

Unvalidated Parameters W/WebGoat

You will use the following Virtual Machine (VM) and application in this lab:

- Windows Server 2012 VM
- WebGoat application from OWASP
- Burp Suite Community Edition from PORTSWIGGER
- Web Browser (Firefox)

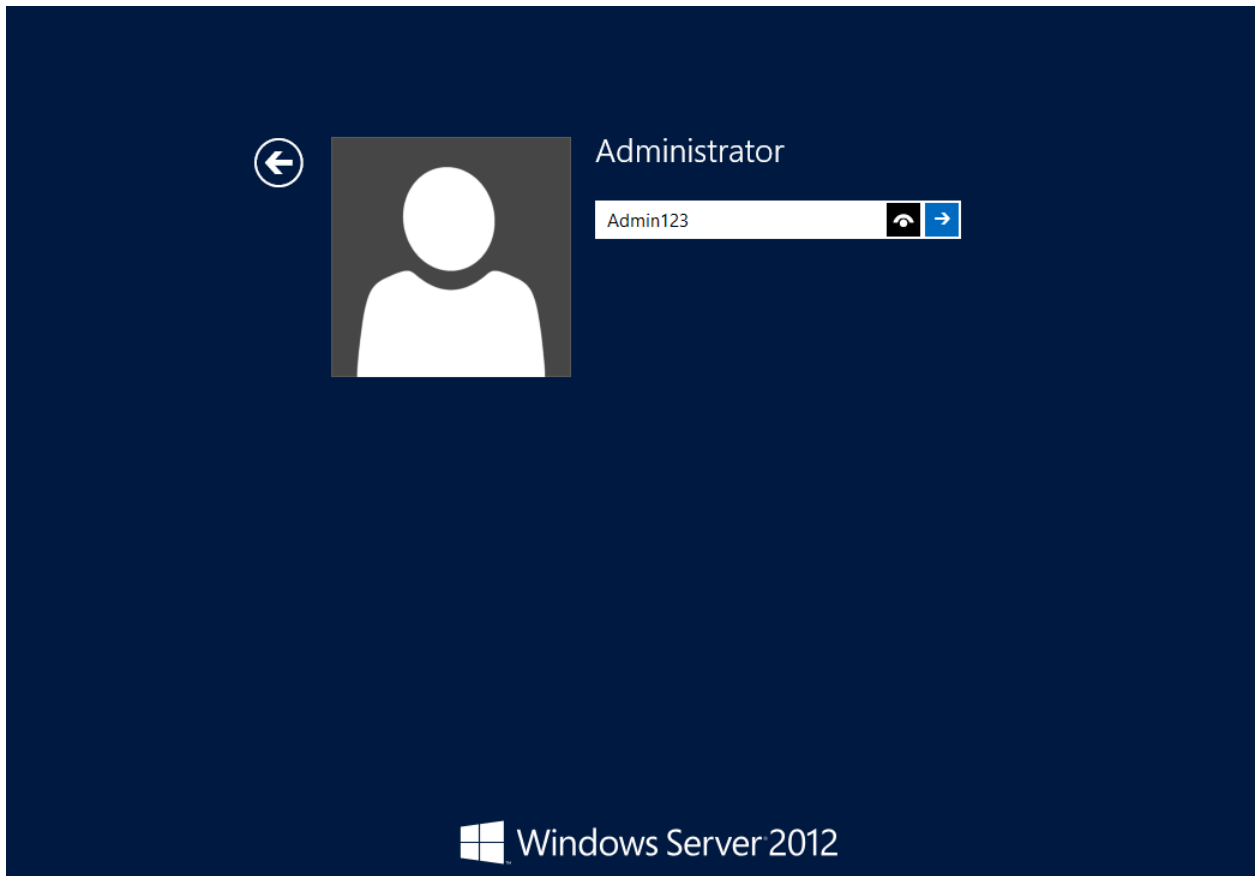
Before we proceed you will need to download and setup the lab environment on your Windows Server 2012 VM.

Step 1

Log in to your Administrator account on your windows server 2012 VM. Use your administrator account credential from your previous lab. If you followed setup according to previous lab setup the following login credential will work.

