authenticator app is an example of 2-factor authentication.

Answer: **True**

4. What is a constrained user interface?

Answer: **Constrains what a user can see on any given interface and restricts what users can do based on their privileges**

Part 2: The Challenge

In this activity, you will assume the role of a pen tester hired by a bank to test the security of the bank's authentication scheme, sensitive financial data, and website interface.

Lab Environment

We'll use the **Web Vulns** lab environment. To access it:

- Log in to the Azure Classroom Labs dashboard.
- Find the card with the title **Web Vulns** or **Web Vulnerability and Hardening**.
- Click the monitor icon in the bottom-right.
- Select **Connect with RDP**.
- Use Credentials (azadmin:p4ssw0rd*)
- The lab should already be started, so you should be able to connect immediately.
- Refer to the [lab setup instructions](#) for details on setting up the RDP connection.

Once the lab environment is running, open the HyperV manager and make sure that the OWASPBWA and Kali box is running.

- Then, login to the Kali VM and navigate to the IP address of the OWASPBWA machine.
- Click the option for 'WebGoat' and start the WebGoat app.
- Use the credentials: guest:guest

On the bottom of the left side of the screen, click on Challenge and then choose The Challenge.

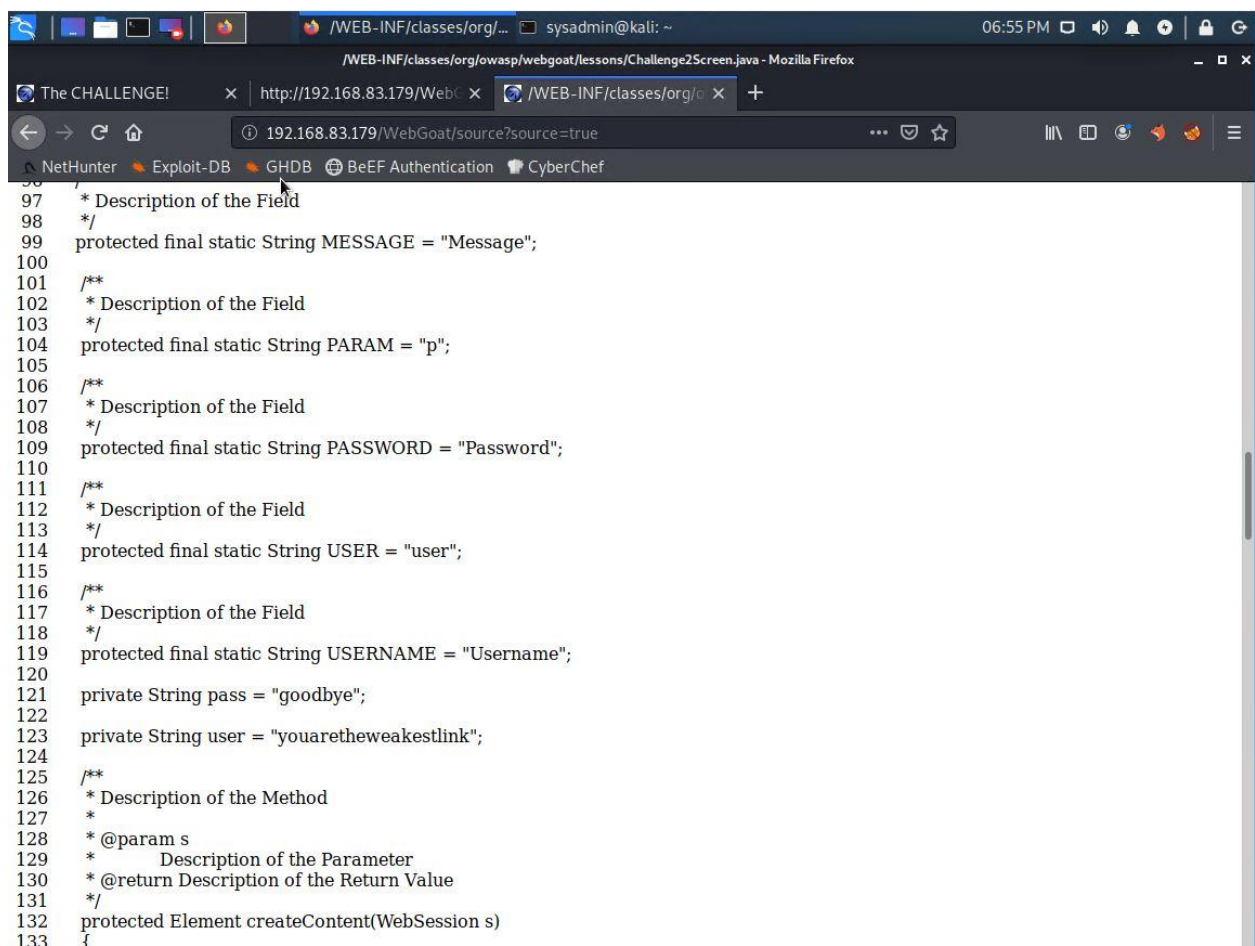
Note: A common issue with this lab is the Challenge activity failing to start successfully. Hit the Restart the Lesson button in the top right if you get an error starting the activity.

