tutorialspoint.com

Security Testing - Quick Guide

tutorialspoint.com

45-57 minutes

What is Security Testing?

Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. By Performing security testing, it is no guarantee that systems are secure but it is important to include the security testing as part of the testing process. It also aims at verifying 6 basic principles as listed below:

- **Confidentiality**
- **Integrity**
- **Authentication**
- **Authorization**
- **Availability**
- **Non-repudiation**

Example

Spotting a security flaw in a web based app involves complex steps and a creative thinking but, attimes a simple tests like the one

below can help expose the most severe security risks. Below is a very basic security test which anyone can perform on any web application :

1. Log into the web application using valid credentials.
2. Log out of the web application.
3. Click the BACK button of the browser. Verify if you are asked to log in again or if you are able go back to the logged in page again.

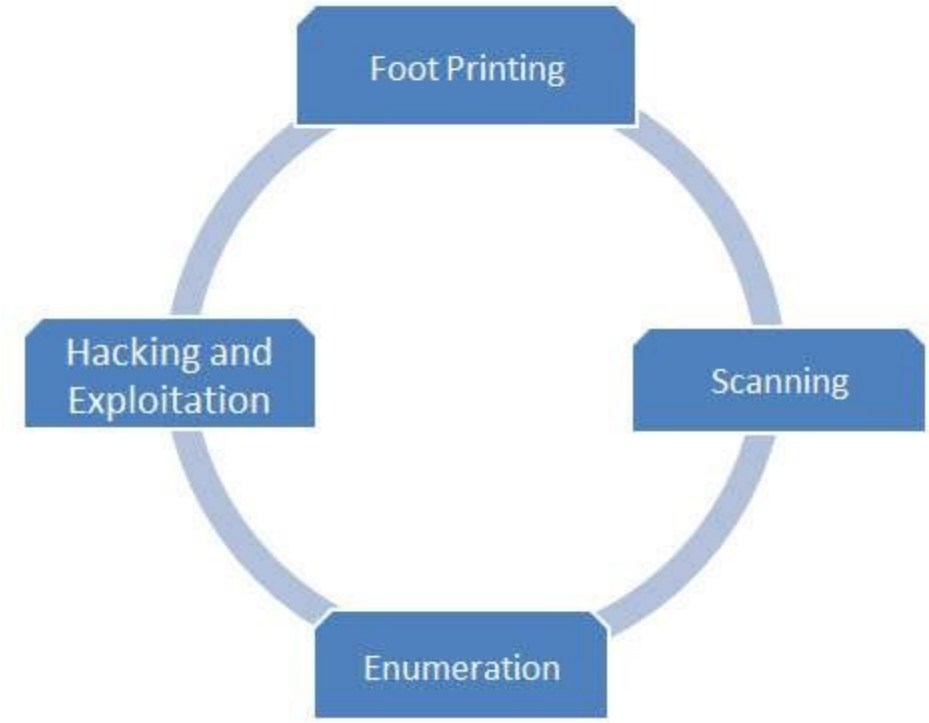
Security Testing - Process

The goal of a penetration test also called ethical hacking, is to evaluate the current security status of IT systems. It is a controlled attack which uncovers security flaws in a realistic way.

As we are involved in the process, documentation should be done phase so that all the steps necessary to reproduce the attack are available readily which is the basis for the detailed report customers receive at the end of a penetration test.

These phases are re-iterated multiple times in a pentest phase which goes hand in hand with the normal SDLC.

Pentest Workflow



The four major phases of security Testing are the following. Click on each one of the phases to understand in detail.

- Foot Printing
- Scanning
- Enumeration
- Exploitation

Security Testing - Malacious Software

Malicious software (malware) is any software that gives partial to full control of the system to the attacker/malware creator. Various forms of Malware are listed below :

- **Viruses** - Virus, a self inserting copies of itself into other computer programs into data file or the boot sector of the hard drive. Upon succesful replication, viruses cause harmful activity on infected hosts such as stealing hard disk space or CPU time.

- **Worms** - A worm is a type of malware which leaves a copy of itself in the memory of each computer in its path.
- **Trojans** - Trojan, non-self-replicating type of malware that contains malicious code which upon execution results in loss or theft of data or possible system harm
- **Adware** Adware also known as freeware or pitchware is a free computer software that contains commercial advertisements that include games, desktop toolbars and utilities. It is a Web-based app and collects Web browser data to target advertisements especially pop-ups.
- **Spyware** - Spyware is infiltration software that anonymously monitors users which enables a hacker to obtain sensitive information from the user's computer. Spyware exploits users and application vulnerabilities that is quite often attached to free online software downloads or to links that are clicked by users.
- **Root kit** - A rootkit is a software used by a hacker to gain admin level access to a computer/network which is installed through a stolen password or by exploiting a system vulnerability without the victim's knowledge.

Preventing Measures:

- Ensure that the operating system and any program you are using is upto date with patches/updates.
- DONOT open strange e-mails, especially ones with attachments which might be any of the malware as mentioned above.
- When downloaded from internet, always check what you install. Do not simply click OK to dismiss pop-up windows. Verify the publisher

before you install them.

- Install anti-virus software; and also ensure you scan and update them regularly. In most cases anti-virus programs remove and prevent viruses, worms, trojans, and some spyware.

Anti Malware Softwares

- Microsoft Security Essentials
- Microsoft Windows Defender
- AVG Internet Security
- Spybot - Search & Destroy
- Avast! Home Edition for personal use
- Panda Internet Security
- MacScan for Mac OS and Mac OS X

HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (ie. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extension of its request methods, error codes and headers.

Basically, HTTP is an TCP/IP based communication protocol, which is used to deliver data such as HTML files, image files, query results etc over the Web. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients request data will sent to the server, and how

servers respond to these requests.

Understanding the protocol is very important to get good hands on Security testing. You will be able to appreciate the importance of the protocol when we intercept the packet data between the webserver and the client.

Basic Features

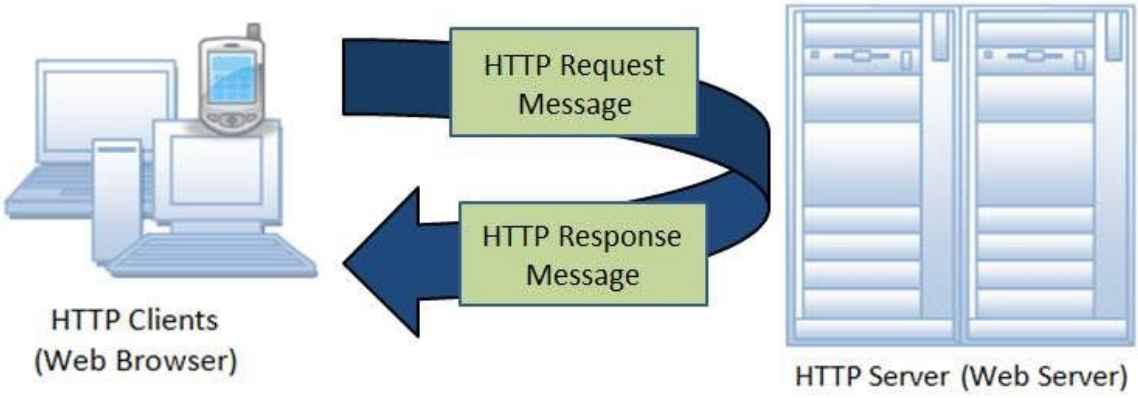
There are following three basic features which makes HTTP a simple but powerful protocol:

- **HTTP is connectionless:** The HTTP client ie. browser initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server process the request and re-establish the connection with the client to send response back.
- **HTTP is media independent:** This means, any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. This is required for client as well as server to specify the content type using appropriate MIME-type.
- **HTTP is stateless:** As mentioned above, HTTP is a connectionless and this is a direct result that HTTP is a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different request across the web pages.

> HTTP/1.0 uses a new connection for each request/response exchange where as HTTP/1.1 connection may be used for one or more request/response exchanges.

Architecture

Following diagram shows a very basic architecture of a web application and depicts where HTTP sits:



The HTTP protocol is a request/response protocol based on client/server based architecture where web browser, robots and search engines, etc. act like HTTP clients and Web server acts as server.

- **Client -** The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.
- **Server -** The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

Disadvantages

- HTTP is NOT a secured protocol.
- HTTP uses port 80 as default for communication.
- HTTP operates at Application Layer.

- No Encryption/digital certificates required for using HTTP

Http Protocol Details

Inorder to understand the HTTP Protocol indepth, click on each on of the below links.

- HTTP Parameters
- HTTP Messages
- HTTP Requests
- HTTP Responses
- HTTP Methods
- HTTP Status Codes
- HTTP Header Fields
- HTTP Security

HTTPS Protocol

HTTPS (Hypertext Transfer Protocol over Secure Socket Layer) or HTTP over SSL is a web protocol developed by Netscape. It is not a protocol but it is just the result of layering the HTTP on top of SSL/TLS (Secure Socket Layer/Transport Layer Security).

Inshort, HTTPS = HTTP + SSL

When Https Required

When we browse, we normally send and receive information using HTTP protocol. So this leads anyone to eavesdrop on the conversation between our computer and the web server. Many a times we need to exchange sensitive information which needs to be

secured and to prevent unauthorized access.

Https protocol used in the following scenarios

- Banking Websites
- Payment Gateway
- Shopping Websites
- All Login Pages
- Email Apps

Basic Working of HTTPS

- Public key and signed certificates are required for the server in HTTPS Protocol.
- Client requests for the https:// page
- When using an https connection, the server responds to the initial connection by offering a list of encryption methods the webserver supports.
- In response, the client selects a connection method, and the client and server exchange certificates to authenticate their identities.
- After this is done, both webserver and client exchange the encrypted information after ensuring that both are using the same key, and the connection is closed.
- For hosting https connections, a server must have a public key certificate, which embeds key information with a verification of the key owner's identity.
- Almost all certificates are verified by a third party so that clients are assured that the key is always secure.

What is Encoding?

Encoding is the process of putting a sequence of characters such as letters, numbers and other special characters into a specialized format for efficient transmission while Decoding is the process of converting an encoded format back into the original sequence of characters. It is completely different from Encryption which we usually misinterpret.

Encoding and decoding are used in data communications and storage. Encoding should NOT be used for transporting sensitive information.

URL Encoding

URLs can only be sent over the Internet using the ASCII character-set and there are instances when URL contains special characters apart from ASCII characters, it needs to be encoded. URLs donot contain spaces and are replaced with a plus (+) sign or with %20.

ASCII Encoding

The Browser(client side) will encode the input according to the character-set used in the web-page and the default character-set in HTML5 is UTF-8.

Following table shows ASCII symbol of the character and its equal Symbol and finally its replacement which can be used in URL before passing it to the server:

ASCII Symbol	Replacement

< 32		Encode with %xx where xx is the hexadecimal representation of the character.
32	space	+ or %20
33	!	%21
34	"	%22
35	#	%23
36	$	%24
37	%	%25
38	&	%26
39	'	%27
40	(	%28
41	)	%29
42	*	*
43	+	%2B
44	,	%2C
45	-	-
46	.	.
47	/	%2F
48	0	0
49	1	1
50	2	2
51	3	3
52	4	4
53	5	5
54	6	6
55	7	7

56	8	8
57	9	9
58	:	%3A
59	;	%3B
60	<	%3C
61	=	%3D
62	>	%3E
63	?	%3F
64	@	%40
65	A	A
66	B	B
67	C	C
68	D	D
69	E	E
70	F	F
71	G	G
72	H	H
73	I	I
74	J	J
75	K	K
76	L	L
77	M	M
78	N	N
79	O	O
80	P	P
81	Q	Q

82	R	R
83	S	S
84	T	T
85	U	U
86	V	V
87	W	W
88	X	X
89	Y	Y
90	Z	Z
91	[	%5B
92	\	%5C
93	]	%5D
94	^	%5E
95	_	_
96	`	%60
97	a	a
98	b	b
99	c	c
100	d	d
101	e	e
102	f	f
103	g	g
104	h	h
105	i	i
106	j	j
107	k	k

108	l	l
109	m	m
110	n	n
111	o	o
112	p	p
113	q	q
114	r	r
115	s	s
116	t	t
117	u	u
118	v	v
119	w	w
120	x	x
121	y	y
122	z	z
123	{	%7B
124	|	%7C
125	}	%7D
126	~	%7E
127		%7F
> 127		Encode with %xx where xx is the hexadecimal representation of the character

What is Cryptography?

Cryptography is the science to encrypt and decrypt data that enables the users to store sensitive information or transmit it across

insecure networks so that it can be read only by the intended recipient.

Data which can be read and understood without any special measures is called plaintext while the method of disguising plaintext in inorder to hide its substance is called encryption.

Encrypted plain text is known as ciphertext and process of reverting the encrypted data back to plain text is known as decryption.

- The science of analyzing and breaking secure communication is known as cryptanalysis. The people who perform the same also known as attackers.
- Cryptography can be either strong or weak and the strength is measured by the time and resources it would require to recover the actual plaintext.
- Hence appropriate decoding tool is required to decipher the strong encypted messages.
- There are some cryptographic techniques available with which even a billion computers doing a billion checks a second, it is not possible to decipher the text.
- As power of computing increases day by day, one has to make their encryption algorithm very strong inorder to protect it from the attackers.

How Encryption Works

A cryptographic algorithm works in combination with a key(can be a word, number, or phrase) to encrypt the plaintext and the same plaintext encrypts to different ciphertext with different keys.

Hence, the encrypted data is completely dependent couple of

parameters viz- the strength of the cryptographic algorithm and the secrecy of the key.

Cryptography Techniques

Symmetric Encryption - conventional cryptography, also known as Conventional encryption in which one key is used both for encryption and decryption. Eg: DES, Triple DES algorithms, MARS by IBM, RC2,RC4, RC5,RC6.

Asymmetric Encryption - It is Public key cryptography that uses a pair of keys for encryption: a public key, which encrypts data, and a private key used for decryption. Public key is published to the people while keeping the private key secret. Eg: RSA, Digital Signature Algorithm (DSA), Elgamal

Hashing - Hashing is ONE way encryption, which the scrambled output that cannot be reversed or at least cannot be reversed easily that is used to validate the integrity of information. Eg: MD5 algorithm. It is used to create Digital Certificates, Digital signatures, Storage of passwords , Verification of communications.

What is Same Origin Policy?

Same Origin Policy(SOP) is an important concept in the web application security model. As per this policy, it permits scripts running on pages originating from the same site which can be a combination of the following

- Domain
- Protocol
- Port

Example

The reason behind this behaviour is security. If you have try.com in one window and gmail.com in another window, then you DONOT want a script from try.com to access or modify the contents of gmail.com or run actions in context of gmail on your behalf.

Below are webpages from the same origin. As explained before, the same origin takes domain/protocol/port into consideration.

- http://website.com
- http://website.com/
- http://website.com/my/contact.html

Below are webpages from a different origin.

- http://www.site.co.uk(another domain)
- http://site.org (another domain)
- https://site.com (another protocol)
- http://site.com:8080 (another port)

Same Origin policy Exceptions for IE

Internet Explorer has two major exceptions to SOP.

- The first one is related to 'Trusted Zones'. If both domains are in highly trusted zone then the Same Origin policy is not applicable completely.
- The second exeption in IE is related to port. IE doesn't include port into Same Origin policy, hence the http://website.com and http://wesite.com:4444 are considered from the same origin and no restrictions are applied.

What is a cookie?

cookie, a small piece of info sent by web server to store on a web browser so that it can later read by the browser. Hence browser remembers some specific personal information. If a Hacker gets hold of the cookie information, will lead to security issues.

Cookies - Properties

- It is Usually small text files, given ID tags that are stored on your computer's browser directory
- It is Used by web developers to help users navigate their websites efficiently and perform certain functions
- When the user browses the same website again, the data stored in the cookie is sent back to the web server to notify the website of the users previous activity.
- Cookies are unavoidable for websites that have huge databases, need logins, have customizable themes.

Cookie Contents

- The name of the server the cookie was sent from
- The lifetime of the cookie
- A value - usually a randomly generated unique number

Types of Cookies

- Session Cookies - These cookies are temporary which will be erased when the user closes the browser. Even if the user logs in again into the a new cookie for that session will be created.

- Persistent cookies - These cookies remain on the hard disk drive unless user wipes them off or they expire. The Cookie's expiry would be dependent on how long they can last.

Testing Cookies

- **Disabling Cookies:** As a tester, we need to verify the access of the website after disabling cookies and to check if the pages are working properly. Navigating to all the pages of the website and watch for app crashes. It is also required to inform the user that cookies are required to use the site.
- **Corrupting Cookies:** Another testing to be performed is by corrupting the cookies. In order to do the same, one has to find the location of the site's cookie and manually edit it with fake / invalid data which can be used access internal information from the domain which inturn can then be used to hack the site.
- **Removing Cookies:** Remove all the cookies for the website and check how the website reacts to it.
- **Cross-Browser Compatibility:** It is also important to check that cookies are being written properly on all supported browsers from any page that writes cookies.
- **Editing Cookies:** If the application uses cookies to store login information then as a tester we should try changing the user in the cookie or address bar to another valid user. Editing the cookie should not let you log in to a different users account.

Viewing/Editing Cookies

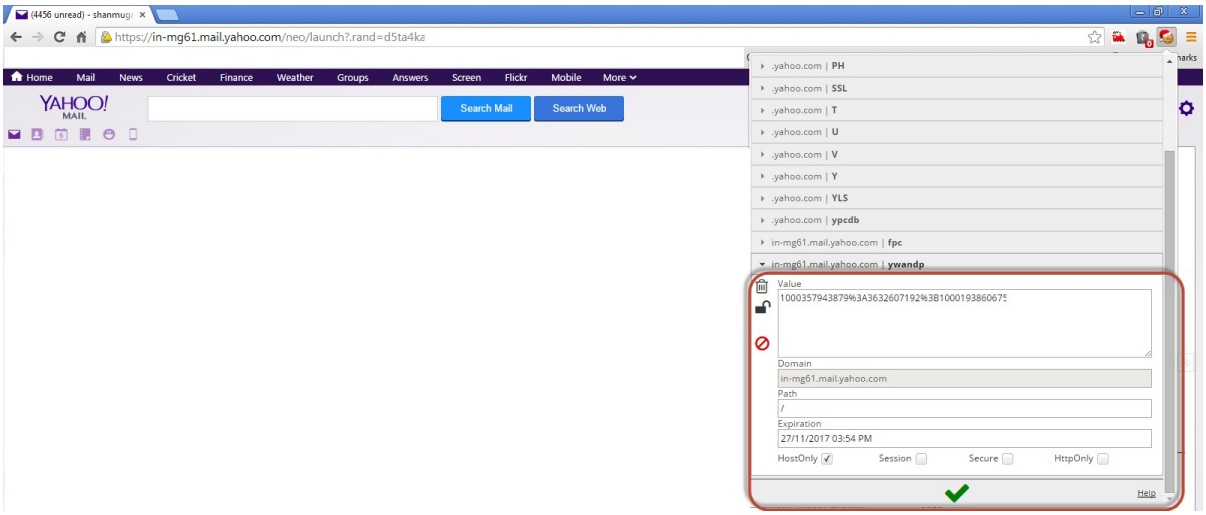
Mordern browsers support viewing/editing of the cookies in the

Browser itself. There are plugins mozilla/chrome using which we will be able to perform the edit successfully.

- Edit Cookies plugin for Firefox
- Edit This Cookie plugin for chrome

Below are the steps one should perform to Edit a cookie.

- Download the plugin for Chrome from here
- Edit the Cookie value just by accessing the 'edit this cookie' plugin from chrome as shown below.



Web Application - PenTesting Methodologies

There are various methodologies/approaches which we can make use as a reference for performing the attaks. Below are the following standards one can take into account while making developing their attack model.

Among the below list, OWASP is the most active and there are lot of contributors. We will focus on OWASP Techniques which each development team takes into consideration before designing a web app.

- PTES - Penetration Testing Execution Standard
- OSSTMM - Open Source Security Testing Methodology Manual
- OWASP Testing Techniques - Open Web Application Security Protocol

OWASP Top 10

The Open Web Application Security Protocol team released the top 10 vulnerabilities that are more prevelant in web in the recent years. Below are the list of security flaws that are more prevelant in a web based application. We will discuss all these techniques in detail in the upcoming chapters.

Application - Hands On

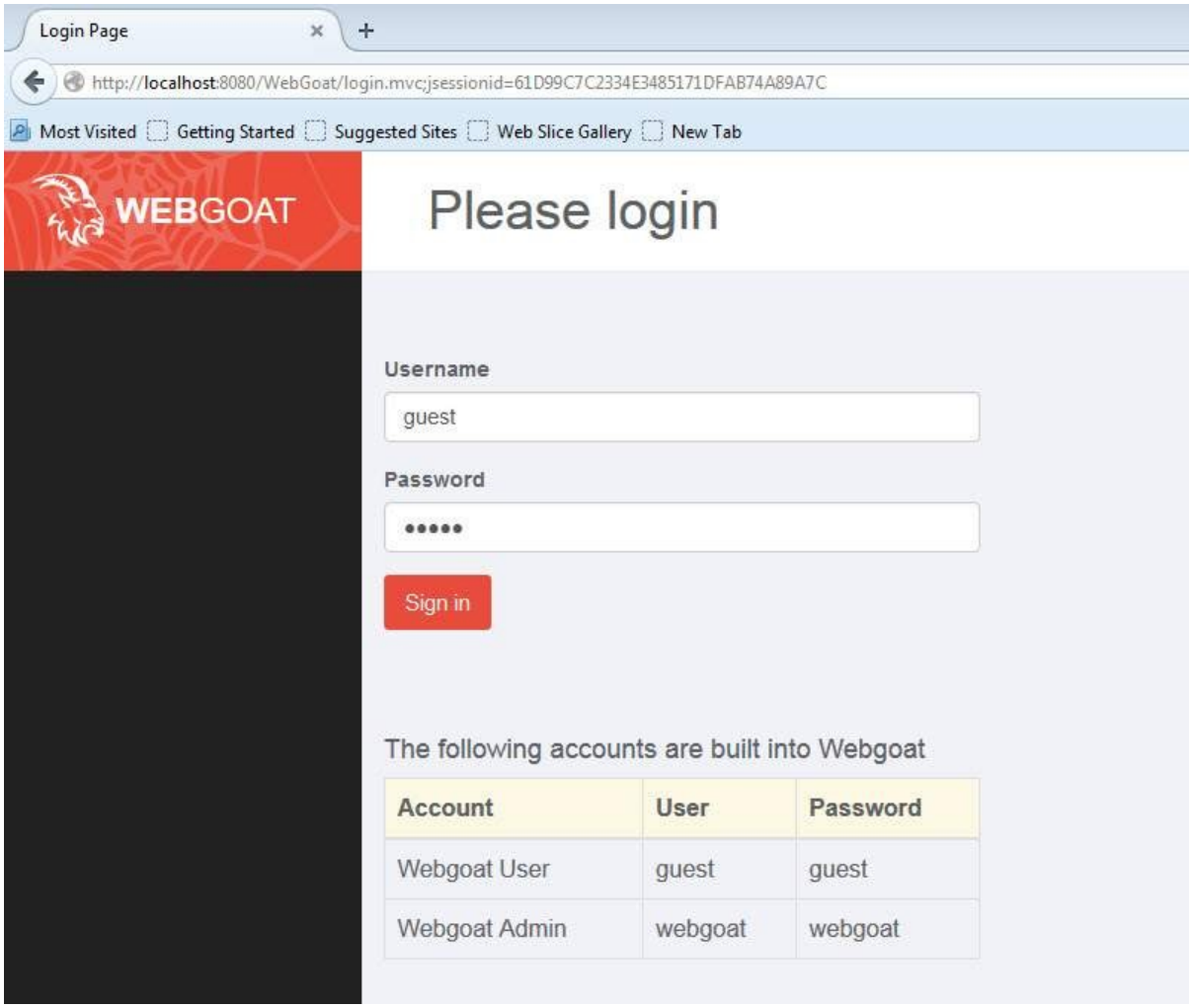
Inorder to understand each one of the techniques, let us work with a sample application. We will perform the attack on 'WebGoat', the J2EE application which has been developed explicitly with security flaws for learning purposes.