User: Administrator

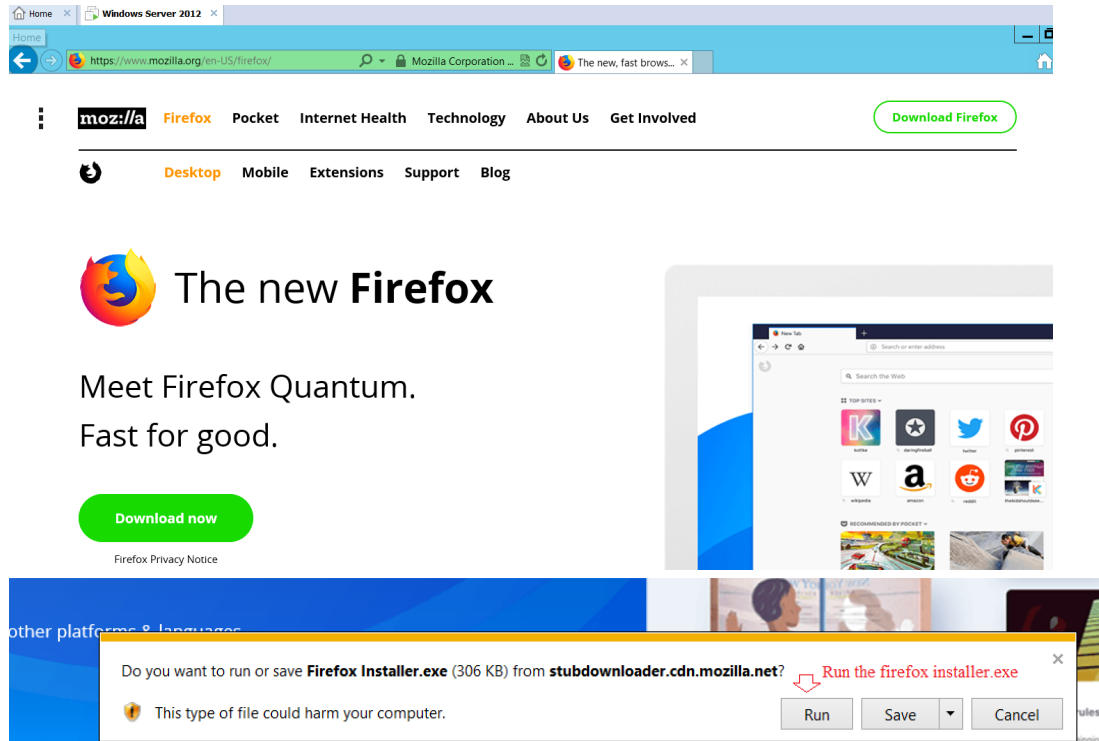
Password: Admin123



Minimize Windows Server manager dashboard. we will use it late to disable windows IIS web server. Open internet explorer and follow the steps to download and setup applications.

Download mozilla firefox web browser for convenience. Enter the following URL on your internet explorer and download the setup file and install firefox on your VM.