The Challenge Instructions

Challenge #1

Your first mission is to break the authentication scheme. There are a number of ways to accomplish this task.

- **Hint #1:** Sometimes, form fields are shy!
- **Hint #2:** Find the hidden JavaScript.
- **Hint #3:** You can append source?source=true to the URL to read the source code.



```
97  * Description of the Field
98  */
99  protected final static String MESSAGE = "Message";
100
101  /**
102  * Description of the Field
103  */
104  protected final static String PARAM = "p";
105
106  /**
107  * Description of the Field
108  */
109  protected final static String PASSWORD = "Password";
110
111  /**
112  * Description of the Field
113  */
114  protected final static String USER = "user";
115
116  /**
117  * Description of the Field
118  */
119  protected final static String USERNAME = "Username";
120
121  private String pass = "goodbye";
122
123  private String user = "youaretheweakestlink";
124
125  /**
126  * Description of the Method
127  *
128  * @param s
129  *       Description of the Parameter
130  * @return Description of the Return Value
131  */
132  protected Element createContent(WebSession s)
133  {
```

Please include a screenshot here of the hidden JavaScript:

After completing the first challenge, you will be provided with an option to continue to the next challenge.

Challenge #2

Next, steal all of the credit card numbers from the database.

- **Hint #1:** Sometimes cookies wear different clothes to change their appearances.
- **Hint #2:** Break your way into the conversation and inject your own ideas.

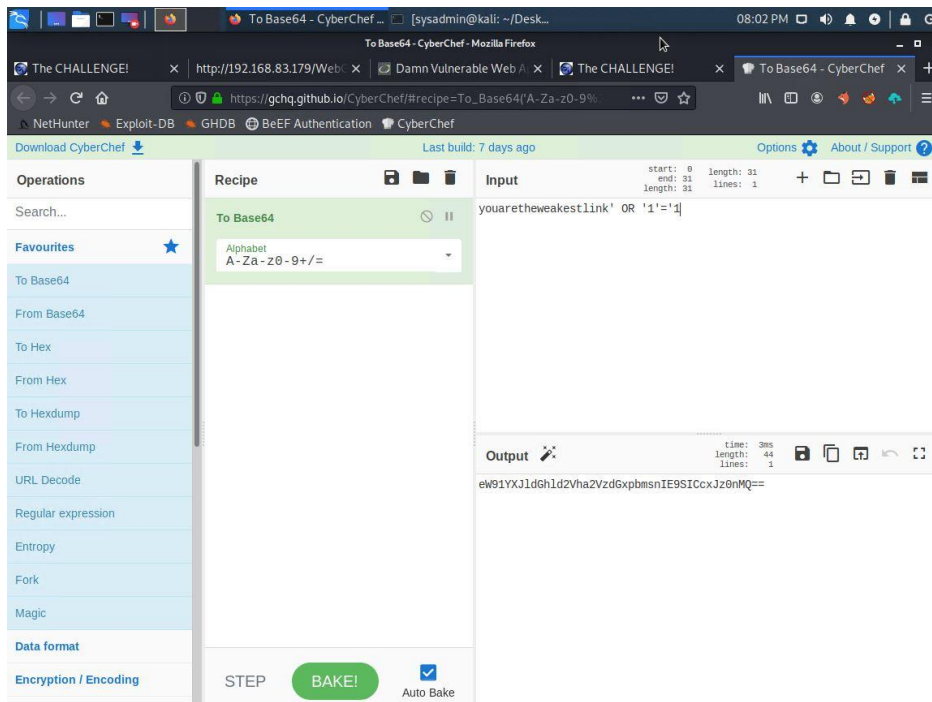
Please include a screenshot here of all the credit card numbers from the database.