The complete details about the webgoat project can be located **here**

To Download the WebGoat Application, Navigate to https://github.com/WebGoat/WebGoat/wiki/Installation-(WebGoat-6.0) and goto downloads section.

To Install the downloaded Application, first ensure that you don't have any application running on Port 8080. It cab installed just using a single command - java -jar WebGoat-6.0.1-war-exec.jar. For more details, WebGoat Installation

Post Installation, we should be able to access the application by navigating to http://localhost:8080/WebGoat/attack and he page would be displayed as shown below.

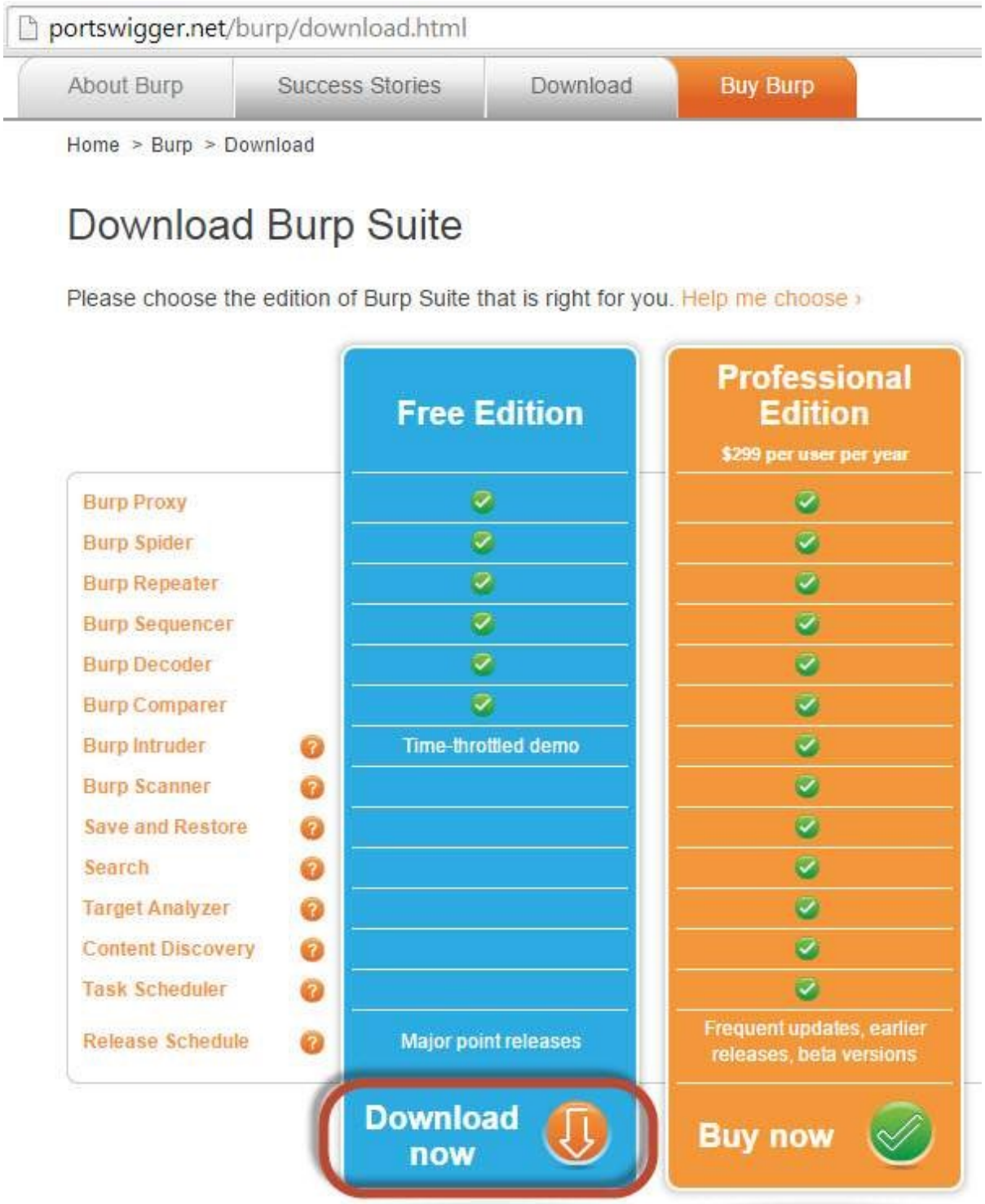


We can use the credentials of guest or admin as displayed in the login page.

Web Proxy

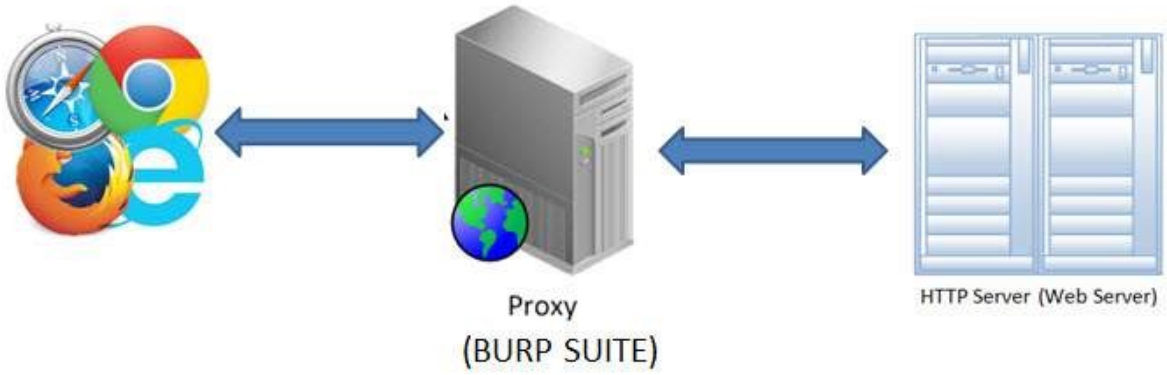
In order to intercept the traffic between client(Browser) and Server(System where Webgoat Application is hosted in our case), we will have to use a web proxy. We will use Burp Proxy and can be downloaded from http://portswigger.net/burp/download.html

It is sufficient to download the free version of burp suite as shown below.

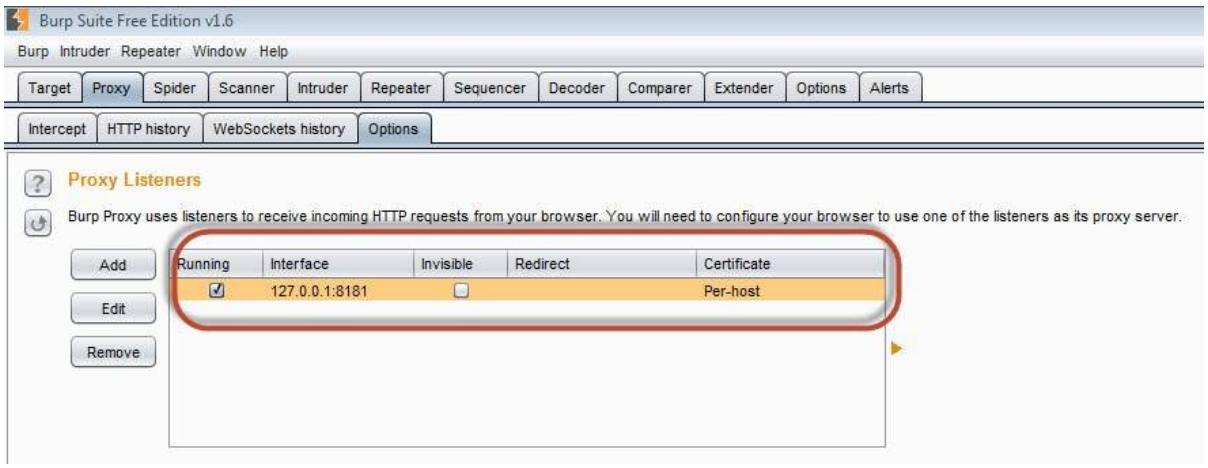


CONFIGURING Burp Suite

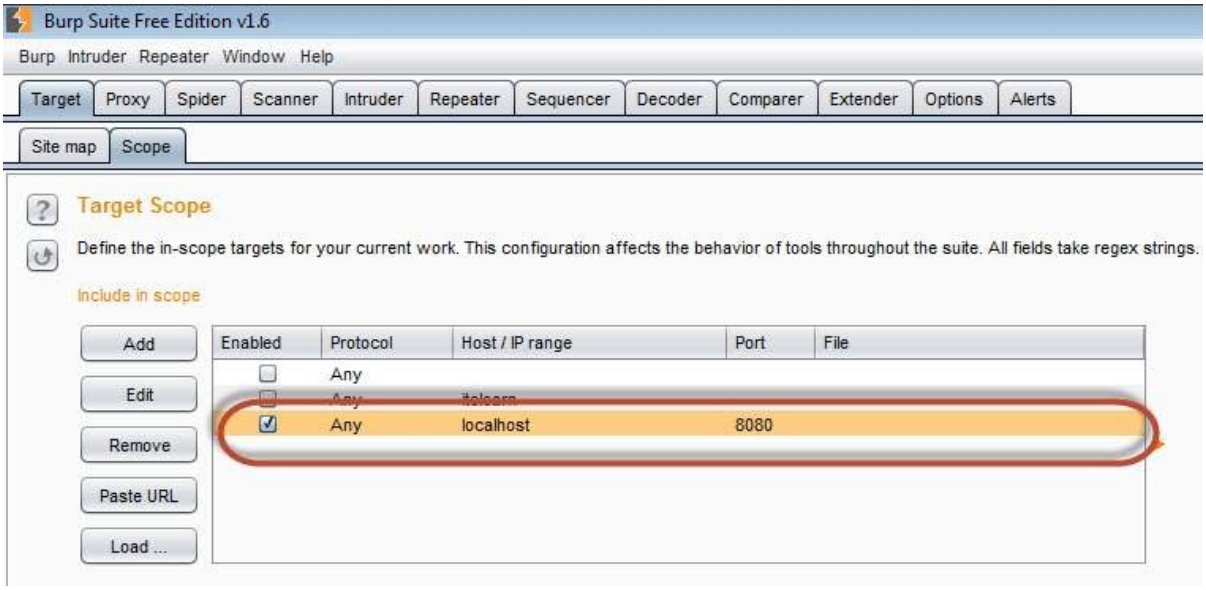
Burp Suite is a web proxy which can intercept each packet of information sent and received by the browser and webserver. This helps us to modify the contents before the client sends the information to the Web-Server.



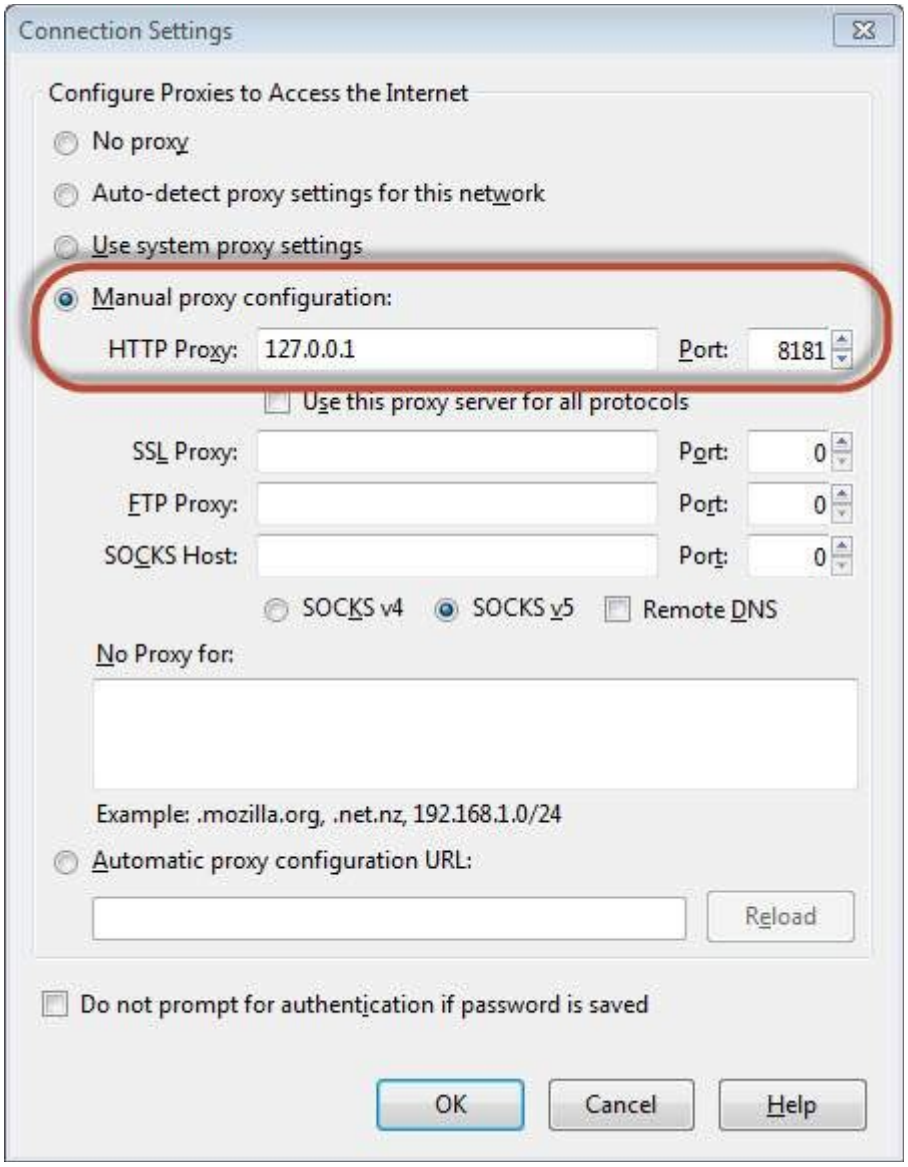
1. The App is installed on port 8080 and Burp is installed on port 8181 as shown below. Launch Burp suite and make the following settings inorder to bring it up in port 8181 as shown below.



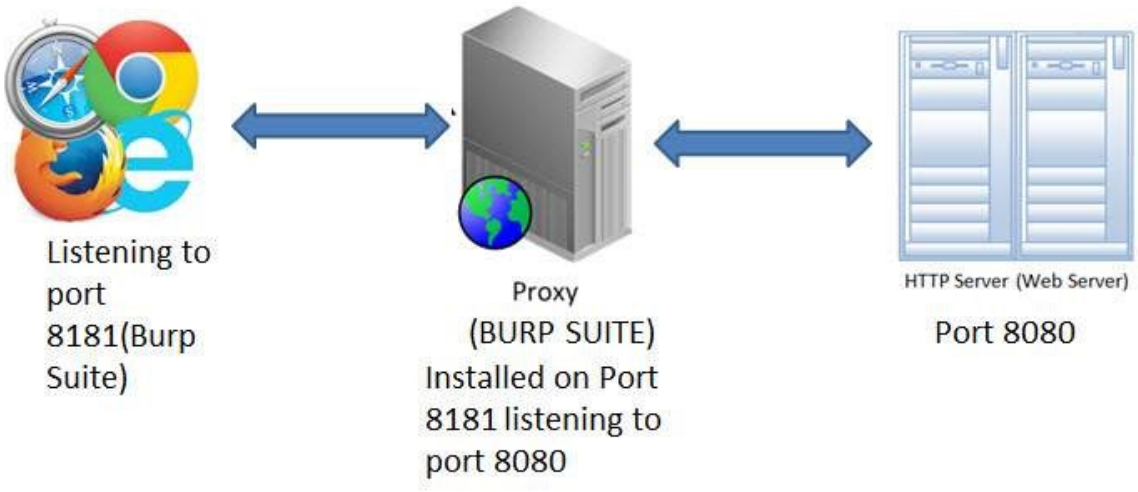
2. We Should ensure that the Burp is listening to Port#8080 where the application is installed so that Burp suite can intercept the traffic. This settings should be done on the scope tab of the Burp Suite as shown below.



3. Then make your browser proxy settings to listen to the port 8181 (Burp Suite port). Thus we have configured the Web proxy to intercept the traffic between client(browser) and the server(Webserver) as shown below



4. The Snapshot of the configuration is shown below with a help of a simple workflow diagram as shown below



Web Application - Injection

Injection technique consists of injecting a SQL query or a command using the input fields of the application. A successful SQL injection can read, modify sensitive data from the database and it can also to delete data from database. It also enables the hacker to perform administrative operations on the database such as shutdown the DBMS/dropping databases.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.

EXAMPLES

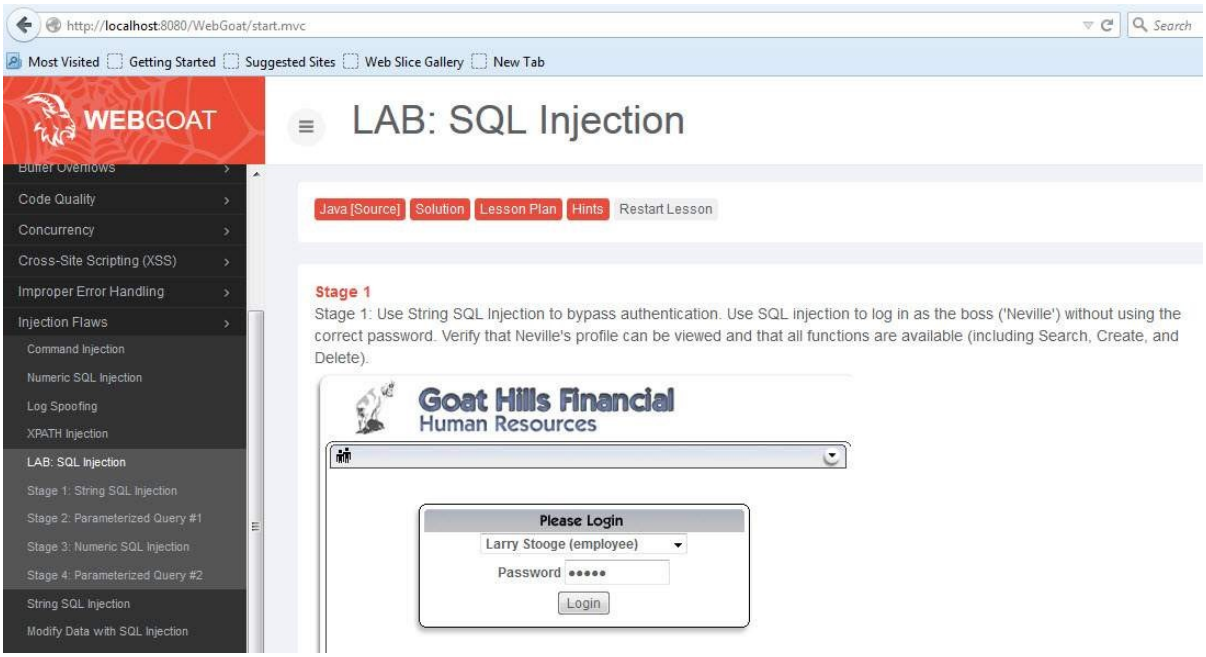
The application uses untrusted data in the construction of the

following vulnerable SQL call

```
String query = "SELECT * FROM EMP WHERE EMPID='" +
request.getParameter("id") + "'";
```