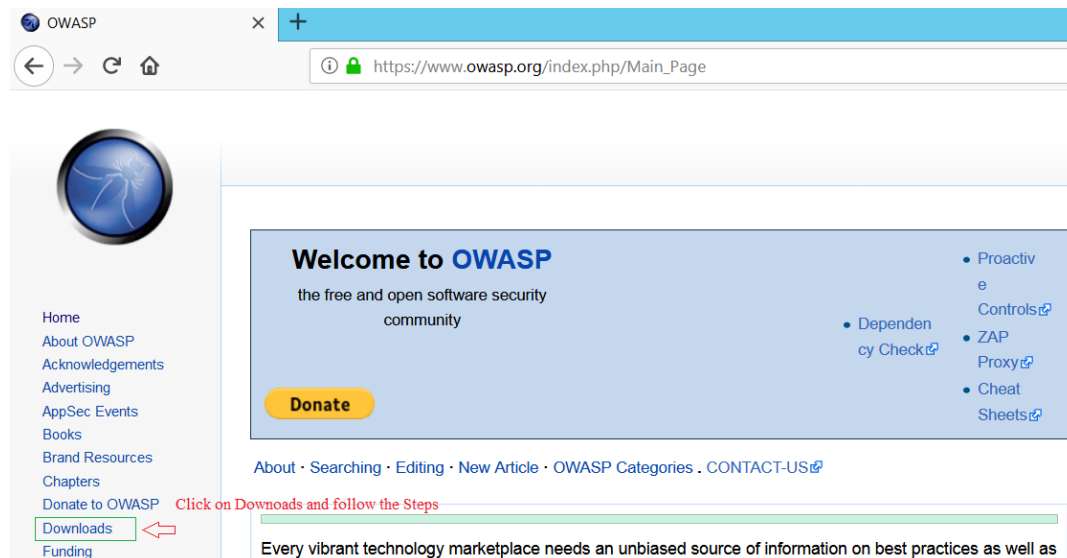
URL: <https://www.mozilla.org/firefox>



Open your firefox web browser and download few more applications to setup your lab environment.

Visit WebGoat official web site and download WebGoat-OWASP_Standard-5.2.

URL : <https://www.owasp.org/>



Category:OWASP Download

[Help](#)

The OWASP Download category should be used to mark any page that has a significant download available. The download should be clearly marked and described near the top of the page. Our old download center is located at [SourceForge](#). Many of our documents and tools are still available there.

[Click on the Link](#)

Redirect to sourceforge.net. Sourceforge is a centralized web-based service that offers software developer a online platform to control and manage free and open-source software project.

OWASP Source Code Center - E X

https://sourceforge.net/projects/owasp/files/

SOURCEFORGE Articles Cloud Storage Business VoIP Intern

Home

Name	Modified	Size	Downloads / Week
ASVS	2013-08-22		9
WebGoat	2008-07-12		122
Pantera	2008-05-29		176

Home / WebGoat

Name	Modified	Size	Downloads / Week
Parent folder			
WebGoat 5.2	2008-07-12		90
WebGoat 5.1	2008-01-19		8

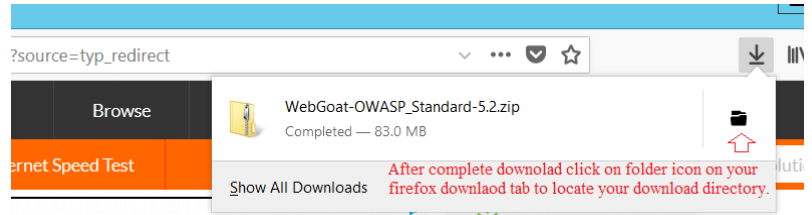
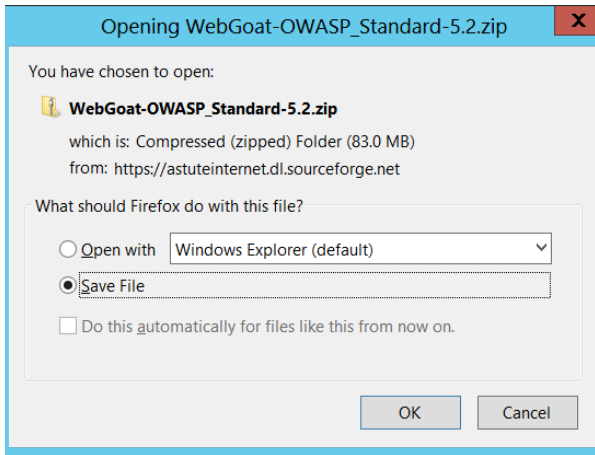
OWASP Source Code Center - E X