After completing the second challenge, you will be provided with an option to continue to the next challenge.

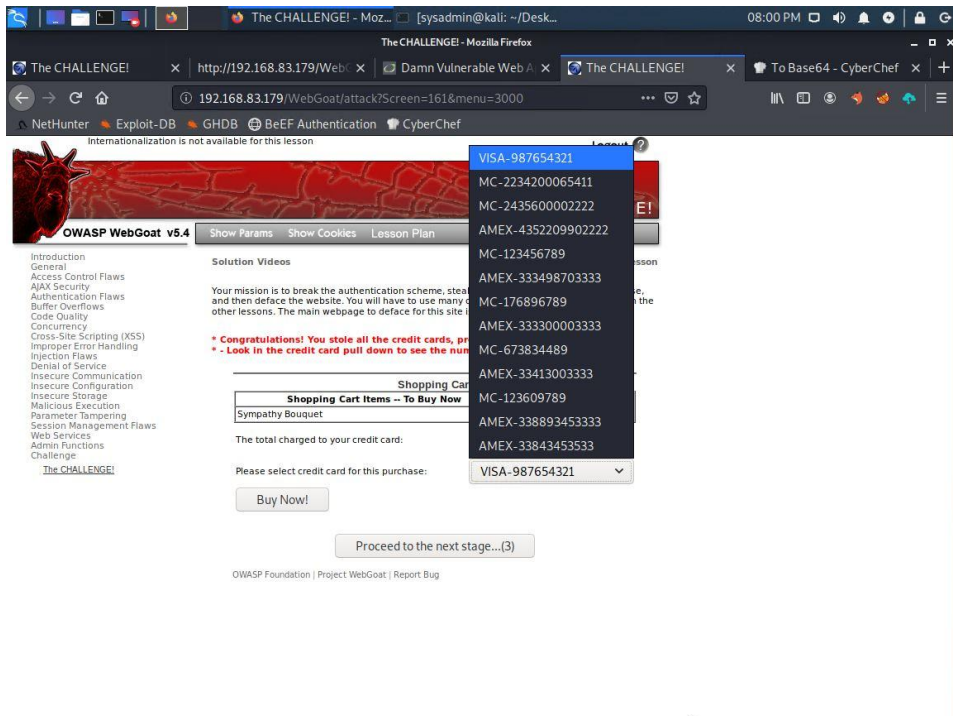
Challenge #3

Your final act is to deface the website using command injection. Follow the walkthrough below to help you get started.

- After completing the second challenge, you will be provided with an option to continue to the next challenge.



- There should be two webpages at the bottom of the window. The one on top is the original, and the one on the bottom is the defaced webpage.



- Start Foxy Proxy (WebScarab) to send all GET/POST requests from Firefox to the WebScarab proxy intercept.



- Click **TCP** and then the **View Network** button and send the request to WebScarab.



- The WebScarab window will open.
 - In the **URL Encoded** tab, find the **File** and **Value** form fields.
 - This is where you will perform your command injection.



- Next, perform a test and see if this shell is vulnerable to command injection.
 - Type the following command into the Value field: `tcp && whoami && pwd`.
 - **Note:** Windows users can type `tcp && dir`. `dir` will return the directory as proof of vulnerability.
 - Click **Accept Changes**.



- On the next window, click **Accept Changes** twice.