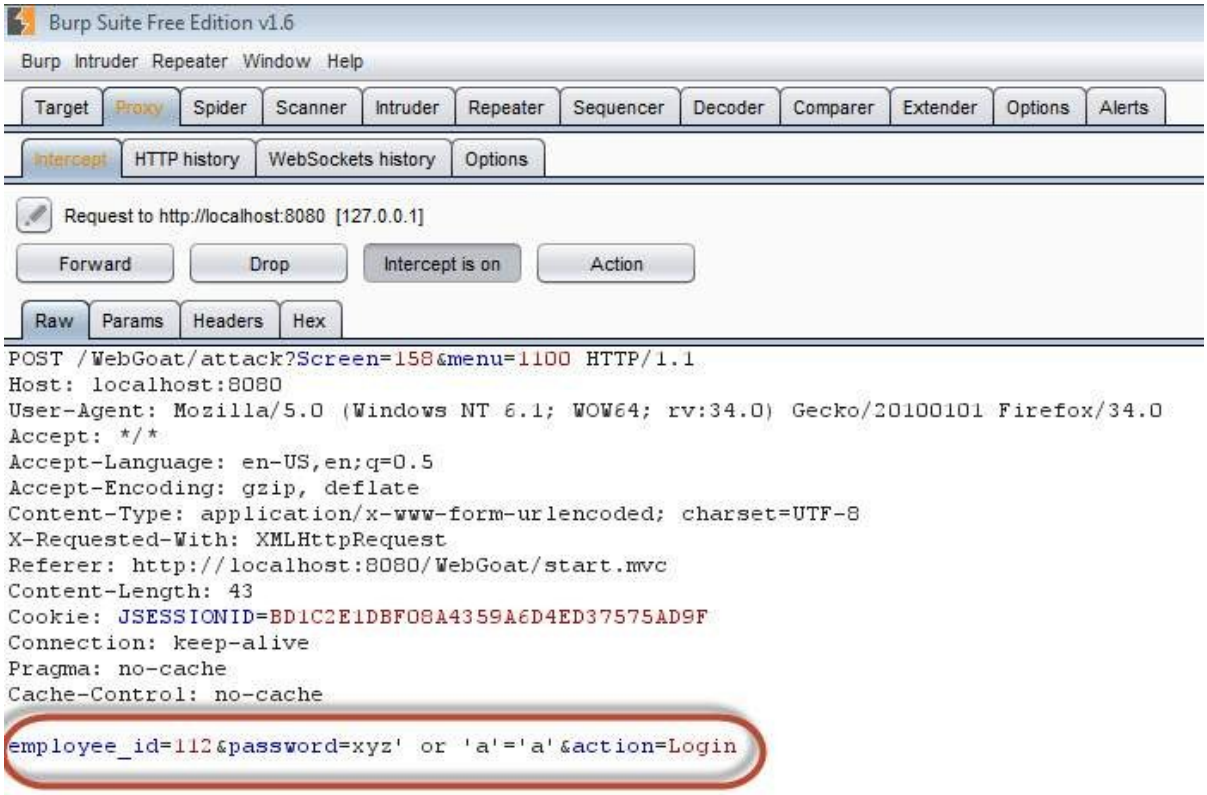
HANDS ON

1. Navigate to the SQL Injection area of the application as shown below.

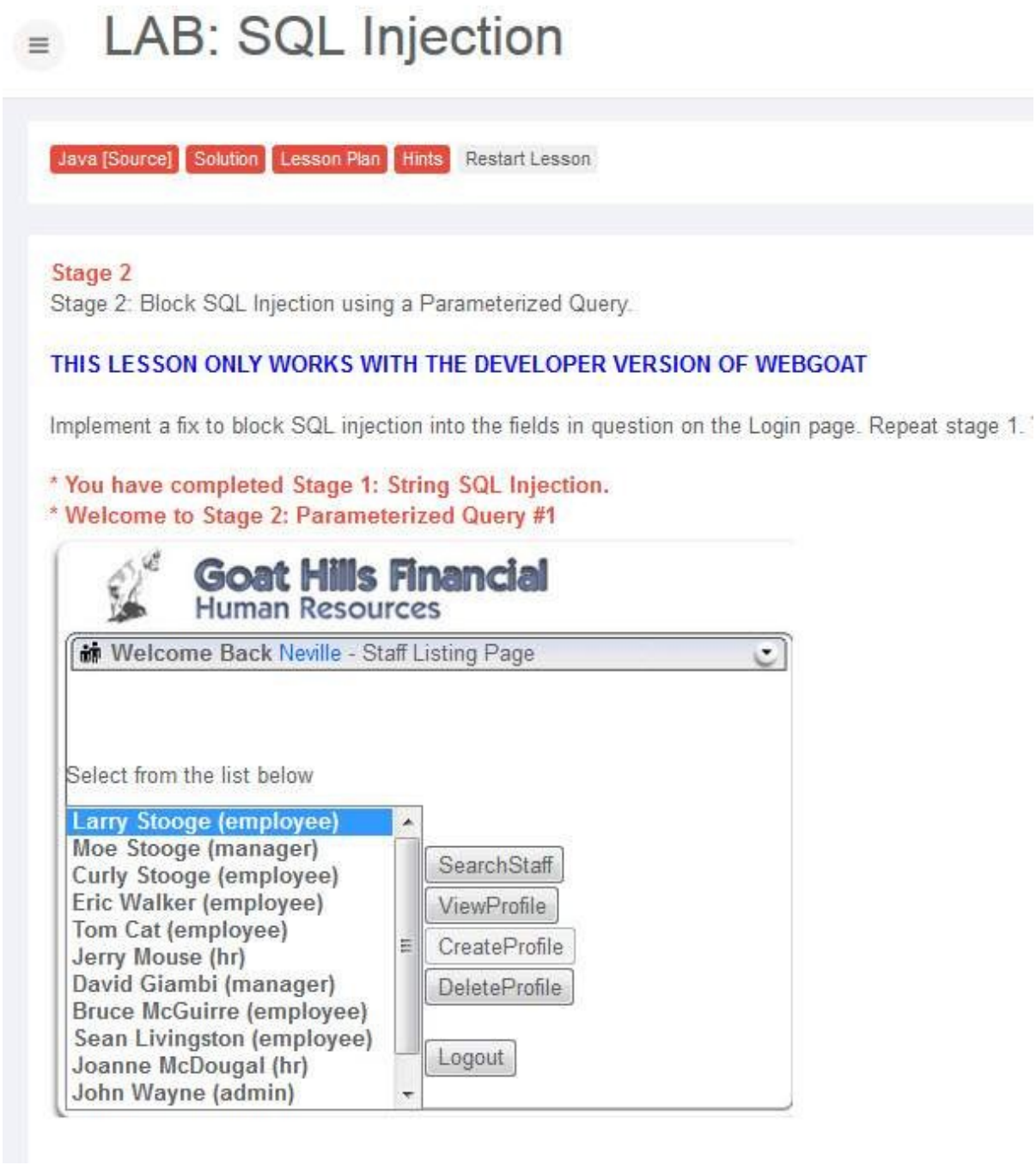


2. As given in the exercise, We will use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).

3. We will Inject a SQL such that we are able to bypass the password by sending the parameter as 'a'='a' or 1=1



4. Post Exploitation we are able to login as Neville who is the Admin as shown below.



Preventing SQL Injection

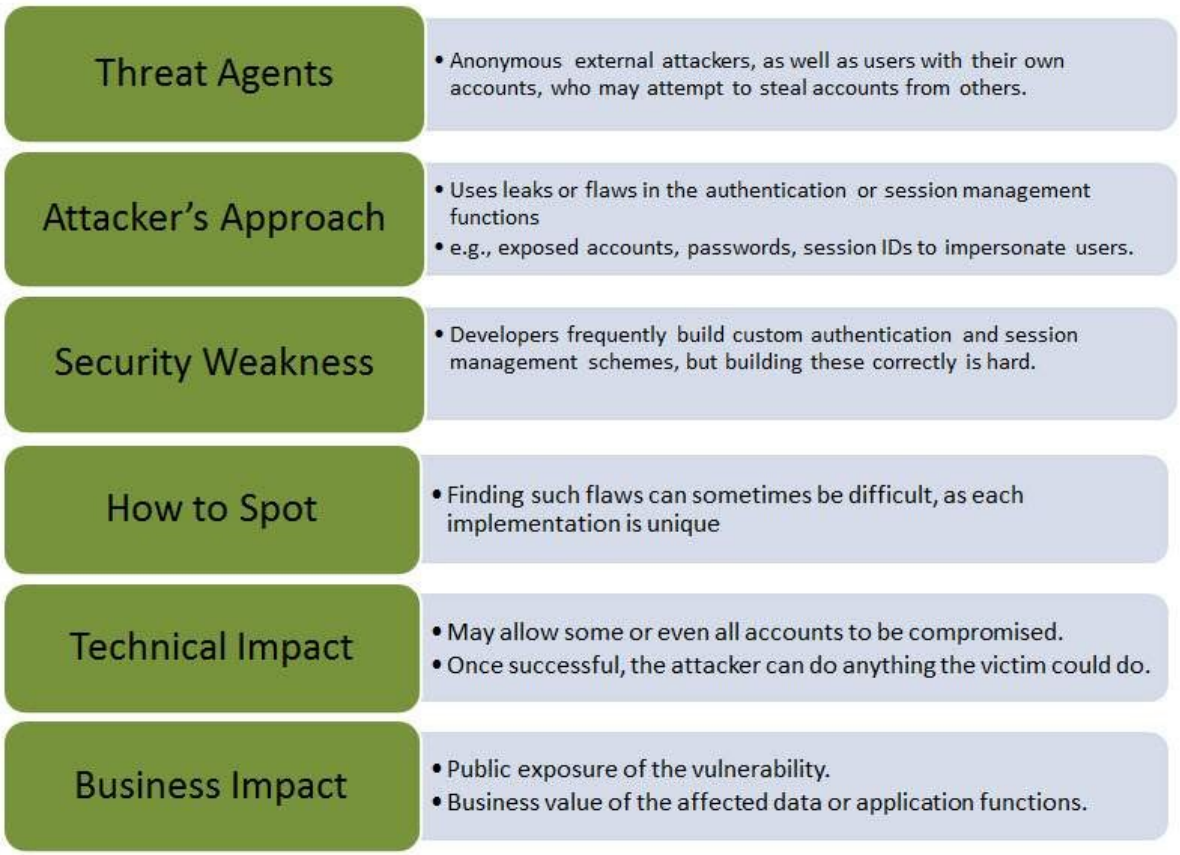
There are plenty of ways to prevent SQL injection. When developers write the code they should ensure that they handle special characters accordingly. There are cheat sheets/prevention techniques available from OWASP which is definitely a guide for developers.

- Using Parameterized Queries
- Escaping all User Supplied Input
- Enable Least Privilege for the database for the end users

Security Testing - Broken Authentication and Session Management Flaws

When authentication functions related to the application are NOT implemented correctly which will allow hackers to compromise passwords or session ID's or to exploit other implementation flaws using other users credentials.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



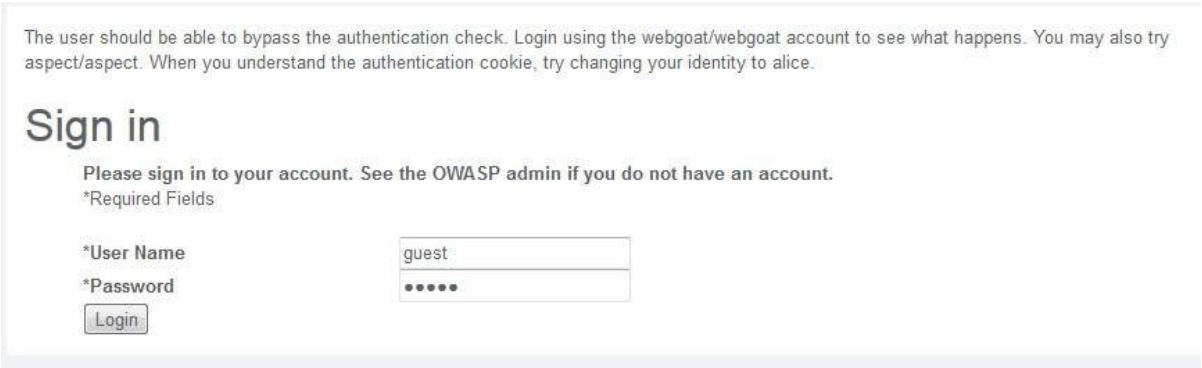
Example

```
An e-commerce application supports URL rewriting,
putting session IDs in the URL:


http://example.com/sale/saleitems
/jsessionid=2P0OC2JSNDLPSKHCJUN2JV/?item=laptop
```

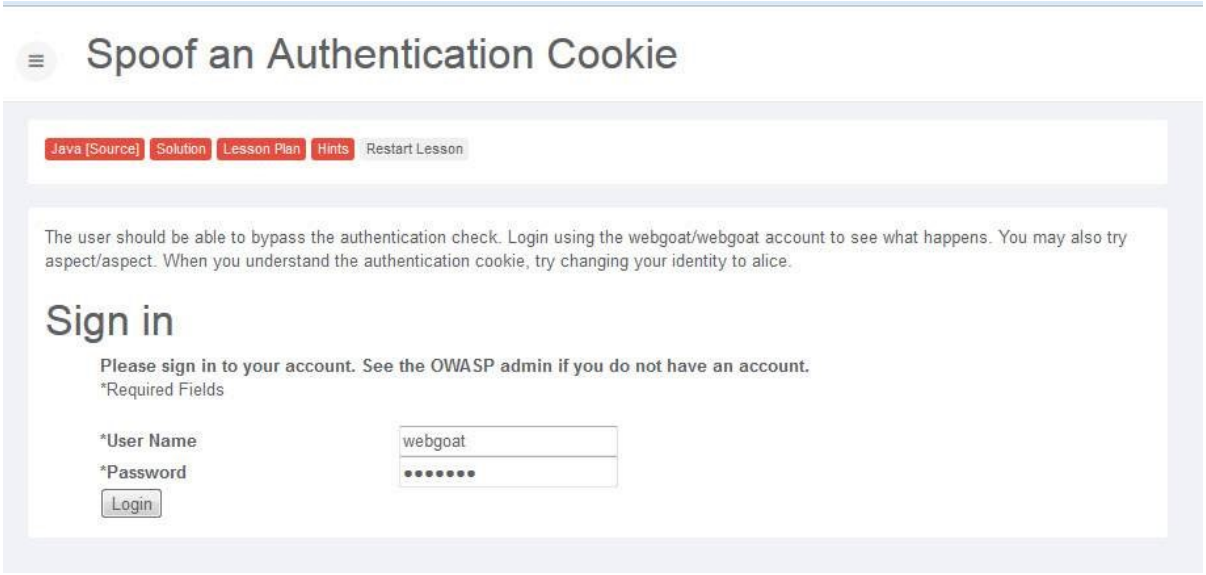
An authenticated user of the site forwards the URL to their friends to know about the discounted sales. He e-mails the above link without knowing that the user is also giving away the session ID's. When his friends use the link they will use his session and credit card.

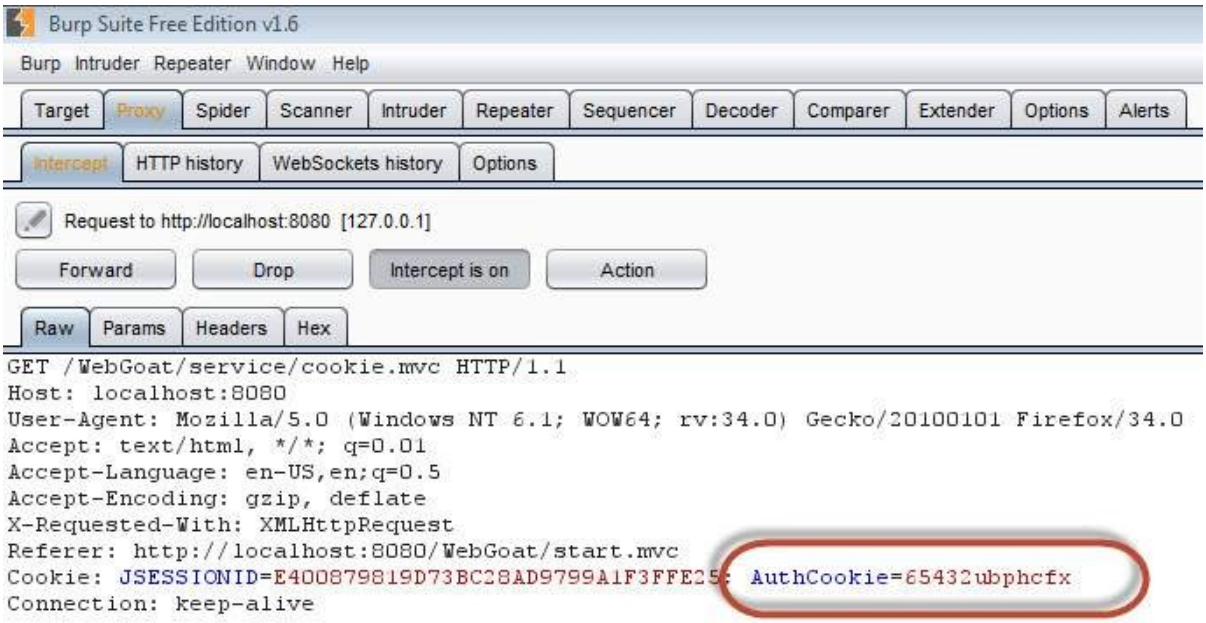
Hands ON

1. Login to Webgoat and navigate to 'Session Management Flaws' Section. Let us bypass the authetication by spoofing the cookie. Below is the snapshot of the scenario.

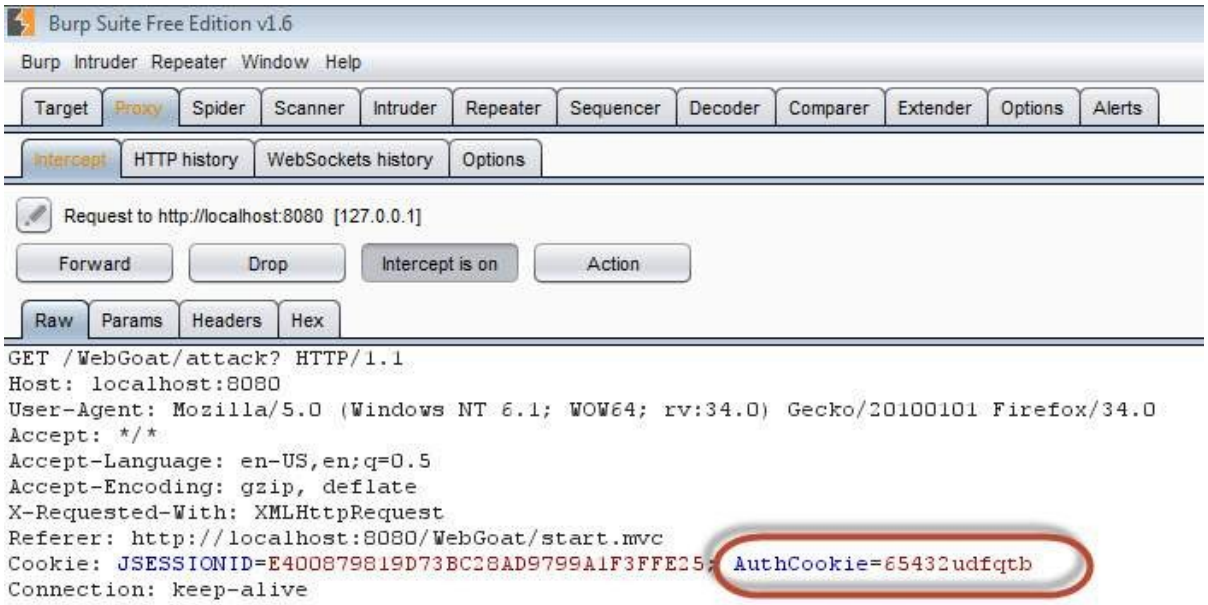


2. When we login using the credentials webgoat/webgoat, we find from Burp Suite that the JSESSION ID is C8F3177CCAFF380441ABF71090748F2E while the AuthCookie=65432ubphcfx upon successful authentication





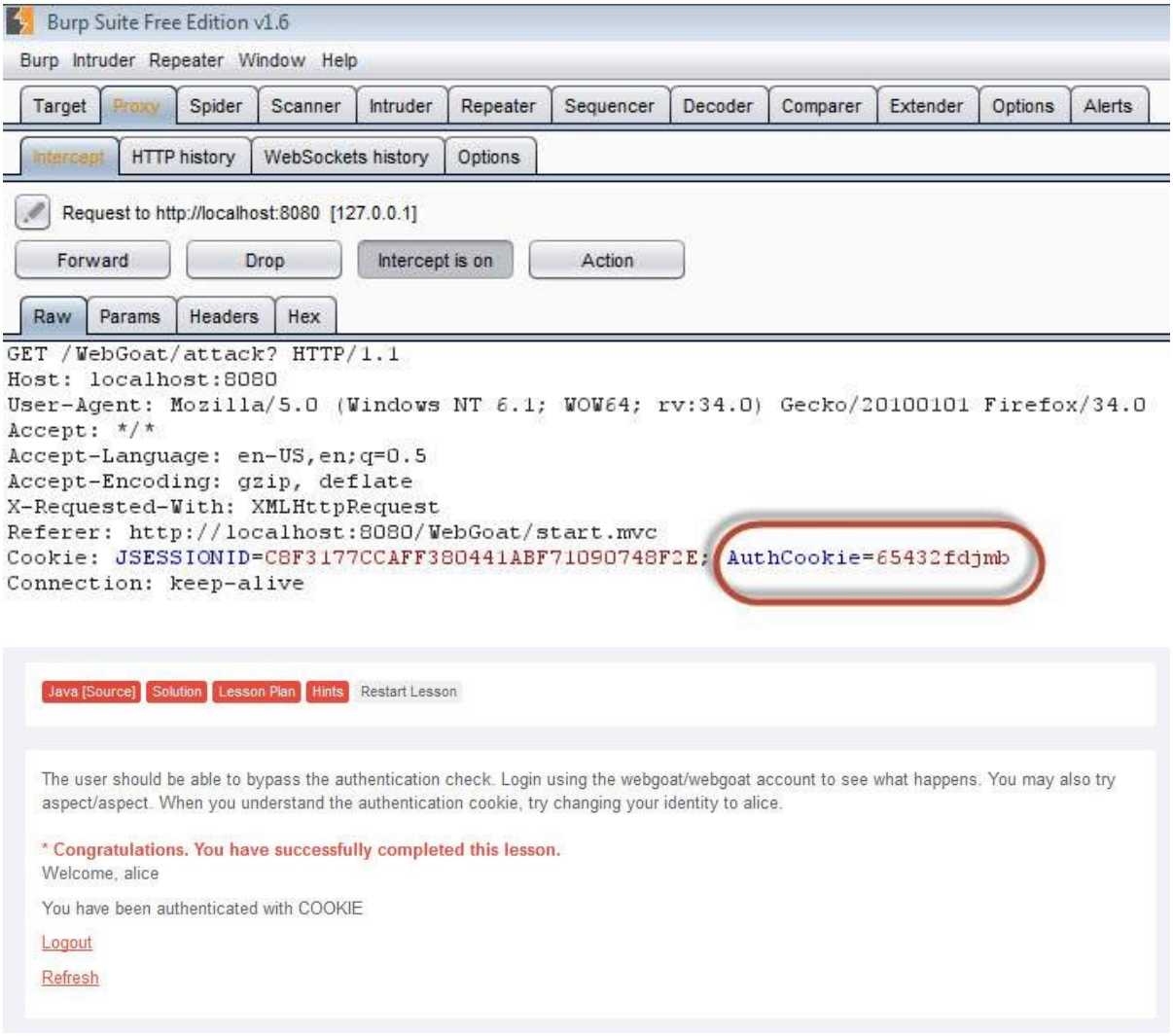
3. When we login using the credentials aspect/aspect, we find from Burp Suite that the JSESSION ID is C8F3177CCAFF380441ABF71090748F2E while the AuthCookie=65432udfqtb upon successful authentication.



4. Now we need to analyze the AuthCookie Patterns. The first half '65432' is common for both authentications. Hence we are now interested in analyzing the last part of the authcookie values viz-ubphcfx for webgoat user and udfqtb for aspect user respectively.

5. If we take a deep look at the auth cookie values, the last part is having the same lenght as that of user name. Hence it is evident that the username is used with some encryption method. Upon trial and errors/brute force mechanisms we find that the after reversing the user name, webgoat we end up with taogbew and then the before alphabet character is what being used as authcookie. i.e ubphcfx

6. If we pass this cookie value and let us see what happens. Upon authenticating as user webgoat, change the authcookie value to mock the user Alice by finding the authcookie for the same by performing step#4 and step#5.



Preventing Mechanisms

Develop a strong authentication and session management controls such that it meets all the authentication and session management requirements defined in OWASP's Application Security Verification Standard

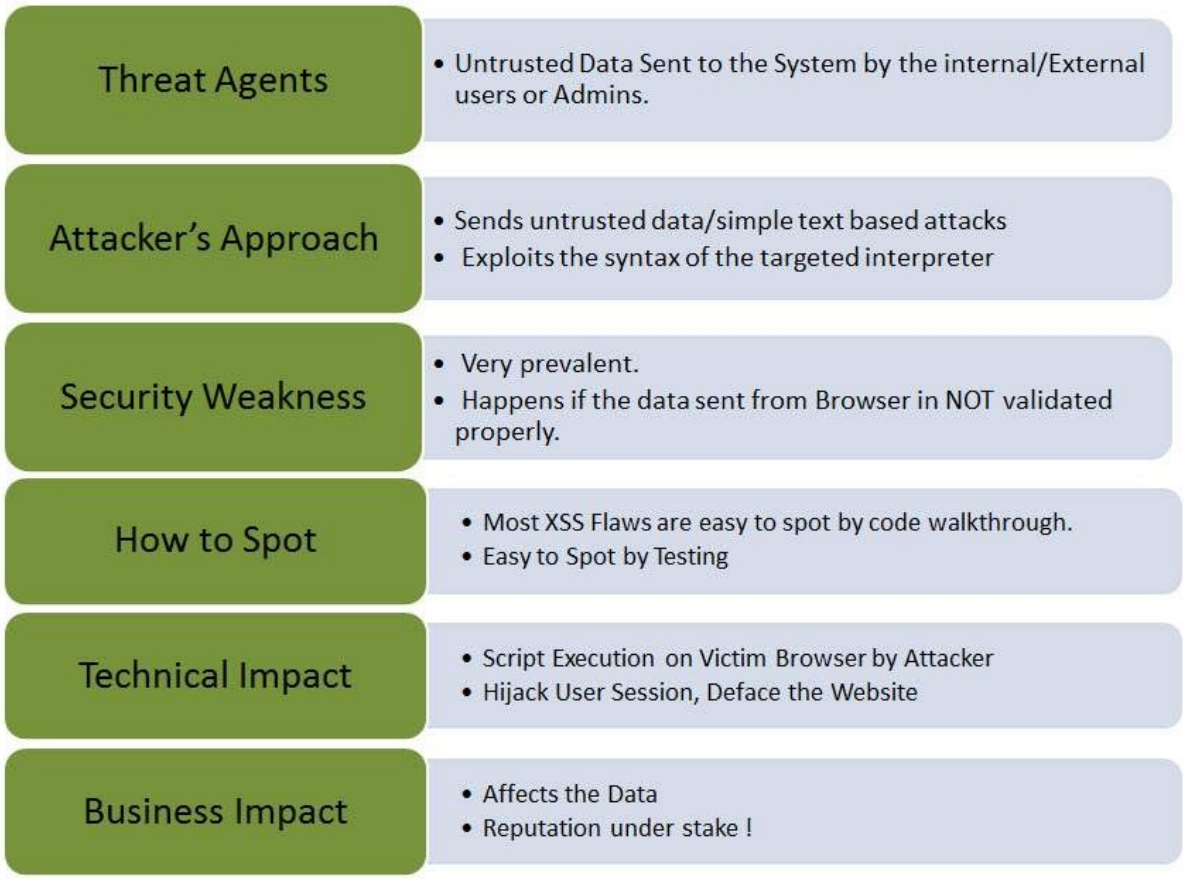
Dev should ensure that they avoid XSS flaws that can be used to steal session IDs.

Security Testing - Cross Site Scripting (XSS)

Cross Site Scripting(XSS) happens whenever an application takes untrusted data and sends it to the client(browser) without validation.

This allows attackers to execute malacious scripts in the victim's browser which can result in user sessions hijack, defacing web sites or redirect the user to malicious sites.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



Types of XSS

- **Stored XSS -** Stored XSS also known as persistent XSS occurs when user input is stored on the target server such as database/message forum/comment field etc. Then the victim is able to retrieve the stored data from the web application.
- **Reflected XSS -** Reflected XSS also known as non persistent XSS occurs when user input is immediately returned by a web

application in an error message/search result or the input provided by the user as part of the request and without permanently storing the user provided data.

- **DOM Based XSS -** DOM Based XSS is a form of XSS when the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser.

Example

The application uses untrusted data in the construction without validation. The special characters ought to be escaped.