https://sourceforge.net/projects/owasp/files/WebGoat/WebGoat 5.2/

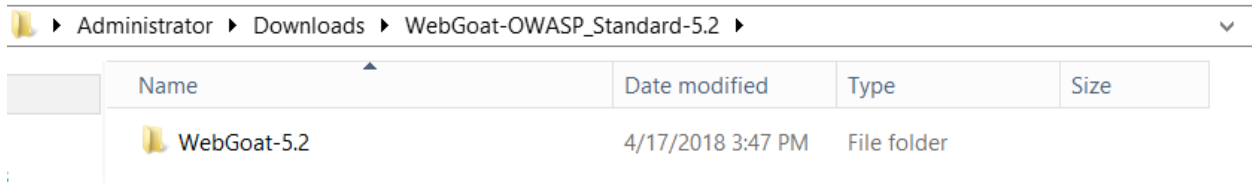
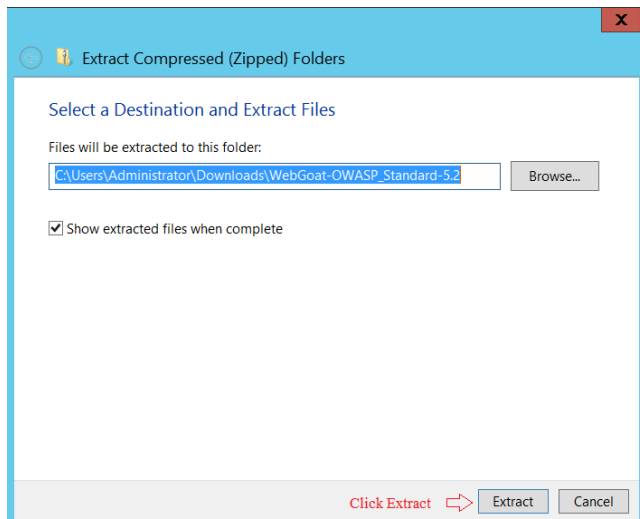
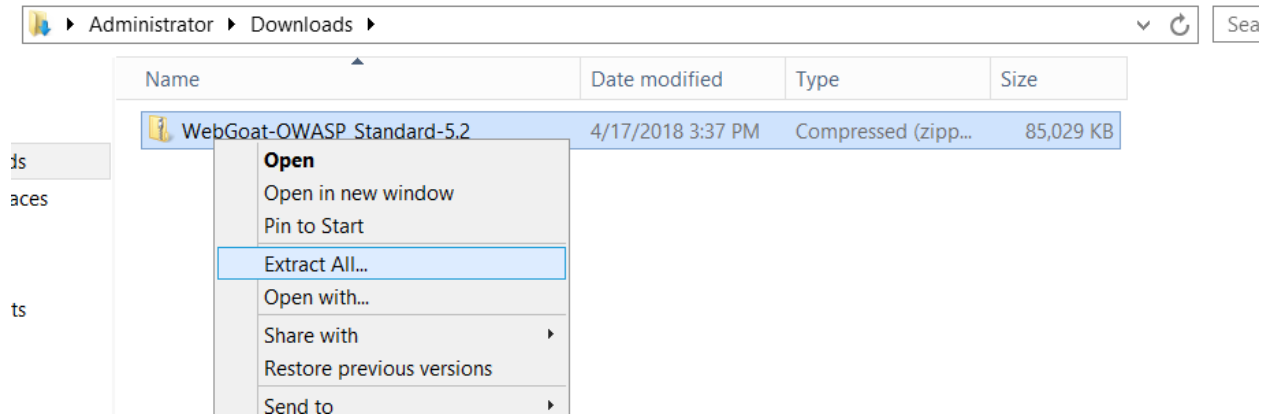
SOURCEFORGE Articles Cloud Storage Business VoIP Intern

Home / WebGoat / WebGoat 5.2

Name	Modified	Size	Downloads / Week
Parent folder			
WebGoat-OWASP_Standard-5.2.zip	2008-07-12	87.1 MB	61
WebGoat-5.2.war	2008-07-12	37.3 MB	9
WebGoat-OWASP_Developer-5.2.zip	2008-07-12	305.0 MB	18
readme-5.2.txt	2008-07-12	7.8 kB	2

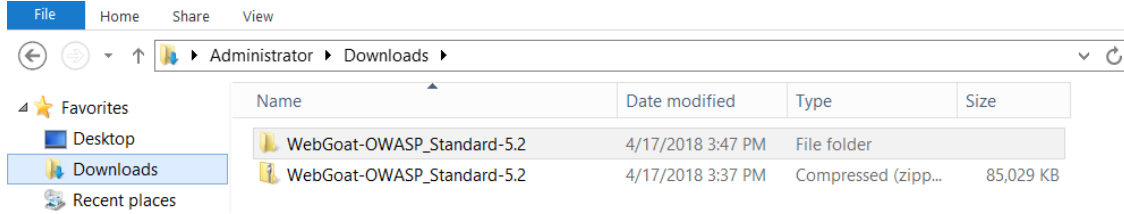


Right click WebGoat-OWASP_Standard-5.2.zip file and click Extract all.