- Scroll to the bottom of the **Current Network Status** window and observe the results for both of the `whoami` and `pwd` commands.



- The results show that we are the root user and our current working directory is `/var/lib/tomcat6`.
- This verifies the vulnerability, so proceed to the next step.
- Next, we'll locate the `webgoat_challenge_guest.jsp` file.
 - Type the following command: `tcp && cd / && find . -iname webgoat_challenge_guest.jsp`.
 - **Note:** Windows users will need to type: `tcp && dir /s 'webgoat_challenge_guest.jsp'`



- The absolute path is:
`./owaspbwa/owaspbwa-svn/var/lib/tomcat6/webapps/WebGoat/webgoat_challenge_guest.jsp`.



- Remember, our present working directory is `/var/lib/tomcat6`. Therefore, the relative path is `webapps/WebGoat/webgoat_challenge_guest.jsp`.

Now it's your turn

- Now that we know where the webpage is, your task will be to deface the website. Keep in mind the following:
 - Use **WebScarab** to perform command injection.
 - When performing command injection, you will need to select a field that WebScarab can return commands to. These fields are typically located in a drop down.
 - You will also need to locate and edit the webpage's source code:
`webgoat_challenge_guest.jsp`
 - Your final command will:
 - Change to the location of the `webgoat_challenge_guest.jsp` file.
 - **and** echo You've been hacked by... followed by your name, to the `webgoat_challenge_guest.jsp` file.

Please include a screenshot of the defaced website.

The CHALLENGE! - Mozilla Firefox

File Act The CHALLENGE! x http://192.168.83.179/M x To Base64 - CyberC x Damn Vulnerable W x The CHALLENGE! x +

192.168.83.179/WebGoat/attack?Screen=161&menu=301

NetHunter Exploit-DB GHDB BeEF Authentication CyberChef

Internationalization is not available for this lesson Logout ?

OWASP WebGoat v5.4

Show Params Show Cookies Lesson Plan

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Web Services
Admin Functions
Challenge
The CHALLENGE!


Solution Videos Restart this Lesson

Your mission is to break the authentication scheme, steal all the credit cards from the database, and then deface the website. You will have to use many of the techniques you have learned in the other lessons. The main webpage to deface for this site is 'webgoat_challenge_guest.jsp'

*** CONGRATULATIONS - You have defaced the site!**

Proceed to the next stage...(4)

Original Website Text



Thank you for using WebGoat! This program is a demonstration of common web application flaws. The exercises are intended to provide hands on experience with application penetration testing techniques.

The WebGoat project is led by Bruce Mayhew. Please send all comments to Bruce at WebGoat@owasp.org.

OWASP ASPECT SECURITY

sysadmin@kali: /var/lib\$

The CHALLENGE! - Mozilla Firefox

File Act The CHALLENGE! x http://192.168.83.179/M x To Base64 - CyberC x Damn Vulnerable W x The CHALLENGE! x +

192.168.83.179/WebGoat/attack?Screen=161&menu=301

NetHunter Exploit-DB GHDB BeEF Authentication CyberChef

Session Management Flaws
Web Services
Admin Functions
Challenge
The CHALLENGE!

OWASP WebGoat v5.4

Thank you for using WebGoat! This program is a demonstration of common web application flaws. The exercises are intended to provide hands on experience with application penetration testing techniques.

The WebGoat project is led by Bruce Mayhew. Please send all comments to Bruce at WebGoat@owasp.org.

OWASP ASPECT SECURITY

WebGoat Authors
Bruce Mayhew
Jeff Williams

WebGoat Design Team
David Anderson
Laurence Gasey (Graphics)
Rogan Dawes
Bruce Mayhew

V5.4 Lesson Contributors
Sherif Koussa
Yannis Pavlosoglou

Special Thanks for V5.4
Brian Cornell (Multitude of bug fixes)
To all who have sent comments

Documentation Contributors
Erwin Geimaert
Aung Khant
Sherif Koussa

Start WebGoat

Defaced Website Text

you_have_been_hacked

sysadmin@kali: /var/lib\$