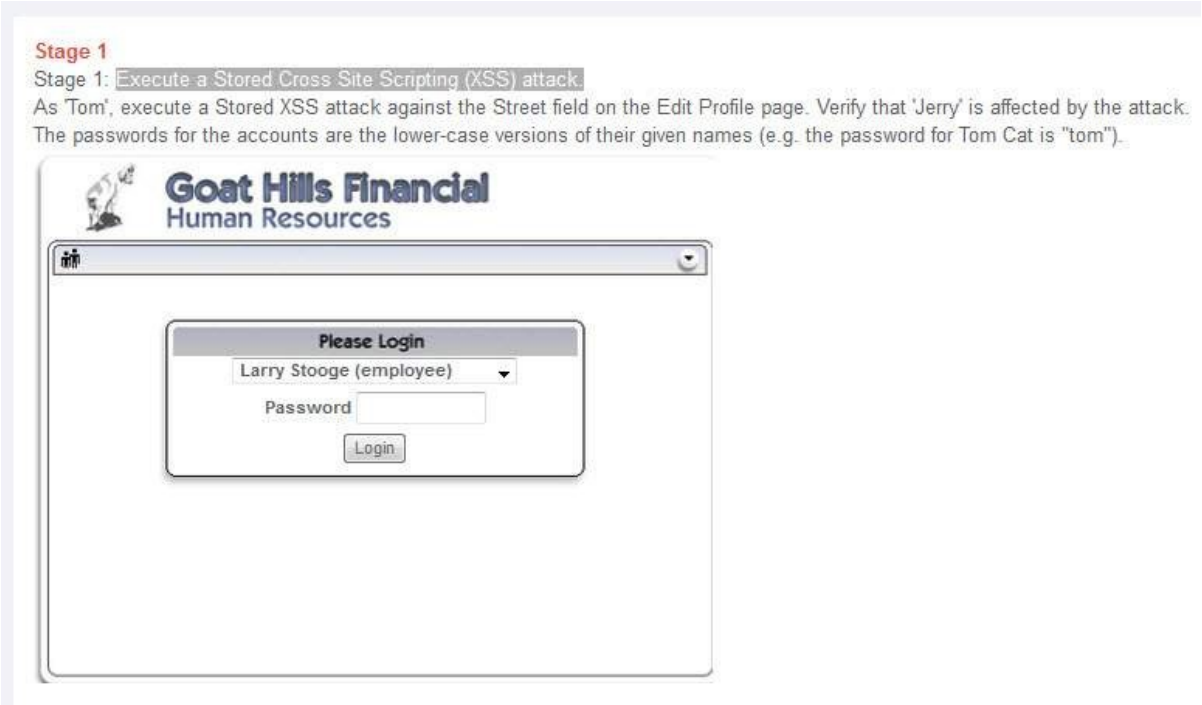
```
http://www.webpage.org/task/Rule1?query=try
```

The attacker modifies the query parameter in their browser to:

```
http://www.webpage.org/task/Rule1?query=<h3>Hello from XSS"</h3>
```

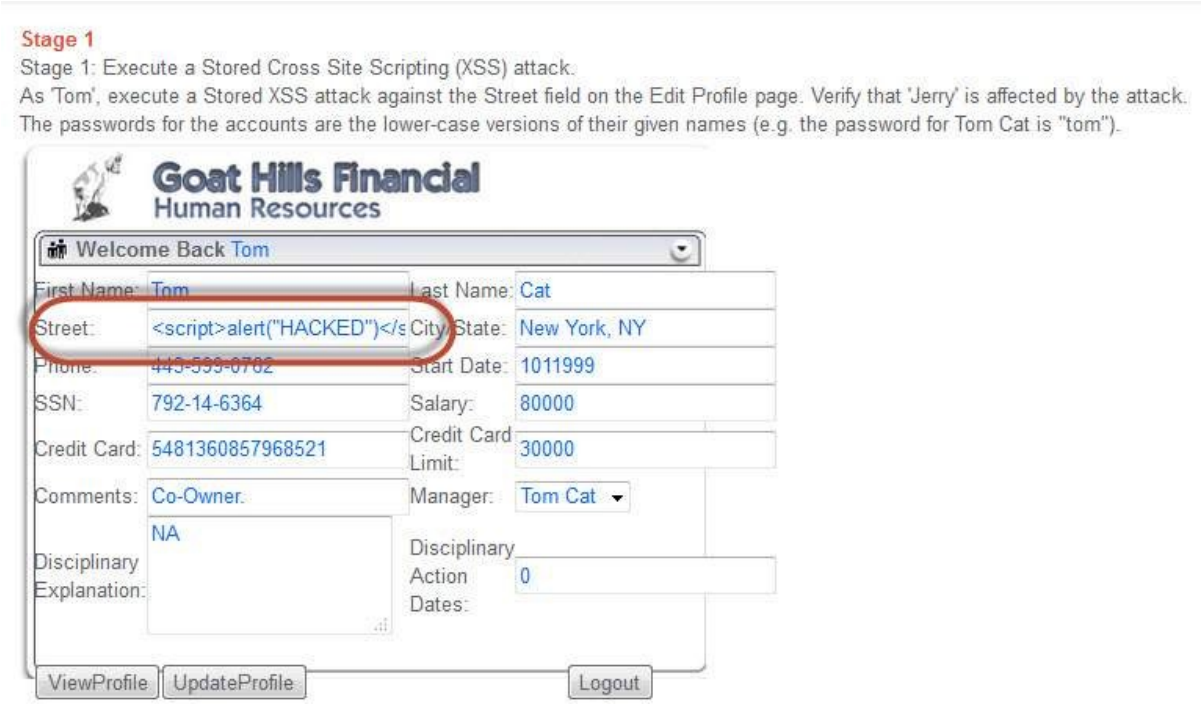
Hands ON

1. Login to Webgoat and navigate to cross site scripting(xss) Section. Let us execute a Stored Cross Site Scripting (XSS) attack. Below is the snapshot of the scenario.



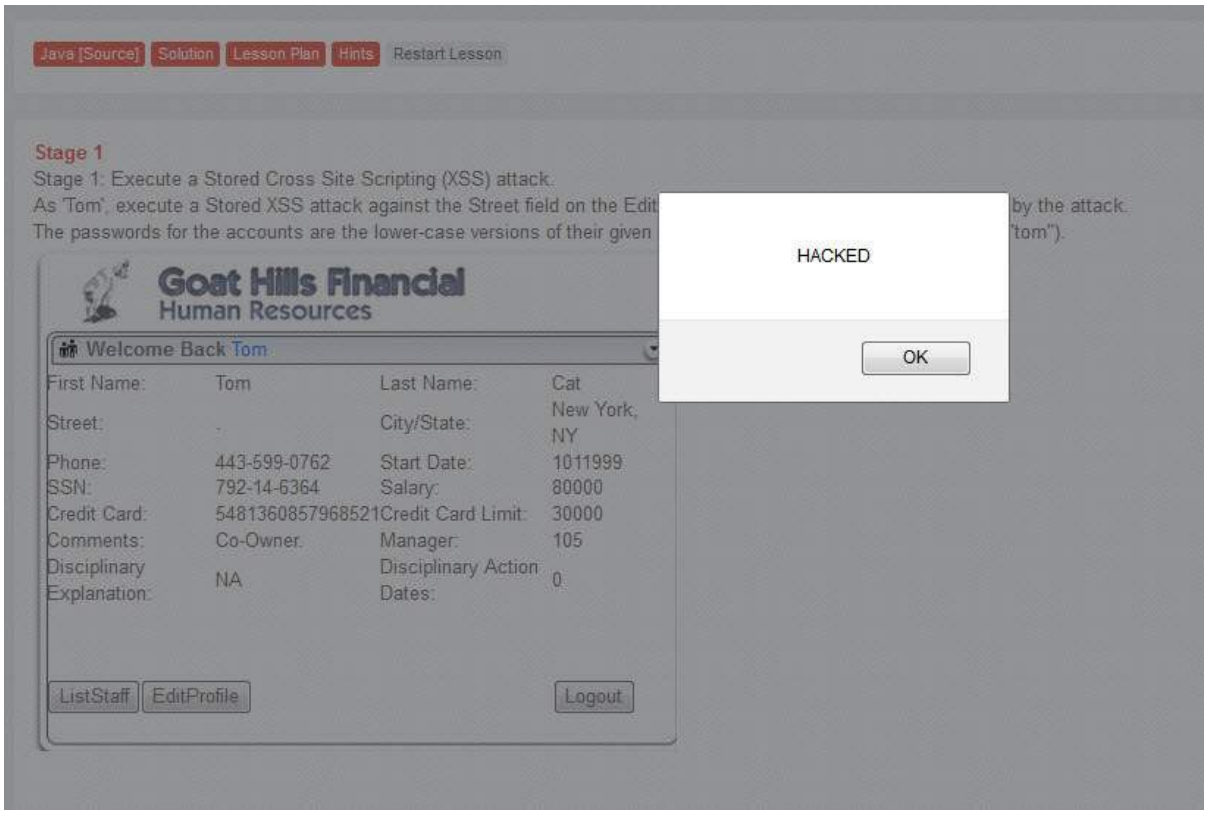
2. As per the scenario let us login as Tom with password 'tom' as mentioned in the scenario itself. Click 'view profile' and get into edit mode. Since tom is the attacker, let us inject java script into those edit boxes.

```
<script> alert("HACKED")</script>
```



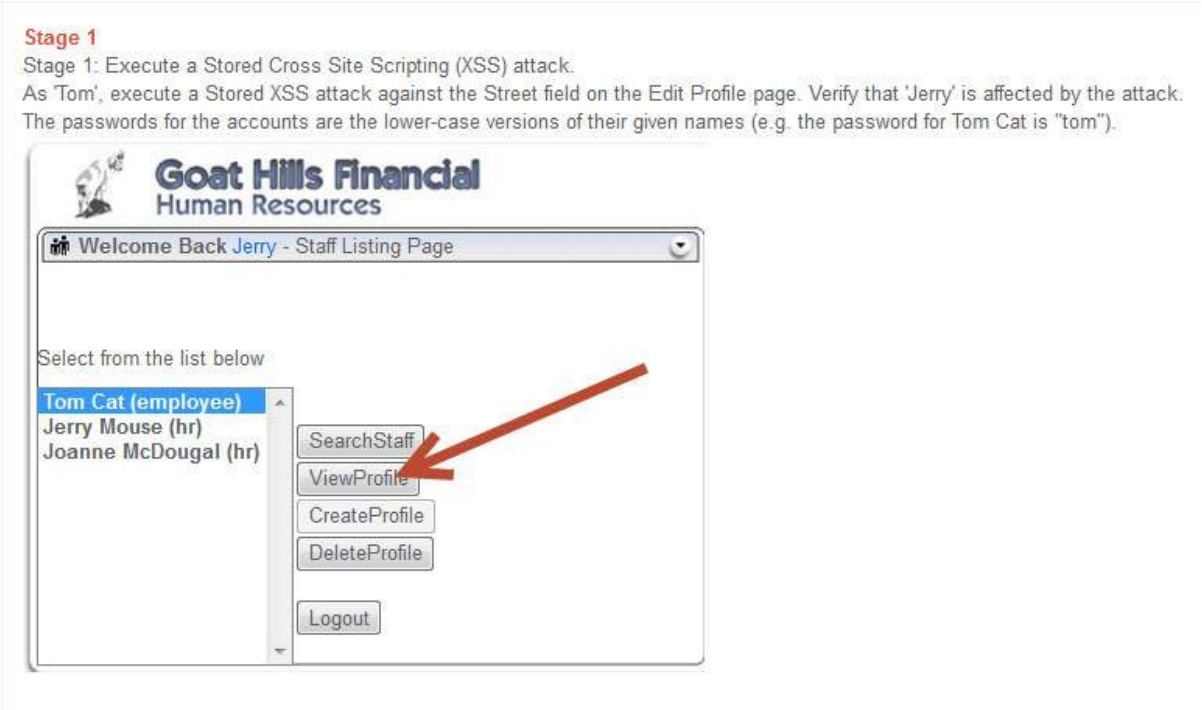
3. As soon as the update is over, tom receives an alert box with the

message "hacked" which means that the app is vulnerable.

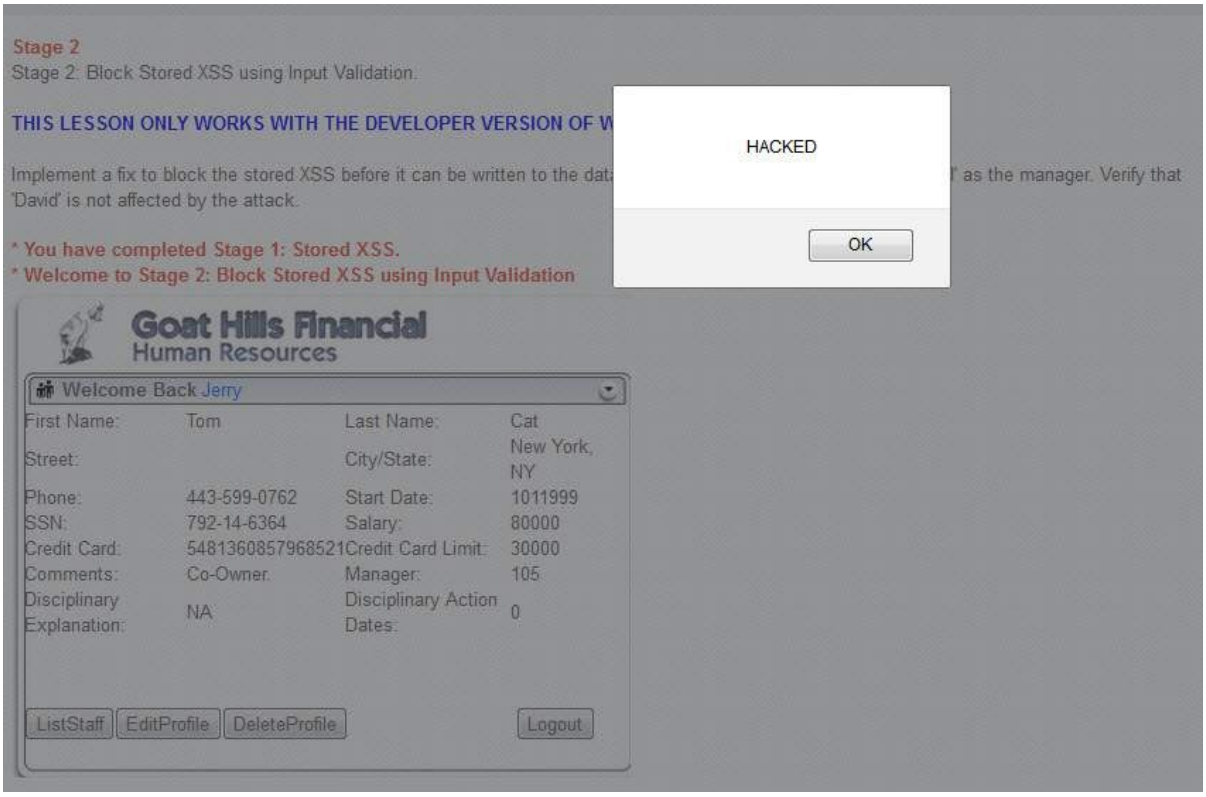


4. Now as per the scenario, we need to login as jerry (HR) and check if jerry is affected by the injected script.

5. After logging in as Jerry, select 'Tom' and click 'view profile' as shown below.



6. While viewing tom's profile from Jerry's account he is able to get the same message box.



7. This message box is just an example, but the actual attacker can perform much more than just displaying a message box.

Preventing Mechanisms

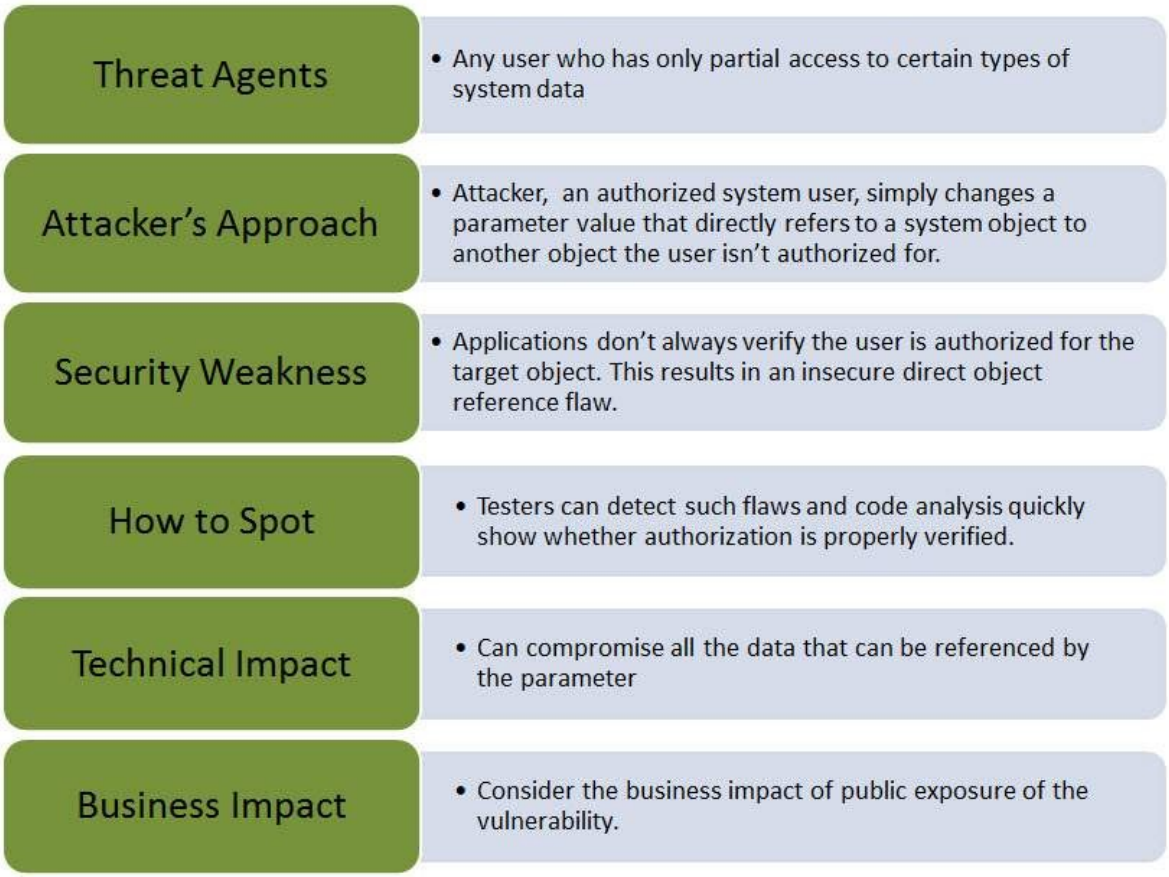
Developers has to ensure that they escape all untrusted data based on the HTML context such as body, attribute, JavaScript, CSS, or URL that the data will be placed into.

For the application that needs special characters as input, there should be robust validation mechanisms in place before accepting them as valid inputs.

Security Testing - Insecure Direct Object References

A direct object reference is likely to occur when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key without any validation mechanism which will allow attackers to manipulate these references to access unauthorized data.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



Example

The App uses unverified data in a SQL call that is accessing account information.

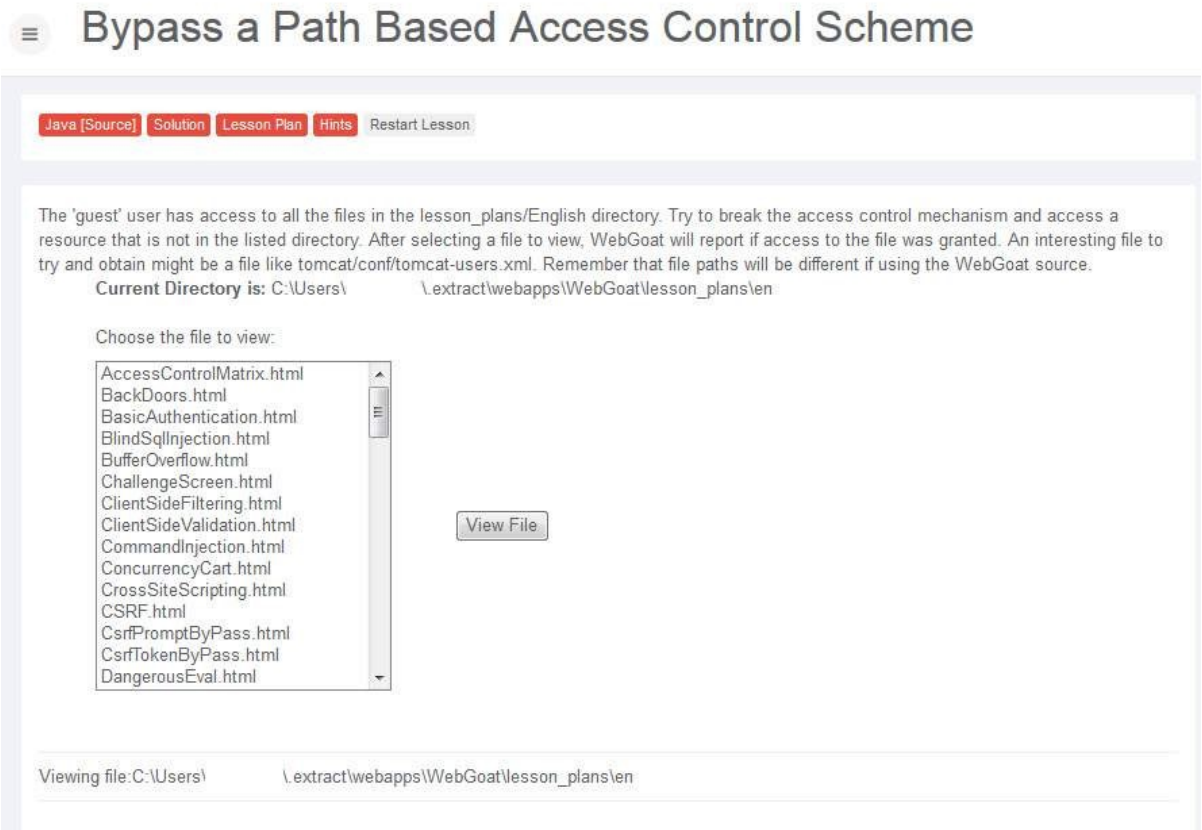
```
String sqlquery = "SELECT * FROM useraccounts
WHERE account = ?";
PreparedStatement st =
connection.prepareStatement(sqlquery , ? );
st.setString( 1, request.getParameter("acct"));
ResultSet results = st.executeQuery( );
```

The attacker modifies the query parameter in their browser to point to Admin.

```
http://webapp.com/app/accountInfo?acct=admin
```

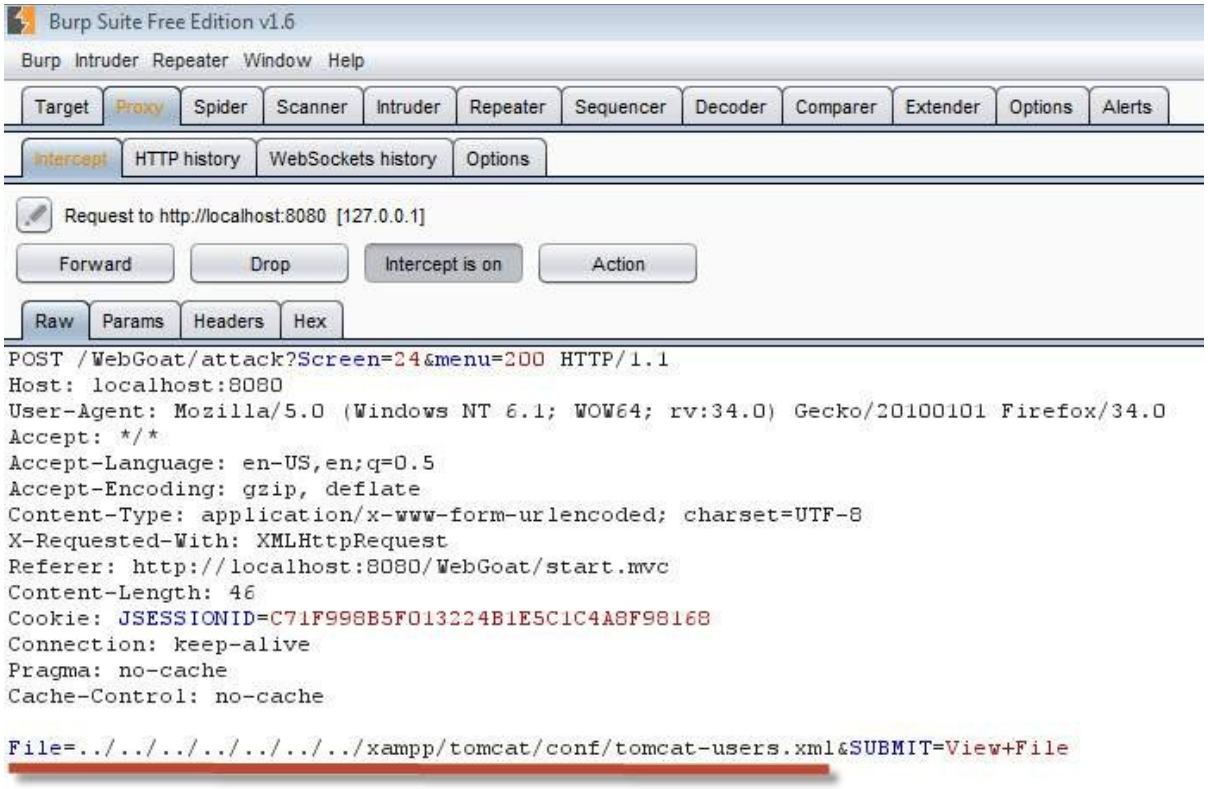
Hands ON

1. Login to Webgoat and navigate to access control flaws Section. The goal is to retrieve the tomcat-users.xml by navigating to the path where it is located. Below is the snapshot of the scenario.