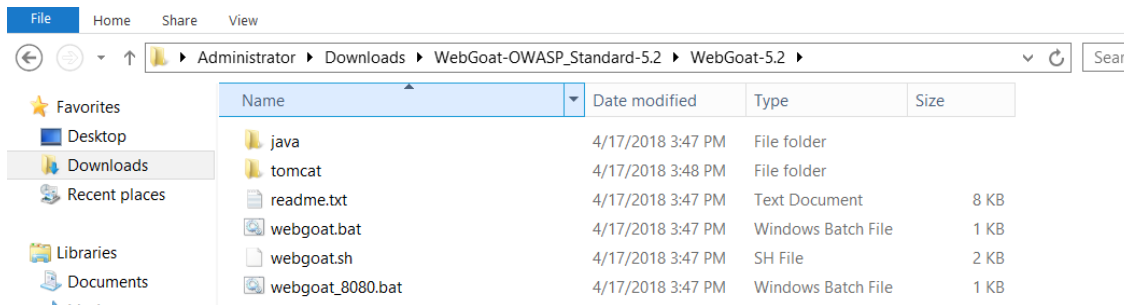


By default, the file is extracted inside the same download folder you can make change to your extract directory according to preference.

Extracted location: C:\Users\Administrator\Downloads\WebGoat-OWASP_Standard-5.2\WebGoat-5.2



The extracted folder contains tomcat webserver, java files and .bat windows executable file.



Step 2

We need to download java to run the WebGoat application. You can download java from java official website. Enter the following URL on your web browser and follow the steps to install java on your Windows Server 2012 VM

URL: <https://java.com/>

IMPORTANT INFORMATION REGARDING THE SECURITY OF JAVA SE

Download Help

JAVA + YOU,
DOWNLOAD
TODAY!

Free Java Download


Click on the Free Java Download Button and Run the executable Java setup file.


- Help Resources**
- » [What is Java?](#)
 - » [Remove Older Versions](#)
 - » [Disable Java](#)
 - » [Error Messages](#)
 - » [Troubleshoot Java](#)
 - » [Other Help](#)
- Offline Installation**
- Trouble downloading?
Try the [offline installer](#)

Download Java for Windows

Recommended Version 8 Update 171 (filesize: 1.79 MB)

Release date April 17, 2018

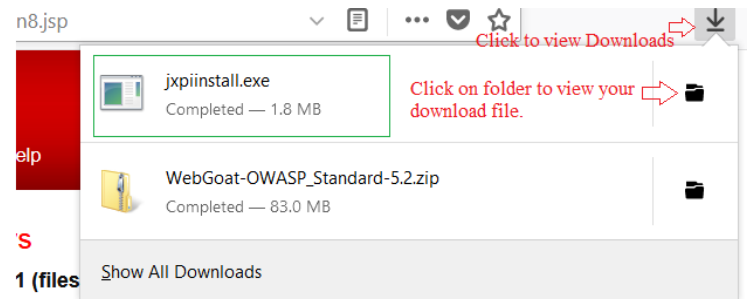
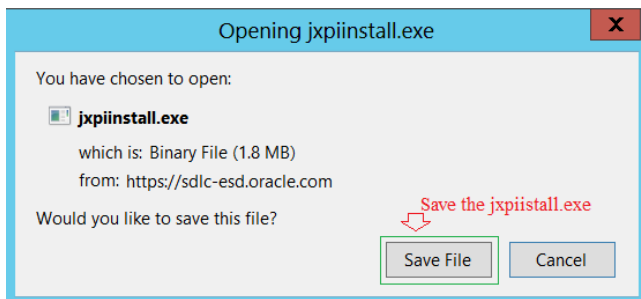
 Downloading and installing Java will only work in Desktop mode on Windows 8 and Windows 8.1. See the [Java on Windows 8 FAQ](#) for more detailed information.

 We have detected you are using Firefox which might be unable to use the Java plugin from this browser. Starting with Firefox Version 52 (scheduled for March 2017), Firefox has removed support for the standard way in which browsers support plugins. [More info](#)

Agree and download your java installation setup file.



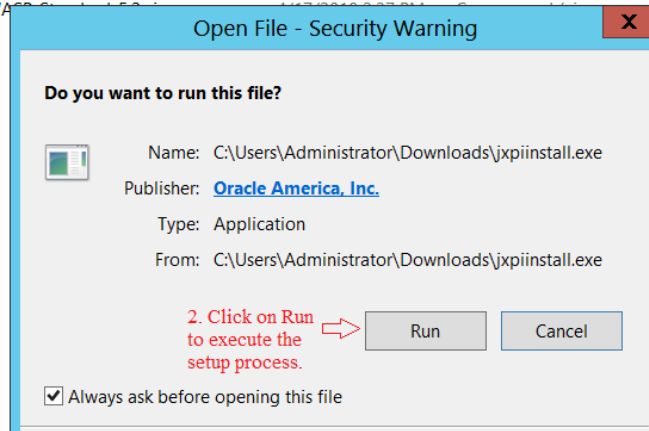
Agree and Start Free Download



Run the java setup file after complete download.

Name	Date modified	Type	Size
WebGoat-OWASP_Standard-5.2	4/17/2018 3:47 PM	File folder	
jxpiinstall.exe	4/17/2018 4:07 PM	Application	1,838 KB
WebGoat-OWASP_Standard-5.2	4/17/2018 3:47 PM	File folder	15,029 KB

1. Double Click on .exe file to begin the java setup



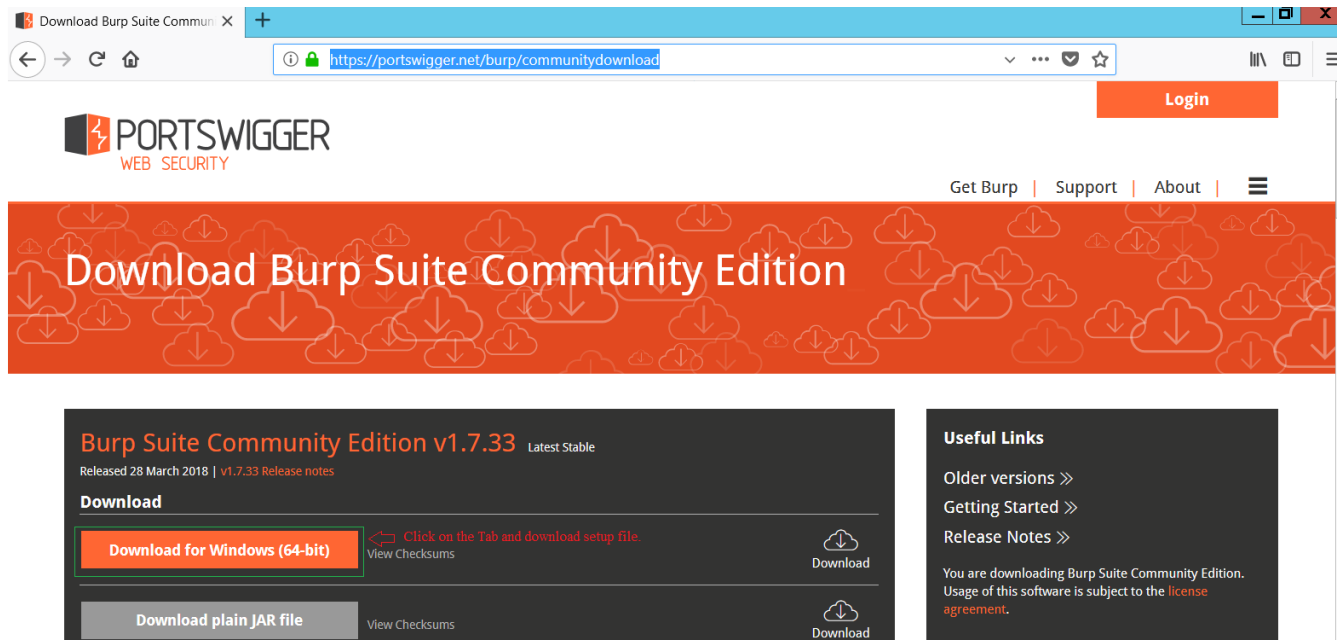
Click on Install and it will take a while download and complete the java setup on you Windows Server 2012 VM. Make sure you need to restart your Web Browser after successfully completing a java setup.