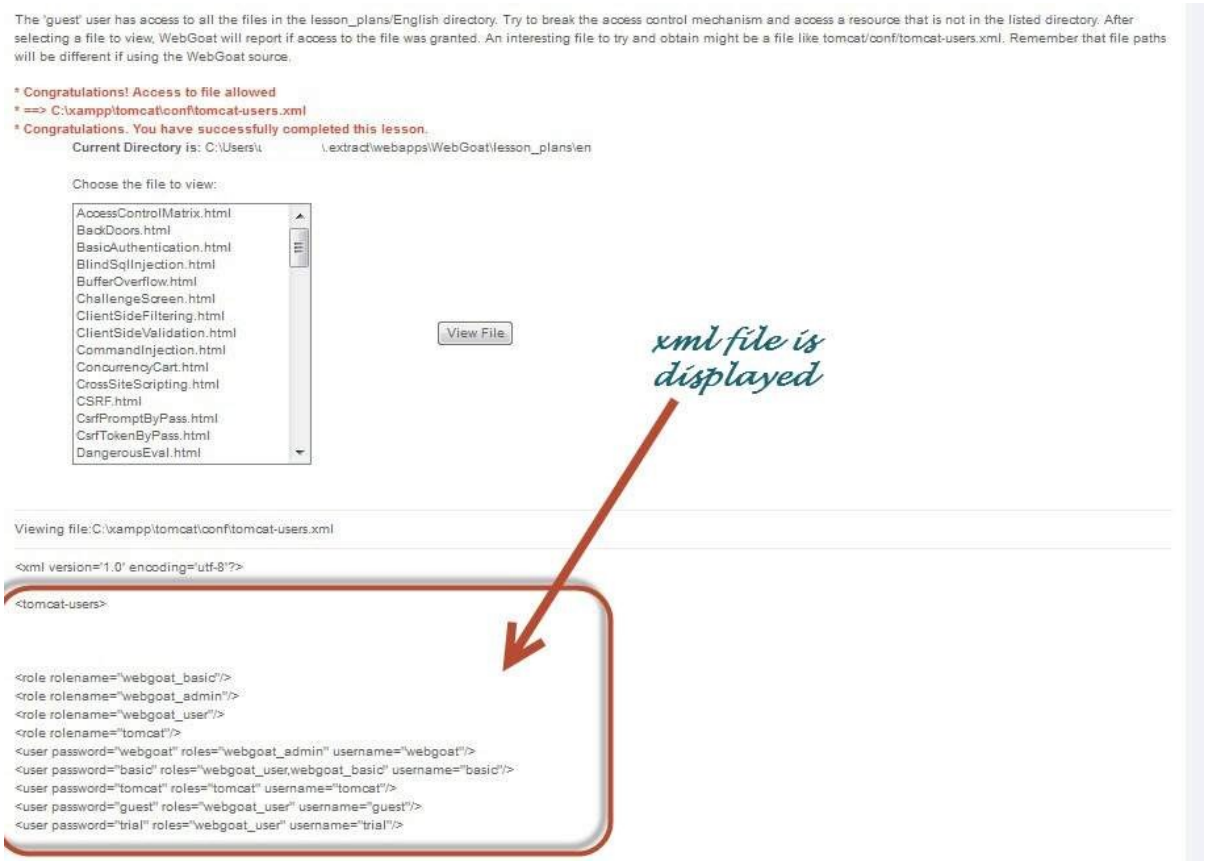


2. The path of the file is displayed in 'the current directory is' field - C:\Users\userName$\.extract\webapps\WebGoat\lesson_plans\en and we also know that the tomcat-users.xml file is kept under C:\xampp\tomcat\conf

3. So we need to traverse all the way out of the current directory and navigate from C:\ Drive. We can perform the same by intercepting the traffic using Burp Suite.



4. If the attempt is successful, it would display the tomcat-users.xml with the message " Congratulations. You have successfully completed this lesson."



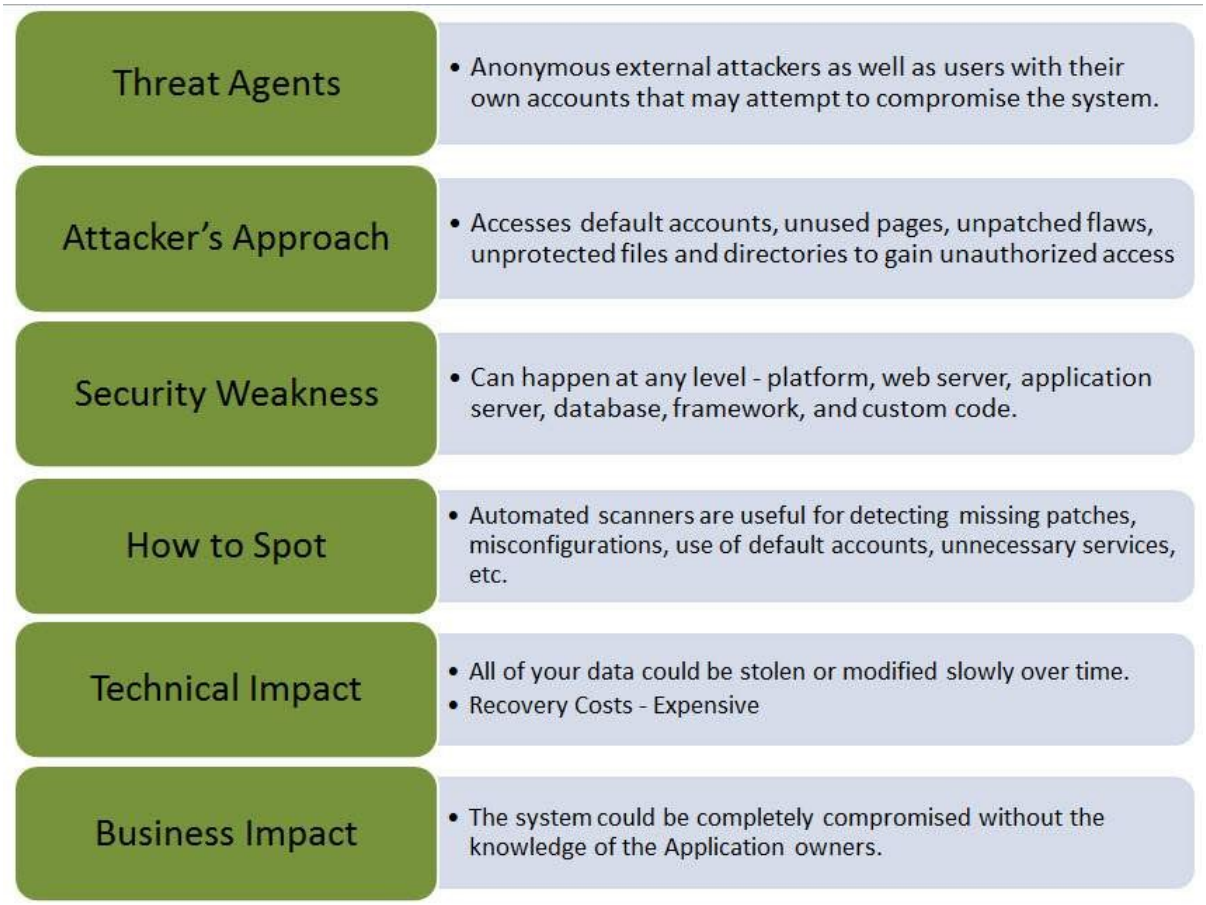
Preventing Mechanisms

Dev can use the below resources/points as a guide to prevent insecure direct object reference during development phase itself.

- Developers should Use only one user or session for indirect object references.
- It is also recommended to check the access before using a direct object reference from an untrusted source.

Security Testing - Security Misconfiguration

Security Misconfiguration arises when Security settings are defined, implemented, and maintained as defaults. Good security requires a secure configuration defined and deployed for the application, web server, database server, and platform. It is equally important to have the software up to date.



Example

Below are some of the classic examples of security misconfiguration :

- If Directory listing is not disabled on the server and if attacker discovers the same then the attacker can simply list directories to find any file and execute it. It is also possible to get the actual code base which contains all your custom code and then to find a serious flaws in the application.
- App server configuration allows stack traces to be returned to users, potentially exposing underlying flaws. Attackers grab those extra information that the error messages provides which is enough for them to penetrate.
- App servers usually comes with sample apps that are NOT well

secured. If not removed from production server would result in compromising your server.

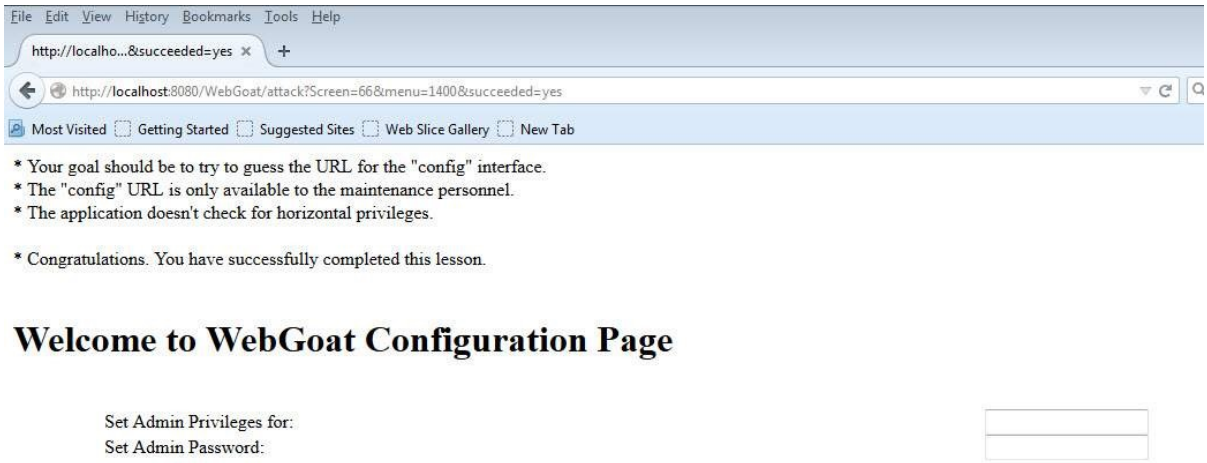
Hands ON

1. Launch Webgoat and navigate to Insecure configuration section and let us try to solve that challenge. Snapshot of the same is provided below:

2. We can try out as many options as we can think of. All we need to find the URL of config file and we all know developers follow kind of naming convention for config files. It can be anything that is listed below. It is usually done by BRUTE force technique.

- **web.config**
- **config**
- **appname.config**
- **conf**

3. Upon trying various options, we find that 'http://localhost:8080 /WebGoat/conf' is successful. The below page is displayed if the attempt is successful



Preventing Mechanisms

All environments such Development, QA, and production environments should all be configured identically using different passwords used in each environment that cannot be hacked easily.

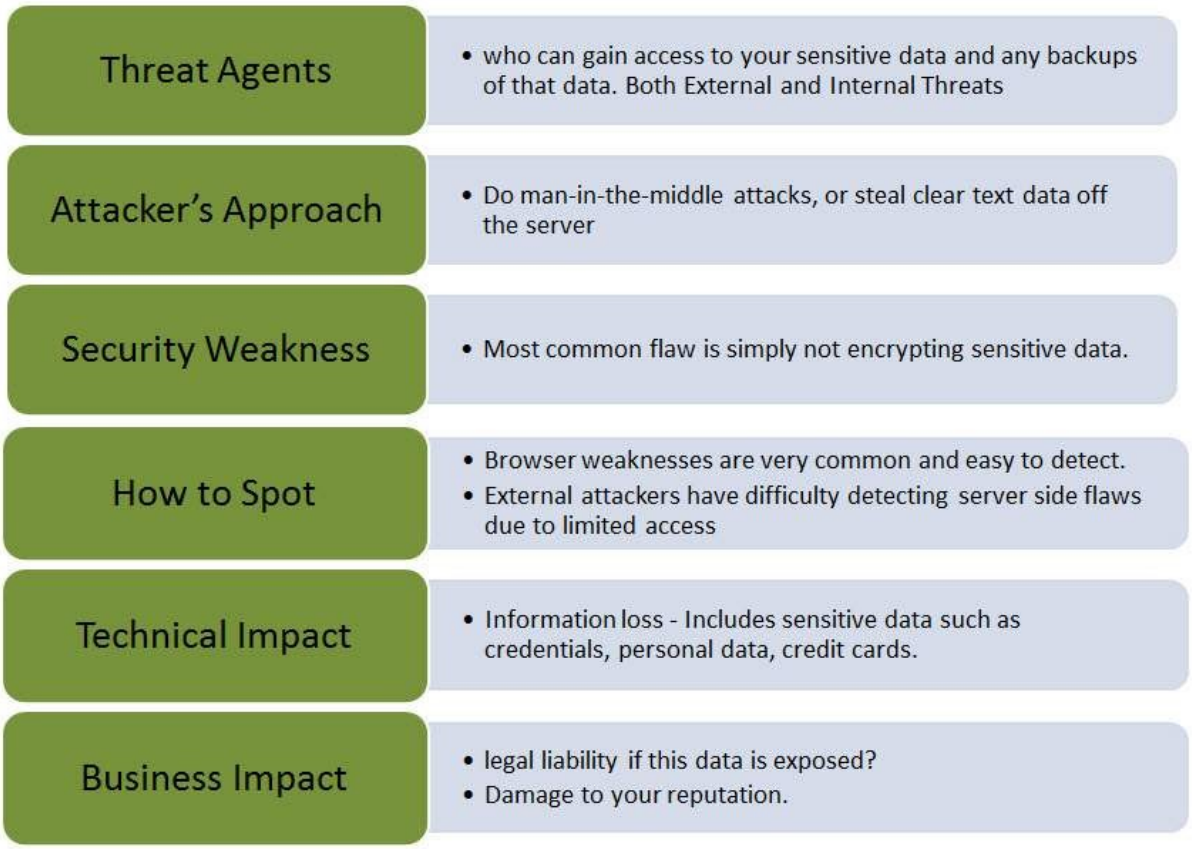
Ensure that a strong application architecture is being adopted that provides effective, secure separation between components

It can also minimize the possiblity of this attack by running automated scans and doing audits periodically .

Security Testing - Sensitive Data Exposure

As the online application keep flooding in day by day, not all applications are secured. Many web applications do not properly protect sensitive user data such as credit cards information/Bank account info/authentication credentials. Hackers might end up stealing those weakly protected data to conduct credit card fraud, identity theft, or other crimes.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



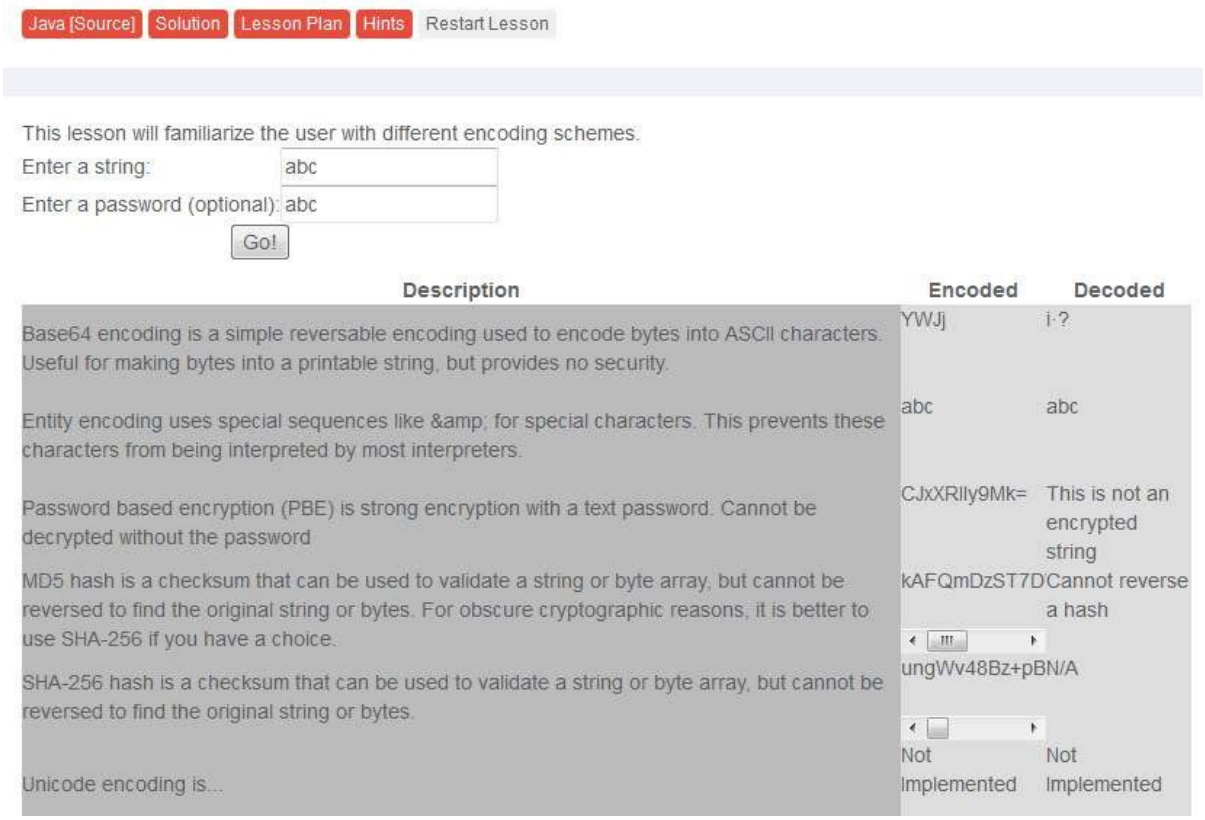
Example

Below are some of the classic examples of security misconfiguration :

- A site simply doesn't use SSL for all authenticated pages. This will enable an attacker to monitor network traffic and steal the users session cookie to hijacks the users session or accessing their private data.
- An application stores the credit card numbers in an encrypted format in a database. Upon retrieval those are decrypted allowing the hacker to perform a SQL injection attack to retrieve all sensitive info in a clear text. This can be avoided by encrypting the credit card numbers using a public key and allowed back-end applications to decrypt them with the private key.

Hands ON

1 . Launch WebGoat and navigate to "Insecure Storage" Section. Snapshot of the same is displayed below.



2 . Enter the username and password. Its time to learn different kind of encoding and encryption methodologies that we discussed previously. More on encoding and encryption, please refer to their corresponding chapters.

Preventing Mechanisms

It is NOT advised to store sensitive data unnecessarily and should be scraped as soon as possible if not required.

It is important to ensure that we incorporate strong and standard encryption algorithms are used and proper key management is in place.

It can also be avoided by Disabling autocomplete on forms that collect sensitive data such as password and disable caching for pages that contain sensitive data.

Missing Function Level Access Control

Most of the web applications verify function level access rights before making that functionality accessible to the user, However, if the same access control checks are NOT performed on the server, hackers will be able to penetrate into the application without proper authorization.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.

Example

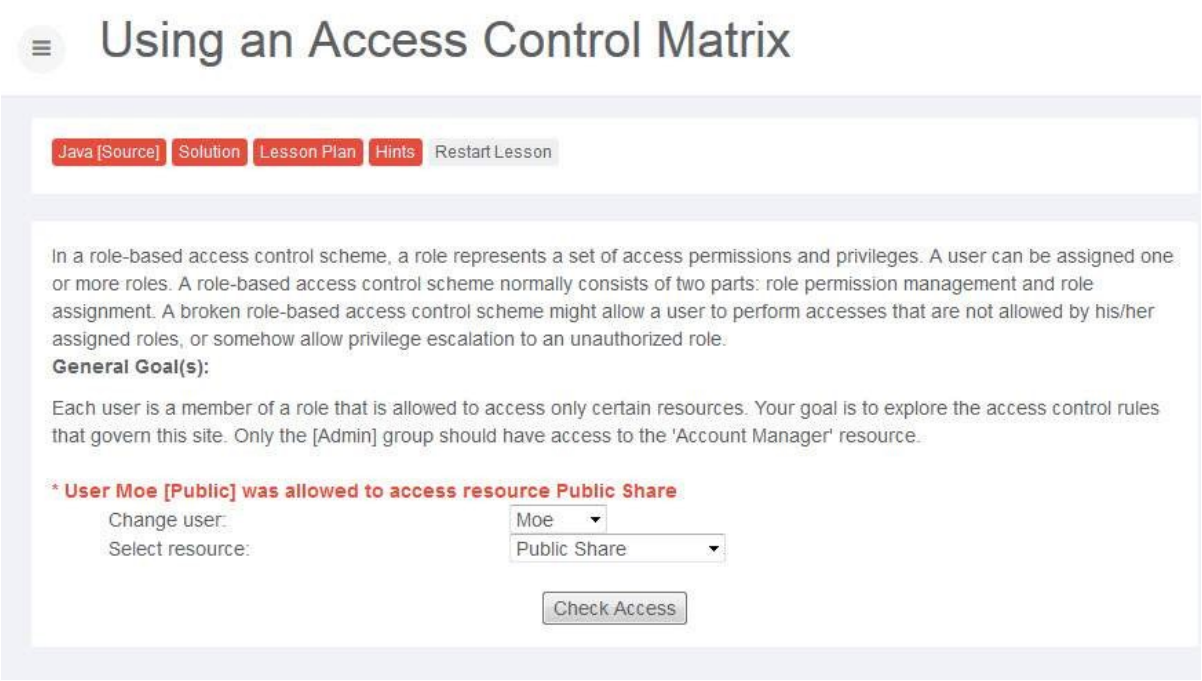
Below is a classic example of Missing Function Level Access Control :

The hacker simply forces target URLs. Usually admin access requires authentication, however, if the application access is NOT verified an unauthenticated user can access admin page.

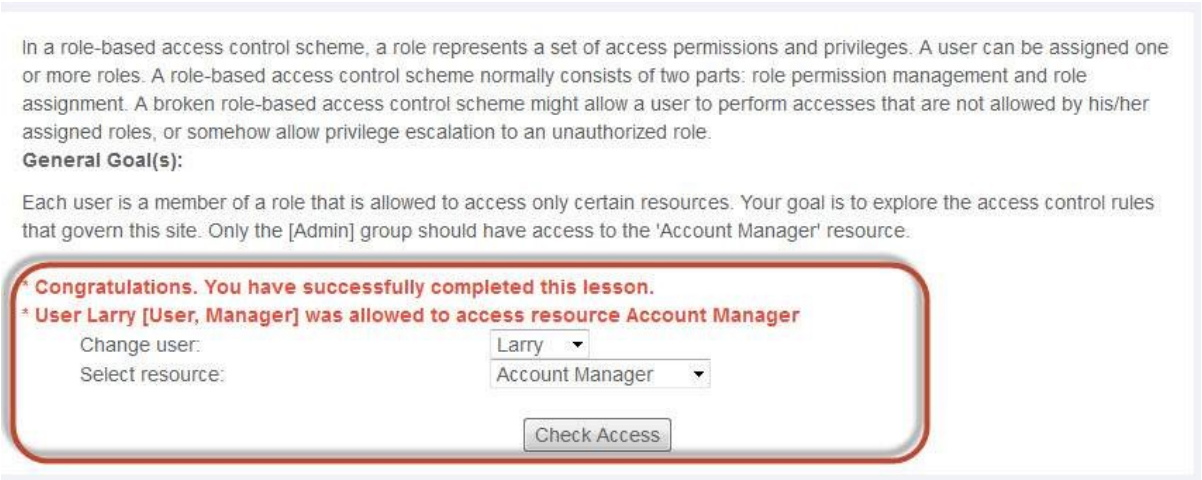
```
' Below URL might be accessible to an
authenticated user
http://website.com/app/standarduserpage

' A NON Admin user is able to access admin page
without authorization.
http://website.com/app/admin_page
```

Hands ON

1 . Let us login as account manager by first going through the list of users and their access previleges.



2 . Upon trying various combinations we can find it out that Larry has access to resource account manager.



Preventing Mechanisms

The authentication mechanism should deny all access by default, and provide access to specific roles for every function

In a workflow based application, verify the user state before allowing them to access any resources.

Cross-Site Request Forgery(CSRF)

A CSRF attack forces an authenticated user(victim) to send a forged HTTP request, including the victim's session cookie to a vulnerable web application which allows the attacker to force the victim's browser to generate request such that the vulnerable app perceives as legitimate requests from the victim.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.

Example

Below is a classic example of CSRF :

1. Let us say, the vulnerable app sends a state changing request as a plain text without any encryption.

```
http://bankx.com/app?action=transferFund&
```

```
amount=3500&destinationAccount=4673243243
```

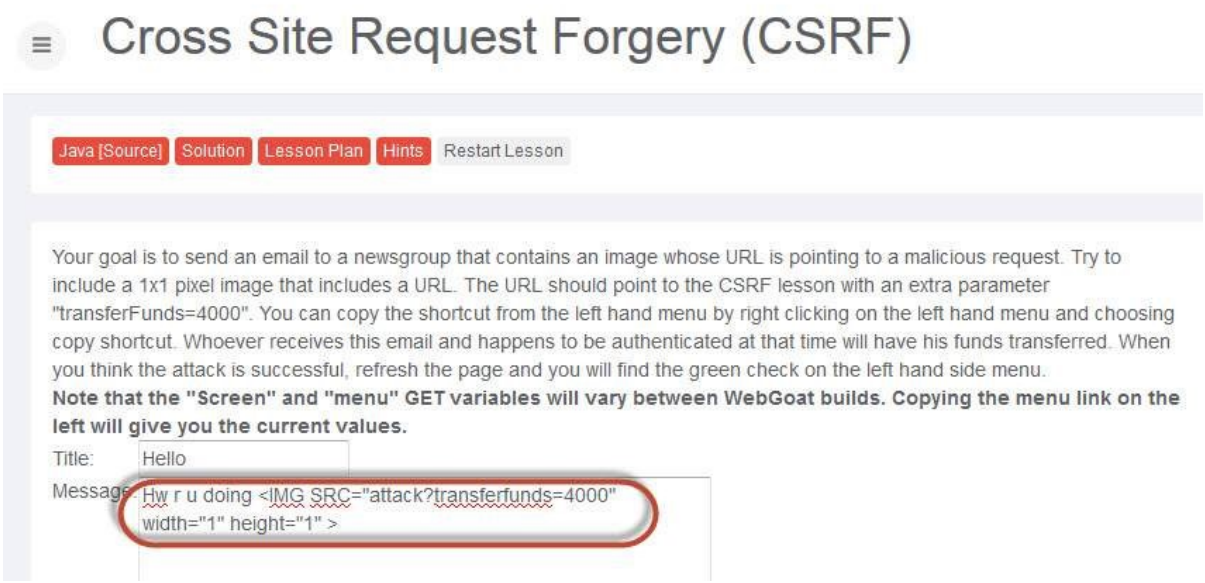
2. Now the hacker constructs a request that will transfer money from the victim's account to the attacker's account by embedding the request in an image that is stored on various sites under the attacker's control:

```
<img src="http://bankx.com
/app?action=transferFunds&amount=14000&
destinationAccount=attackersAcct#" width="0"
height="0" />
```

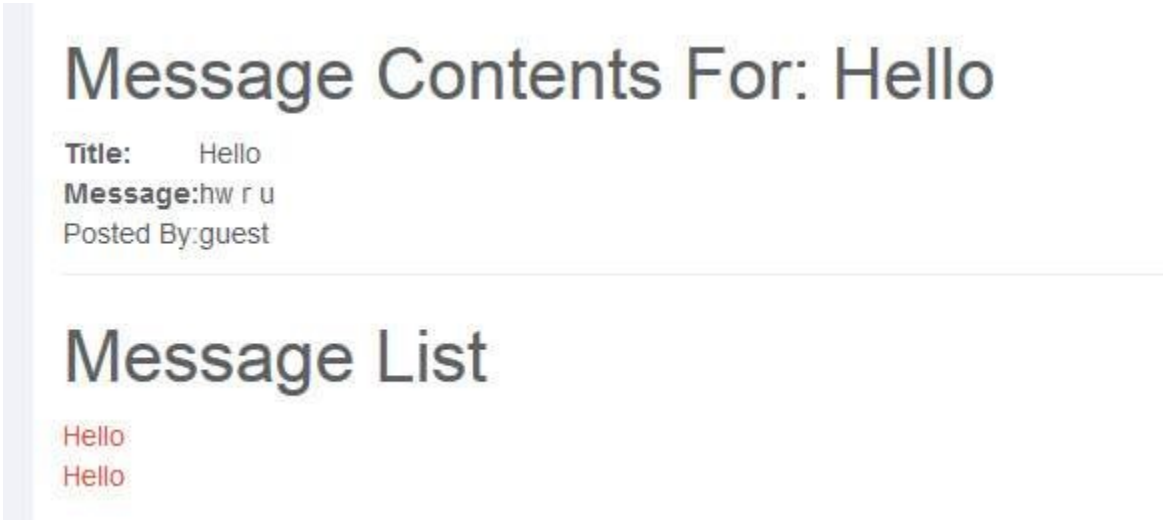
Hands ON

1 . Let us perform a CSRF forgery by embedding a javascript into an image. The snapshot of the problem is listed below.

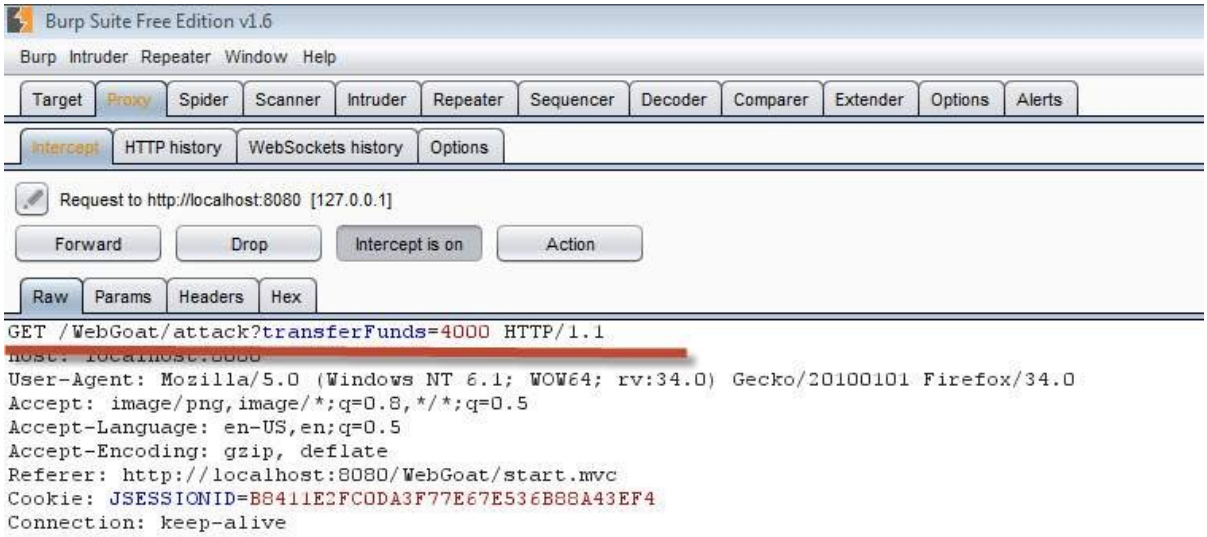
2 . Now we need to mock up the transfer into a 1x1 image and make the victim to click on the same.



3 . Upon submitting the message, the message is displayed as highlighted below.



3 . Now if the victim clicks the below URL, the transfer would be executed which can be found be intercepting the user action using burp suite. We are able to see the transfer by spotting it in Get message as shown below.



4 . Now upon clicking refresh the lesson completion mark would be shown.

Preventing Mechanisms

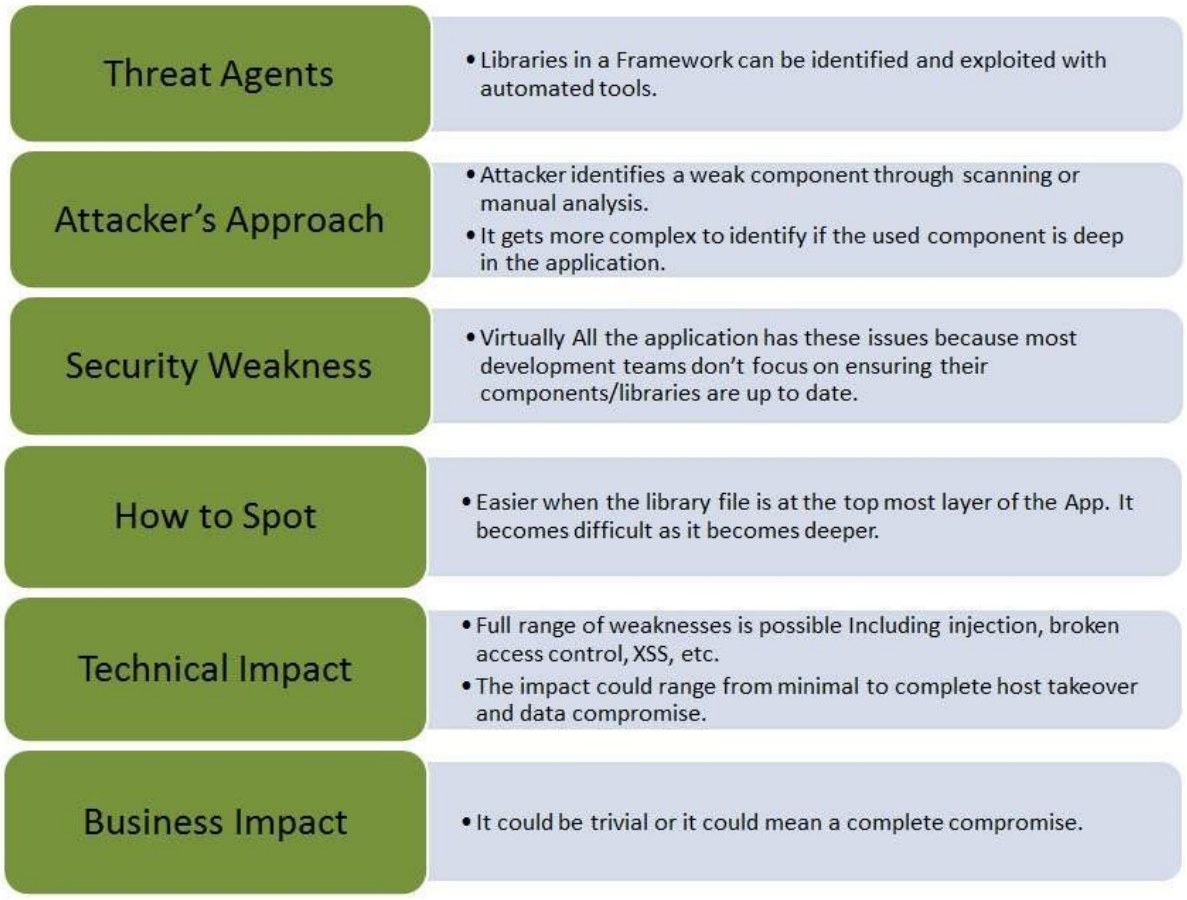
CSRF can be avoided by creating an unique token in a hidden field which would be sent in the body of the HTTP request rather than in an URL, which is more prone to exposure.

Forcing the user to reauthenticate or proving that they are a user in order to protect CSRF (e.g - CAPTCHA).

Using Components with Known Vulnerabilities

This kind of threat occurs when the Components such as libraries, frameworks used within the app almost always executes with full privileges. If a vulnerable component is exploited it makes hackers job easier to cause a serious data loss or server takeover.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



Example

Below are the examples of Using Components with Known Vulnerabilities :

- Attackers could invoke any web service with full permission By failing to provide an identity token.
- Remote-code execution with Expression Language injection vulnerability was introduced through the Spring Framework for Java based apps.

Preventing Mechanisms

Identify all components and the versions that are being used in the webapps not just restricted to database/frameworks.

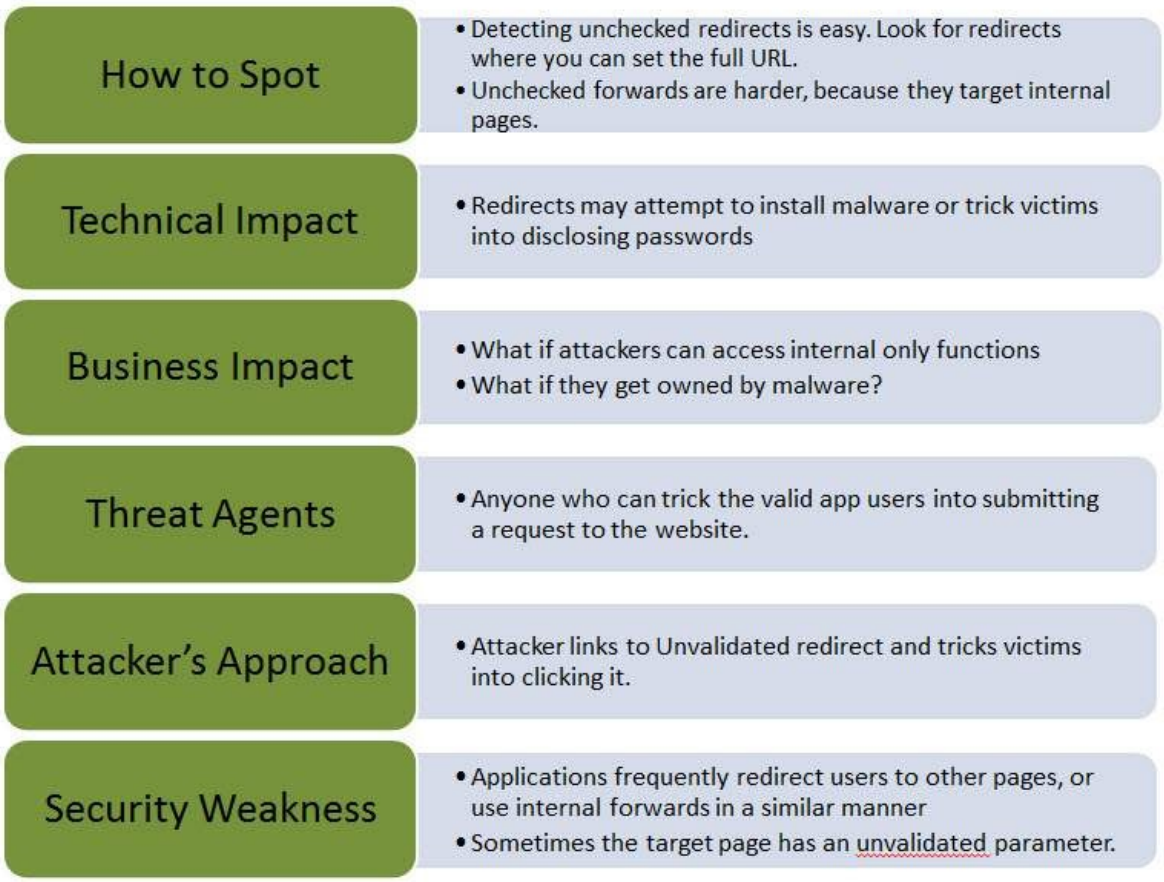
Keeping all the components such as public databases, project mailing lists upto date.

It is important to add security wrappers around components that are vulnerable in nature.

Unvalidated Redirects and Forwards

Most Web applications on net frequently redirect and forward users to other pages or other external websites, however, without validating the credibility of those pages, hackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Let us understand Threat Agents, Attack Vectors, Security Weakness, Technical Impact and Business Impacts of this flaw with the help of simple diagram.



Example

Below are some of the classic examples of Unvalidated Redirects and Forwards:

1. Let us say, the application has a page - redirect.jsp which takes a parameter redirectrul. The hacker add an malacious URL that redirects users which performs phishing/installs malware.

```
http://www.mywebapp.com
/redirect.jsp?redirectrul=hacker.com
```

2. All web application used to forward users to different parts of the site. Inorder to achieve the same, some pages use a parameter to indicate where the user should be redirected if an operation is successful. The attacker crafts an URL that will pass the

application's access control check and then forwards the attacker to administrative functionality for which the attacker has not got the access.

```
http://www.mywebapp.com
/checkstatus.jsp?fwd=appadmin.jsp
```

Preventing Mechanisms

It is better to avoid using redirects and forwards.

If unavoidable then it should be done without involving user parameters in redirecting the destination.

AJAX Security

Asynchronous Javascript and XML (AJAX) is one of the latest techniques used to develope web application inorder to give a rich user experience. Since it is a new technology there are many security issues that are yet to be completed established and below are the few security issues in AJAX.

- The attack surface is more as there are more inputs to be secured.
- It also Exposes the internal functions of the applications.
- Failure to protect authentication information and sessions.
- A very narrow line between client-side and server-side hence there are possibilities of committing security mistakes.

Example

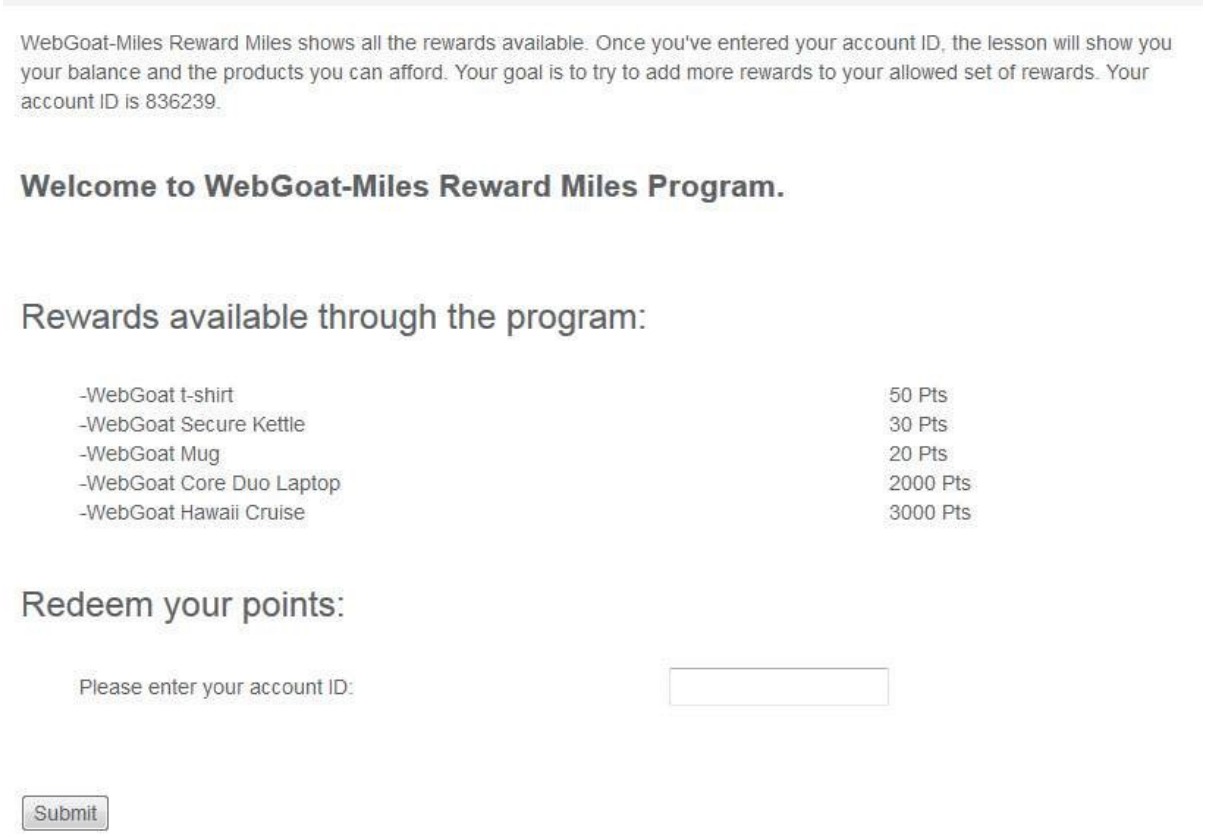
Below is an examples for AJAX Security:

In 2006, a worm infected yahoo mail service using XSS and AJAX

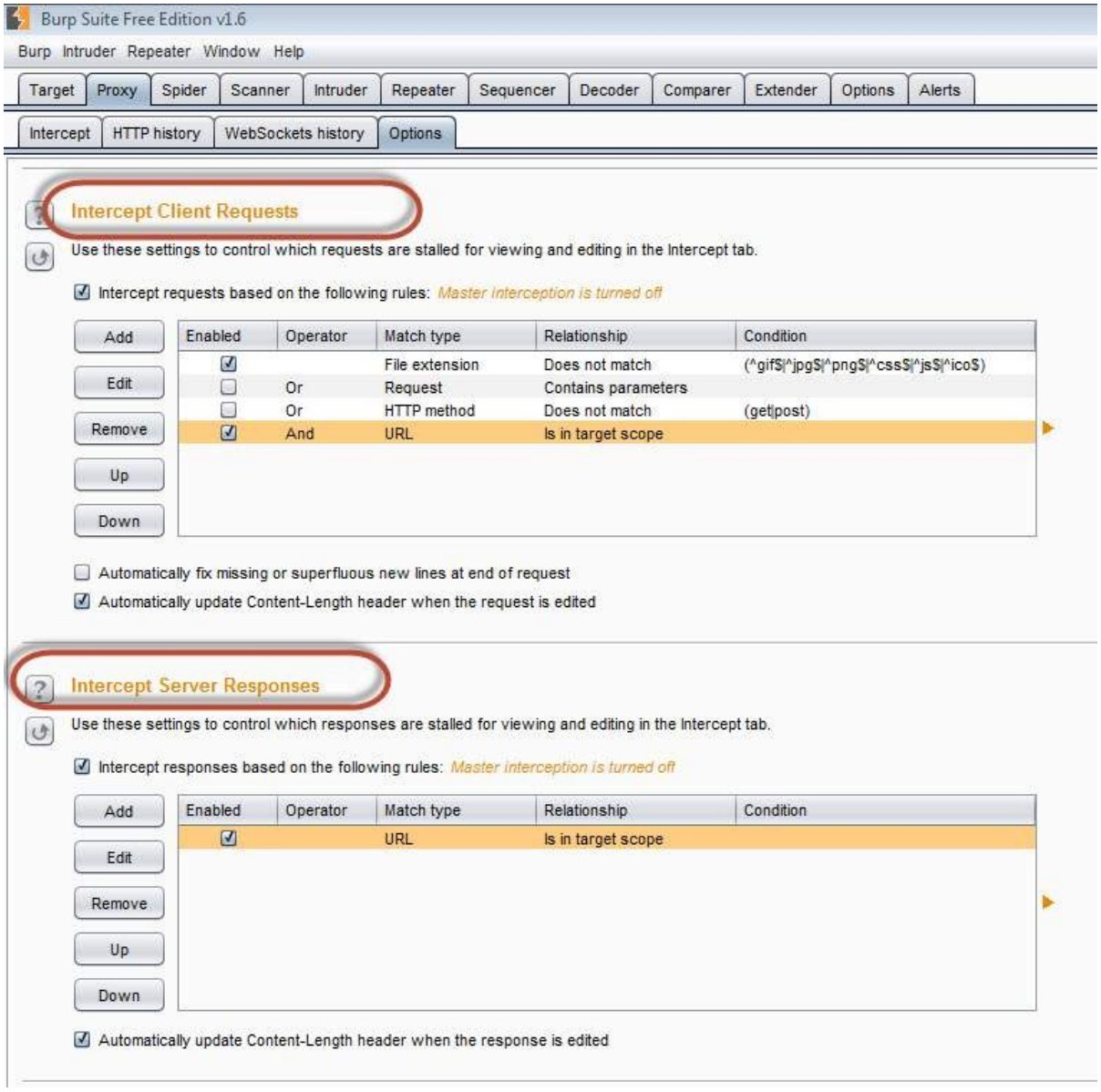
that took advantage of a vulnerability in Yahoo Mail's onload event handling. When an infected email was opened, the worm executed its JavaScript, sending a copy to all the Yahoo contacts of the infected user.

Hands ON

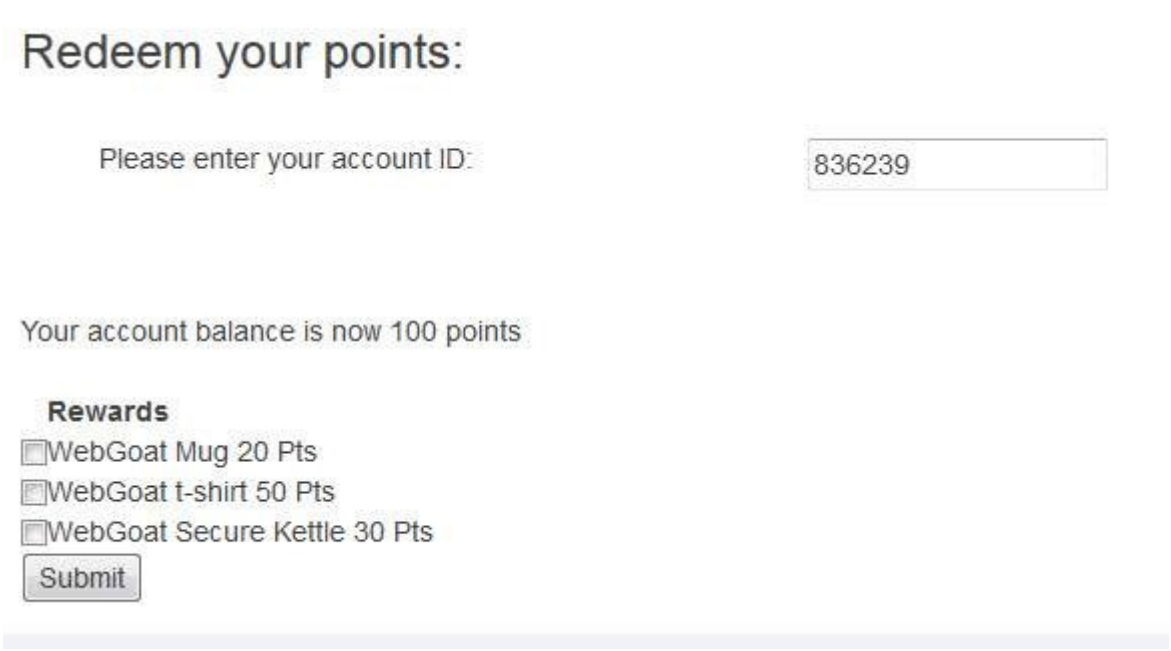
1 . We need to try to add more rewards to your allowed set of reward using XML injection. Below is the snapshot of the scenario.



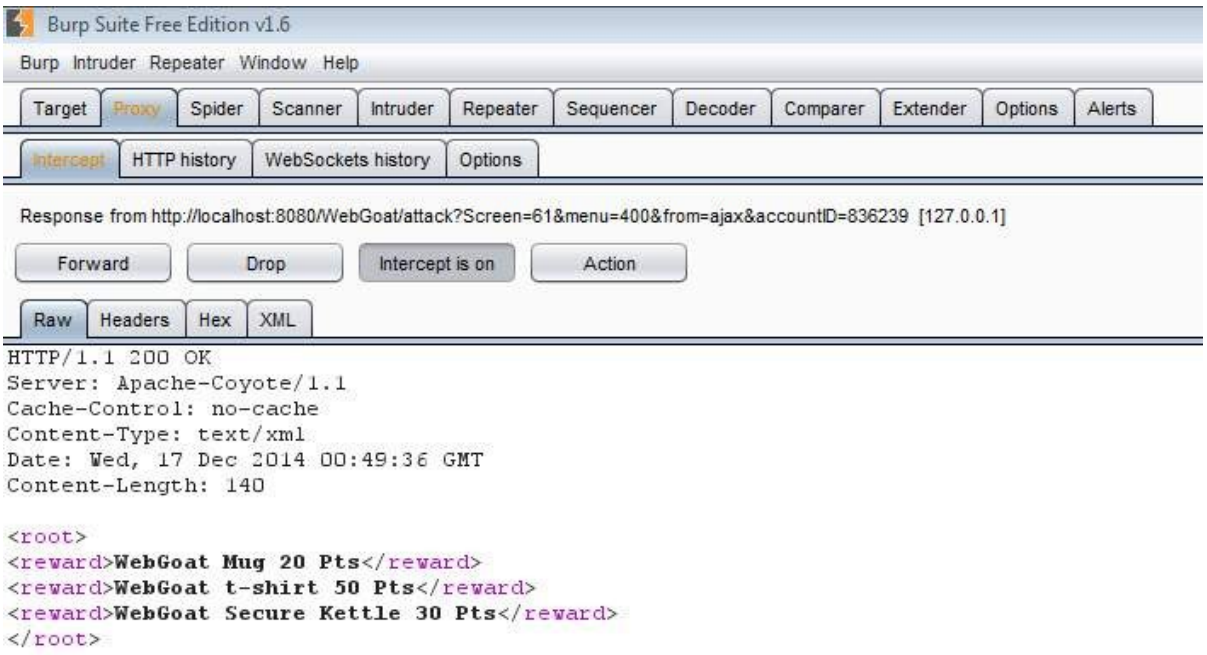
2 . Make sure that we intercept both request and response using Burp Suite. Settings of the same as shown below.



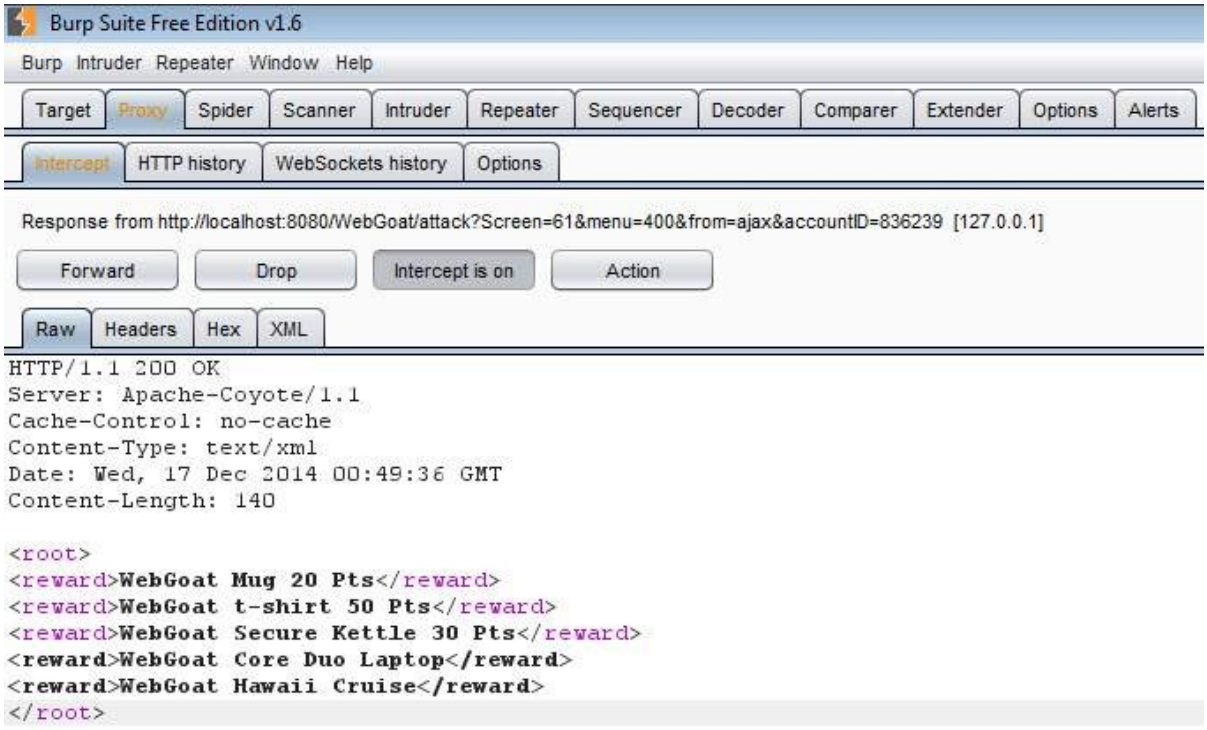
3 . Enter the account number as given in the scenario. We will be able to get a list of all rewards that we are eligible for. We are eligible for 3 rewards out of 5.



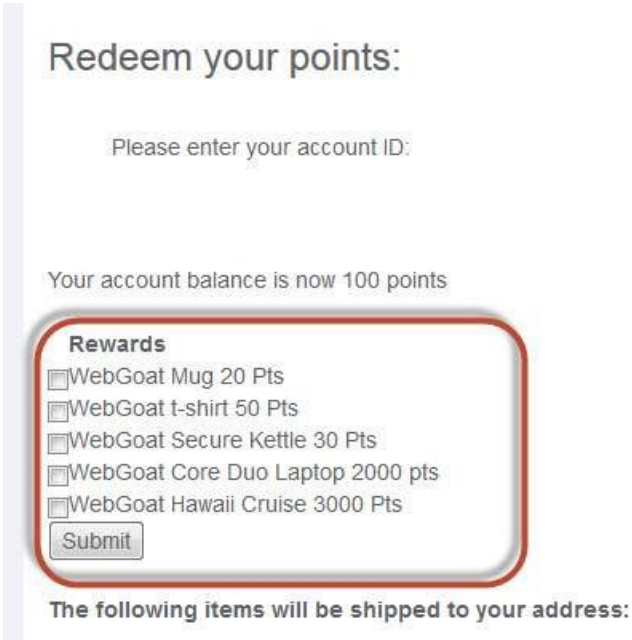
4 . Now let us click 'Submit' and see what we get in the response XML. As shown below the three rewards that are we are eligible are passed to us as XML.



5 . Now let us edit those XML's and add the other two rewards as well.



5 . Now all the rewards would be displayed to the user for them to select. Select the ones that we added and click 'SUBMIT'



6 . Message would appear that "* Congratulations. You have successfully completed this lesson."

Preventing Mechanisms

In Client side :

- Use .innerText instead of .innerHtml.
- Don't use eval.
- Don't rely on client logic for security.
- Avoid writing serialization code.
- Avoid building XML dynamically.
- Never transmit secrets to the client.
- Don't perform encryption in client side code.
- Don't perform security impacting logic on client side.

In Server side :

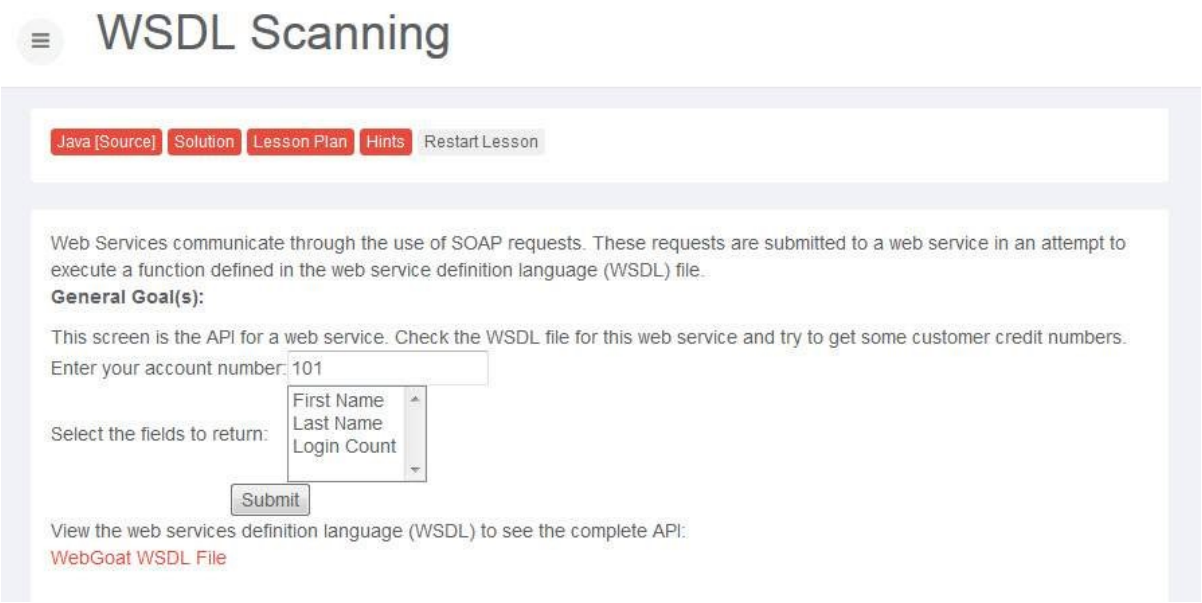
- Use CSRF protection.
- Avoid writing serialization code.
- Services can be called by users directly.
- Avoid building XML by hand, use the framework.
- Avoid building JSON by hand, use an existing framework.

Web Service Security

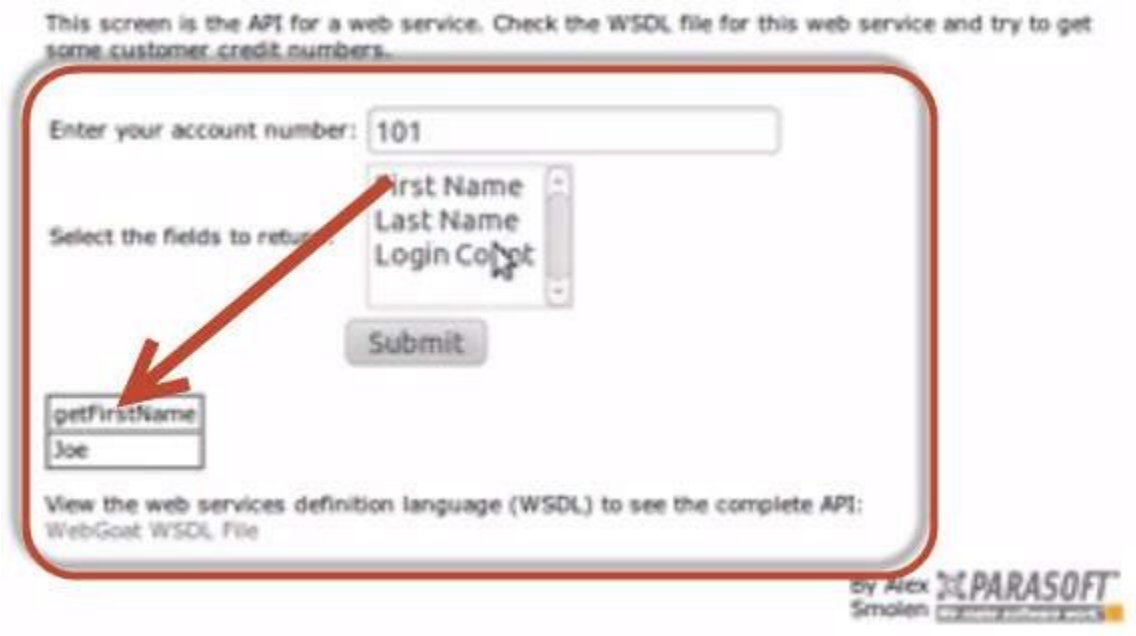
In mordern web based applications the usage of webservices is inevitable and they are prone for attacks as well. Since the web services request fetch from multiple websites developers have to take few additional measures inorder to avoid any kind of penetration by hackers.

Hands ON

1 . Navigate to web services area of Webgoat and goto WSDL Scanning. We need to now get credit card details of some other account number. Snapshot of the scenario is as mentioned below.



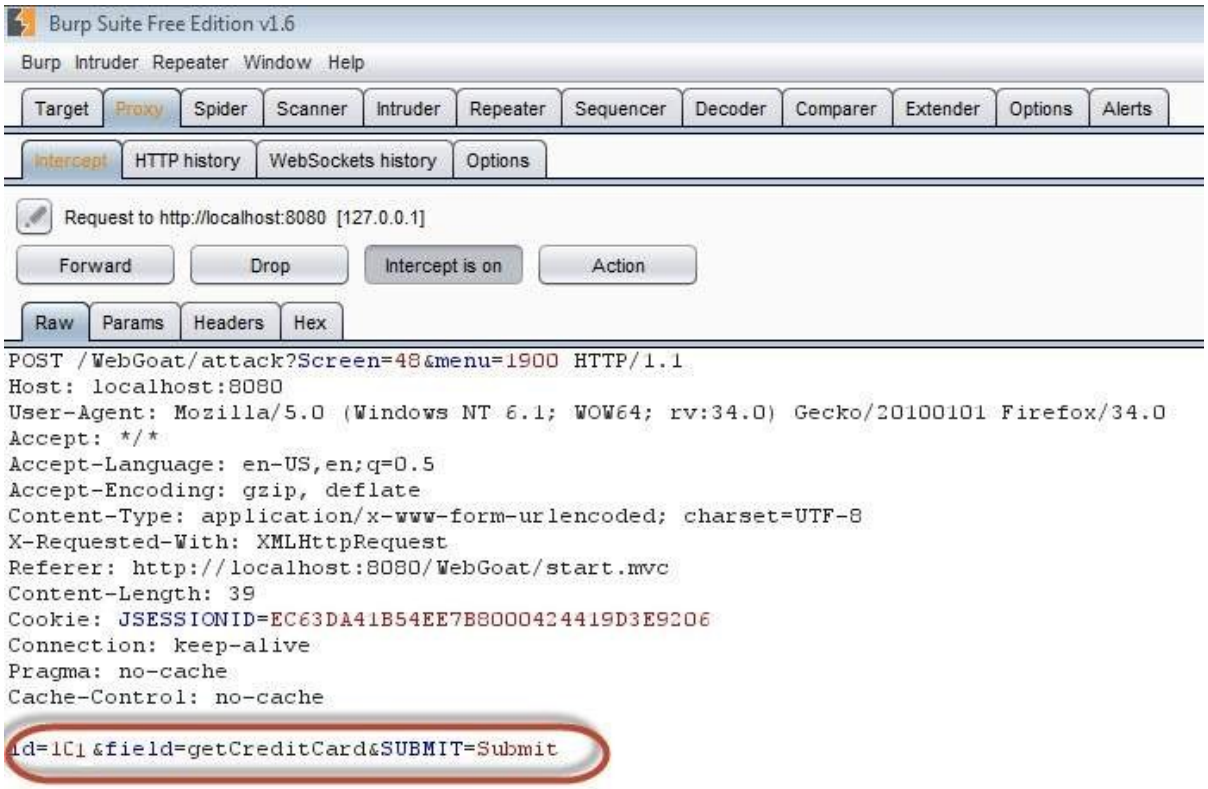
2 . If we select the first name, the 'getFirstName' function call is made through SOAP request xml.



3 . By opening the WSDL, we are able to see that there is a method to retrieve credit card information as well 'getCreditCard'. Now let us tamper the inputs using Burp suite as shown below

```
- <wsdl:portType name="WSDLScanning">
  - <wsdl:operation name="getCreditCard" parameterOrder="id">
      <wsdl:input message="impl:getCreditCardRequest" name="getCreditCardRequest"/>
      <wsdl:output message="impl:getCreditCardResponse" name="getCreditCardResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getLoginCount" parameterOrder="id">
      <wsdl:input message="impl:getLoginCountRequest" name="getLoginCountRequest"/>
      <wsdl:output message="impl:getLoginCountResponse" name="getLoginCountResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getLastName" parameterOrder="id">
      <wsdl:input message="impl:getLastNameRequest" name="getLastNameRequest"/>
      <wsdl:output message="impl:getLastNameResponse" name="getLastNameResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getFirstName" parameterOrder="id">
      <wsdl:input message="impl:getFirstNameRequest" name="getFirstNameRequest"/>
      <wsdl:output message="impl:getFirstNameResponse" name="getFirstNameResponse"/>
    </wsdl:operation>
```

4. Now let us tamper the inputs using Burp suite as shown below



5 . We are able to get the credit card information of other users.

Preventing Mechanisms

Since SOAP messages are XML-based, all passed credentials have to be converted to text format. Hence one has to be very careful in passing the sensitive information which has to be always encrypted.

Protecting message integrity by implementing the mechanisms like checksum applied to ensure packet's integrity.

protecting message confidentiality - Asymmetric encryption is applied to protect the symmetric session keys, which, in many implementations, are valid for one communication only and are subsequently discarded.

Buffer Overflows

A buffer overflow arises when a program tries to store more data in a temporary data storage area(buffer) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information can overflow into adjacent buffers hence corrupting the valid data held in them.

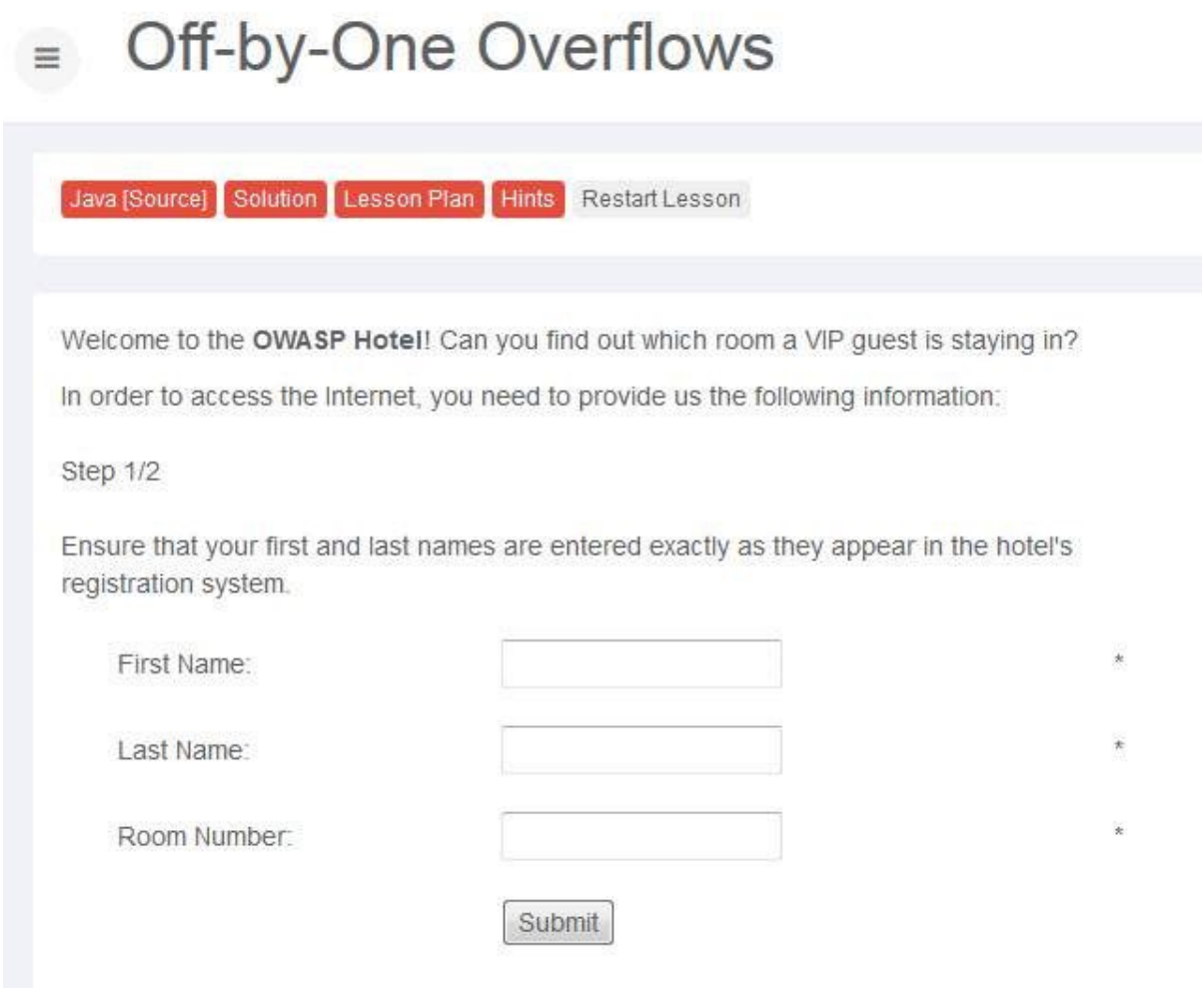
Example

Below is a classic examples of buffer overflow. It demonstrates a simple buffer overflow that is caused by the first scenario in which relies on external data to control its behavior. There is no way to limit the amount of data that user has entered and the behavior of the program depends on the how many characters the user has put inside.

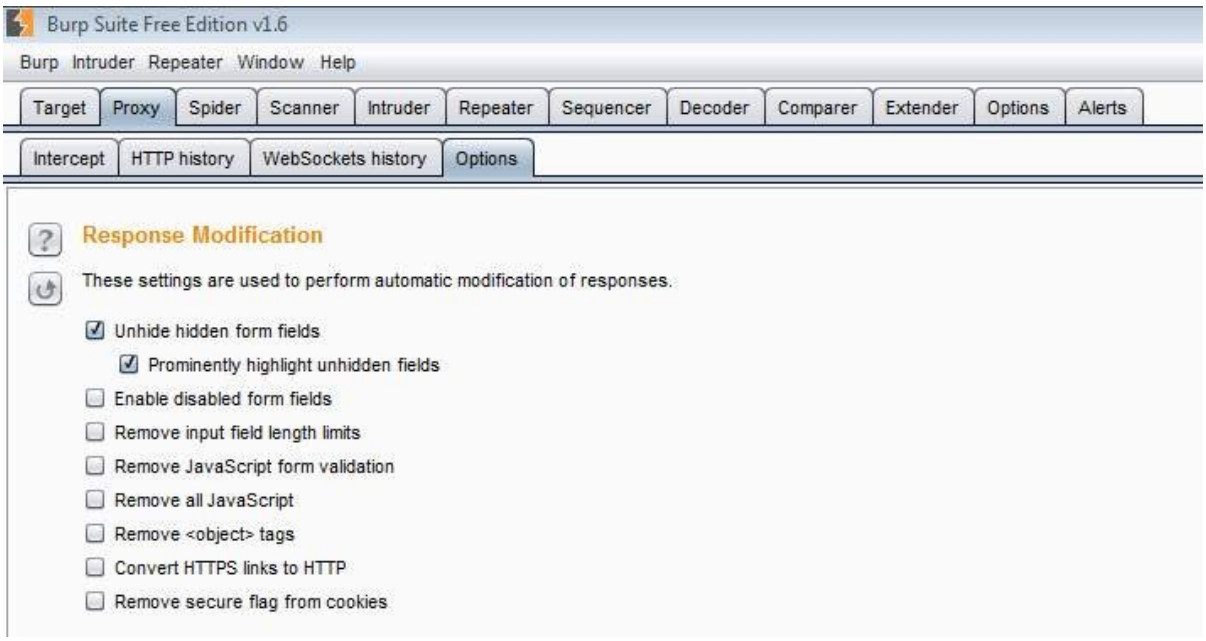
```
...
    char bufr[BUFSIZE];
    gets(bufr);
    ...
```

Hands ON

1 . We need to login with our name and room number to get the internet access. Below is the snapshot of the scenario.

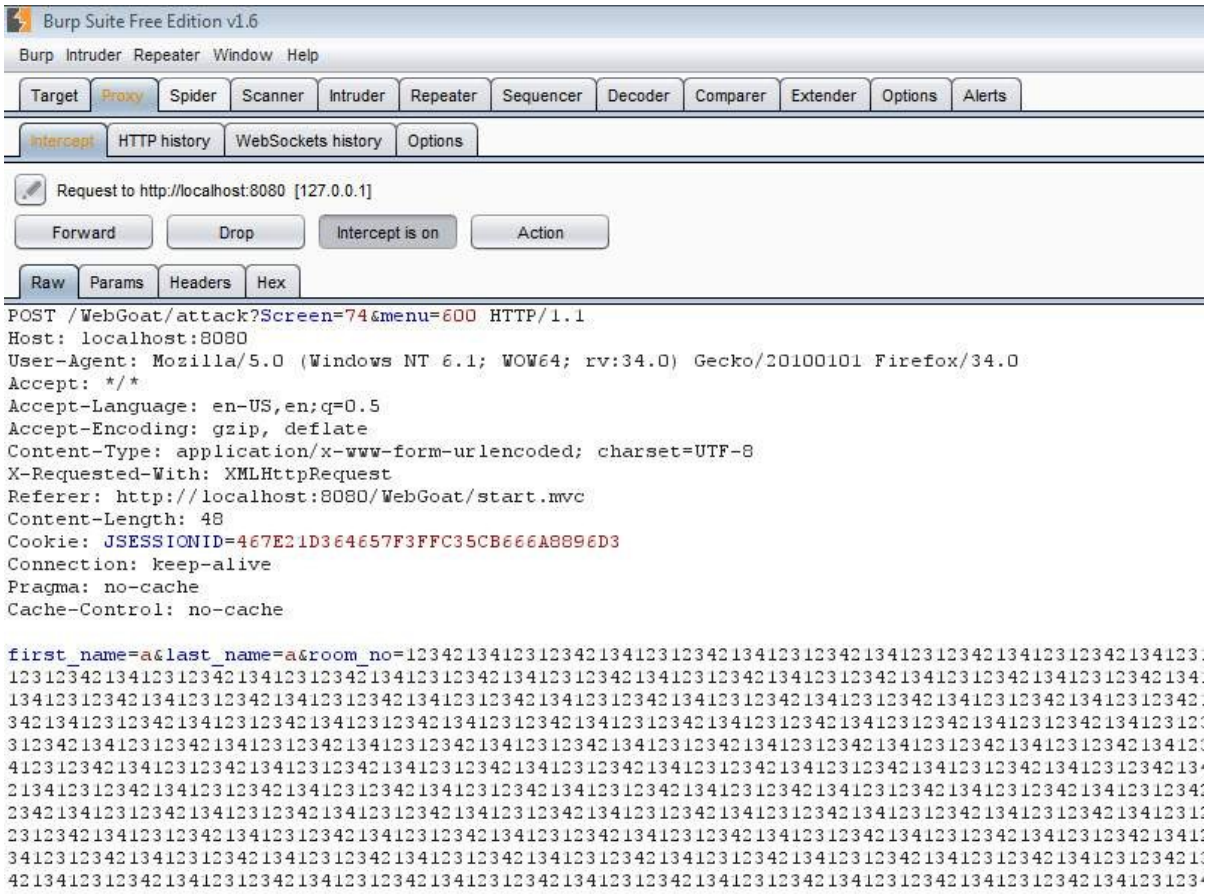


2 . Before we begin we will also enable "Unhide hidden form fields" in Burp Suite as shown below.

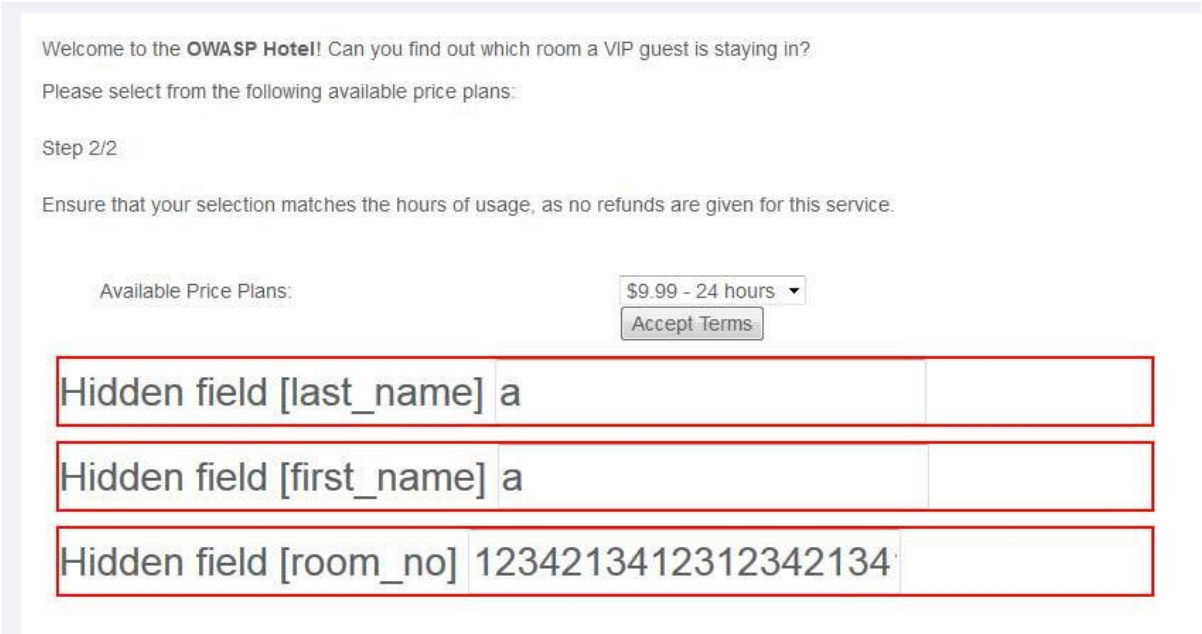


3 . Now let us send an input in name and room number field. We

also try and inject a pretty big number in the room number field.



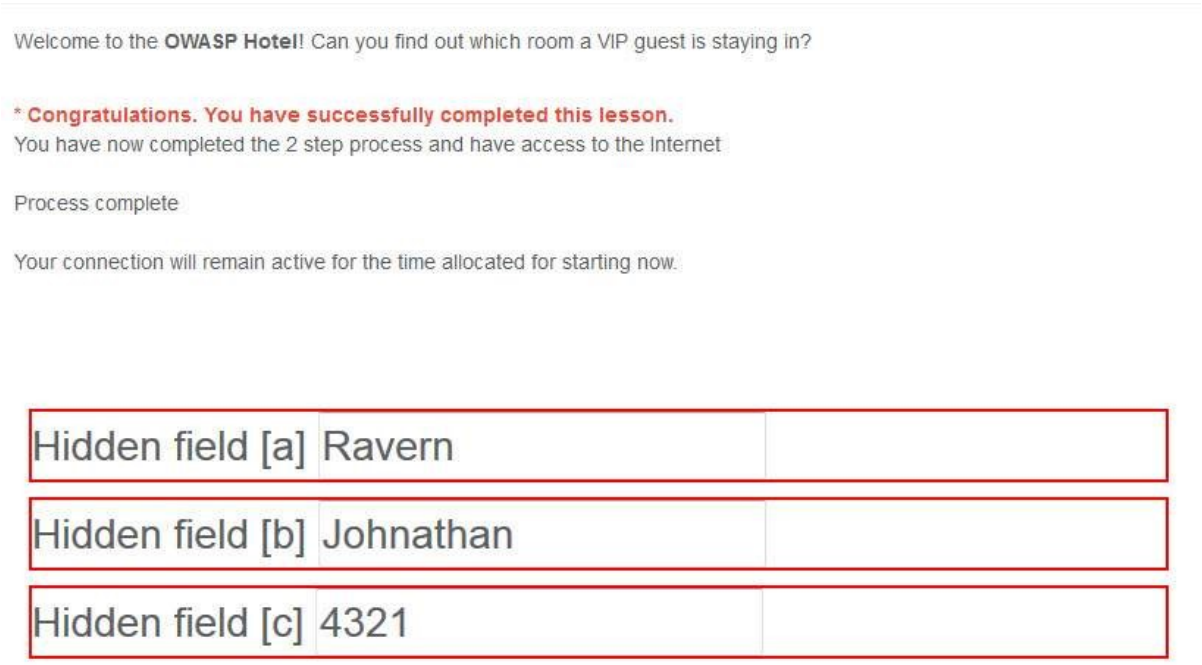
4 . The hidden fields are displayed as shown below and let us click accept terms.



5 . The attack is successful such that as a result of buffer overflow, it started reading the adjacent memory locations and displayed to the

user as shown below.

6 . Now let us login using the data displayed. After logging, the following message is displayed.



Preventing Mechanisms

- Code Reviewing.
- Developer training.
- Compiler tools
- Developing Safe functions
- Periodical Scanning

Denial of Service

Denial of Service(DOS) attack is an attempt by hackers to make a network resource unavailable. It is usually temporary or indefinitely interrupt the host which is connected to the internet. These attacks typically target services hosted on mission critical web servers such as banks, credit card payment gateways.

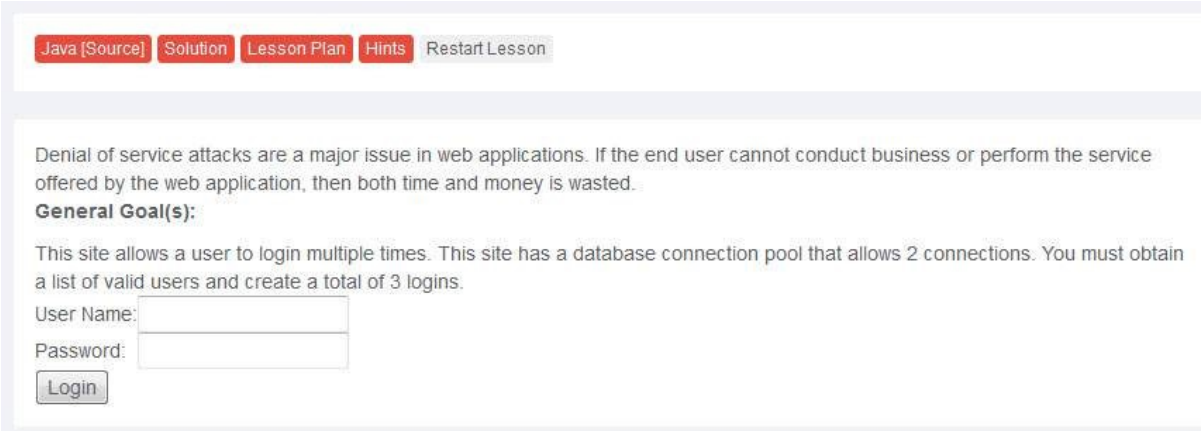
Symptoms of DOS

- Unusually slow network performance.
- Unavailability of a particular web site.
- Inability to access any web site.
- Dramatic increase in the number of spam emails received.
- Long term denial of access to the web or any internet services.
- Unavailability of a particular web site.

Hands ON

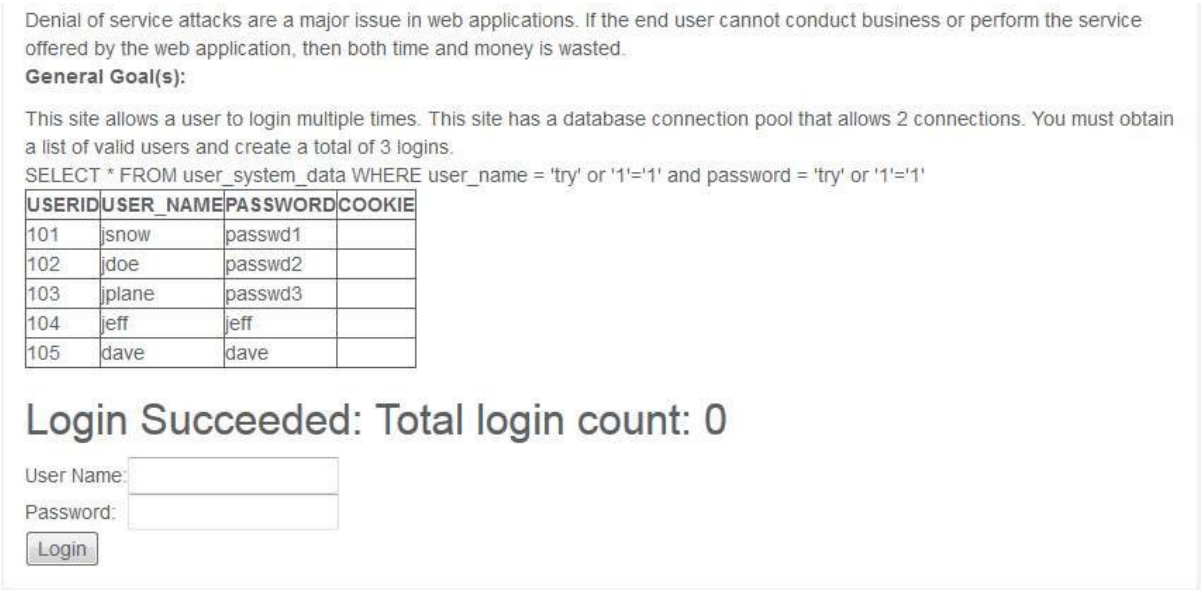
1 . Launch WebGoat and navigate to 'Denial of Service' section. The snapshot of the scenario is given below. We need to login multiple

times there by breaching maximum DB thread pool size.



2 . First we need to get the list of valid logins. We will use SQL Injection in this case.

3 . If the attempt is successful, then it displays all valid credentials to the user.



4 . Now login with each one of these user in atleast 3 different sessions inorder to make the DoS attack successful. As we know

that DB connection can handle only 2 threads, by using all logins it will create 3 threads which makes the attack successful.

Preventing Mechanisms

Perform thorough input validations. It is always better to expect worst case scenarios.

Avoid highly CPU consuming operations.

It is better to seperate Data disks from system disks.

Malacious File Execution

Developers often directly use or concatenate potentially vulnerable input with file or assume that input files are genuine. When the data is NOT checked properly, this can lead to the vulnerable content being processed or invoked by the web server.

Example

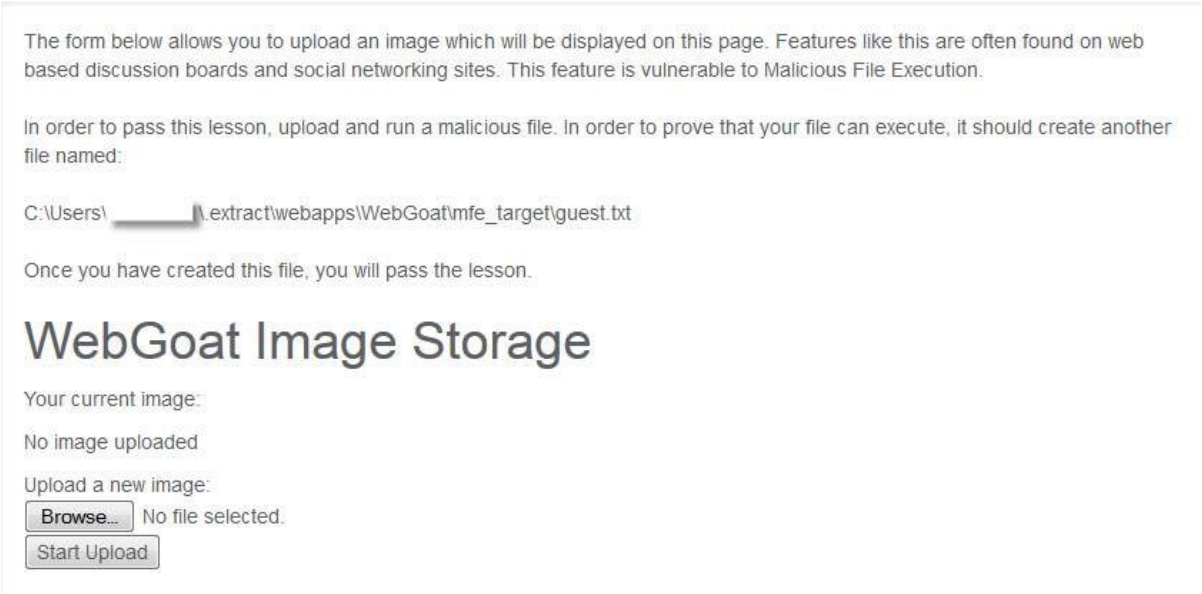
Below are some of the classic examples of :

- Upload .jsp file into web tree.
- Upload .gif to be resized.
- Upload huge files.

- Upload file containing tags.
- Upload .exe file into web tree.

Hands ON

1 . Launch WebGoat and navigate to Malacious file execution section. The snapshot of the scenario is given below.



2 . Inorder to complete this lesson we need to upload guest.txt in the above said location.

3 . Let us create a jsp file with the such that the guest.txt file is created on executing the jsp. The Naming of the jsp has no role to play in this context as we would be executing the content of the jsp file.

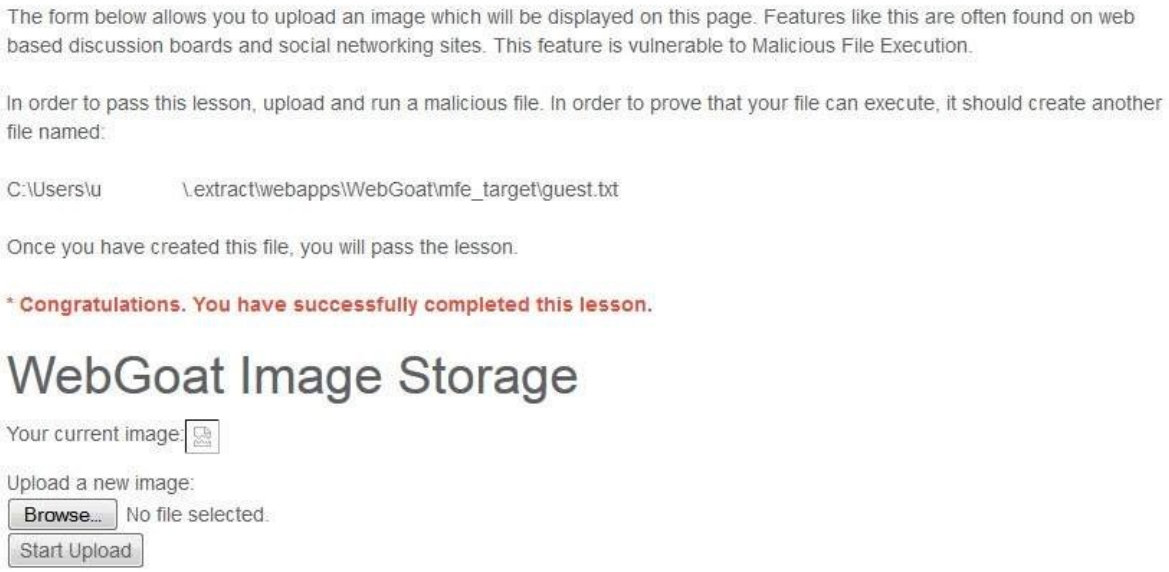
```
 <HTML> <% java.io.File file = new
java.io.File("C:\\Users\\username$\\.extract
\\webapps\\WebGoat\\mfe_target\\guest.txt");
file.createNewFile(); %> </HTML>
```

4 . Now upload the jsp file and copy the link location of the same after upload. The upload is expecting an image but we are

uploading a jsp.

5 . By navigating to the jsp file there wont be any message to the user.

6 . Now refresh the session where you have uploaded the jsp file and you will get the message that "you have successfully completed the lesson".



Preventing Mechanisms

Securing Sites using Web Site Permissions.

Adopting Countermeasures for Web Application Security

Understanding the Built-In User and Group Accounts in IIS 7.0

Automation Tools

There are various tools that are available to perform security testing of an application. There are few tools that can perform end to end security testing while some are dedicated to spot a particular type of flaw in the system.

Open Source tools

Below are the some of the open source testing tools which can be used for security testing purposes.

Specific Tool sets

Following are the tools that can help us to spot a particular type of vulnerabilities in the system.

Commercial Black Box Testing tools

Below are some of the commercial Black box testing tools which helps us to spot security issues in the application that we develop.

Free Source Code Analyzers

Commercial Source Code Analyzers