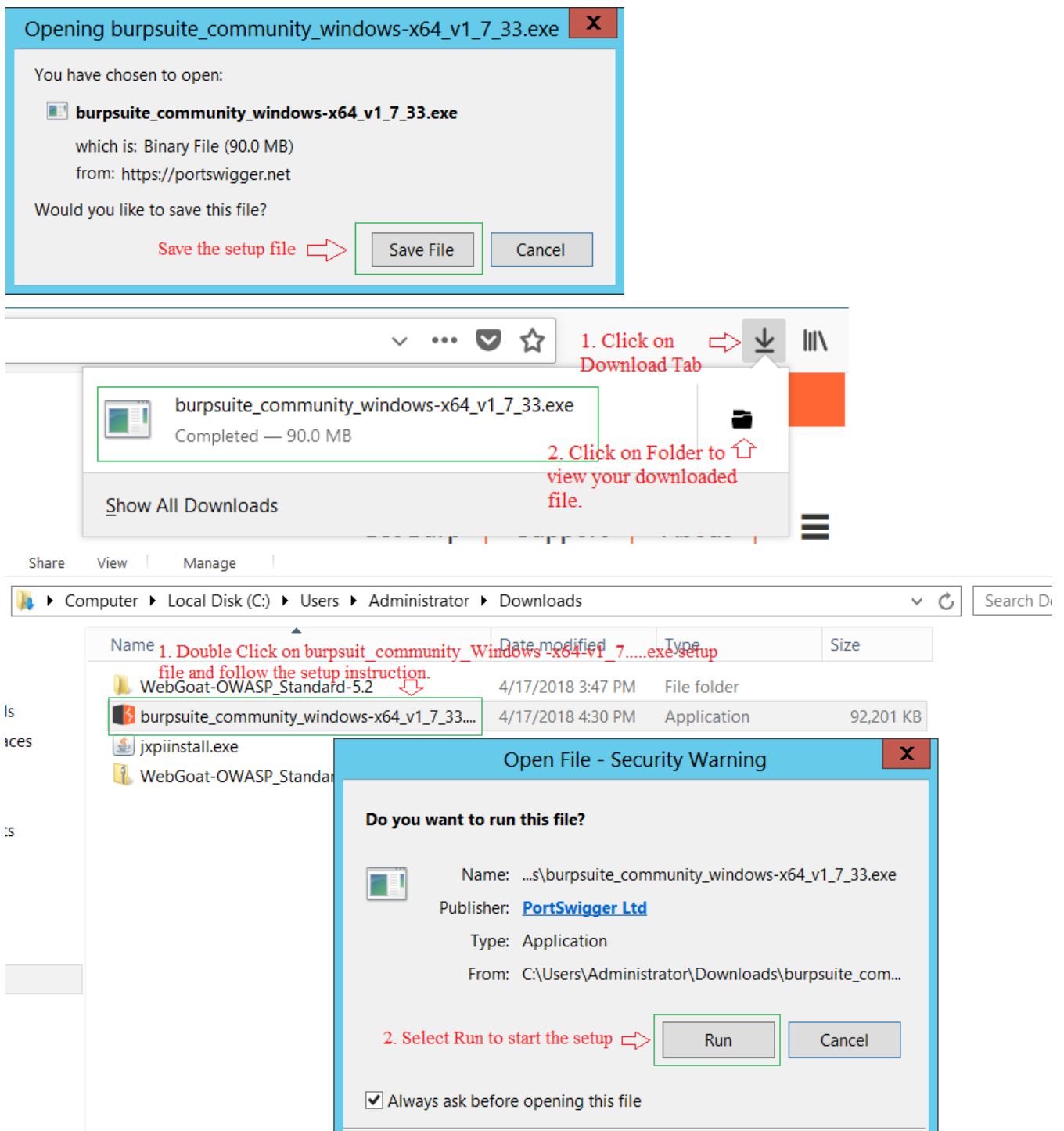


Step 3

We need to install Burp Suite on our windows server 2012 VM. Open your web browser and enter the following URL to download Burp Suite free community edition.

URL: <https://portswigger.net/burp/communitydownload>




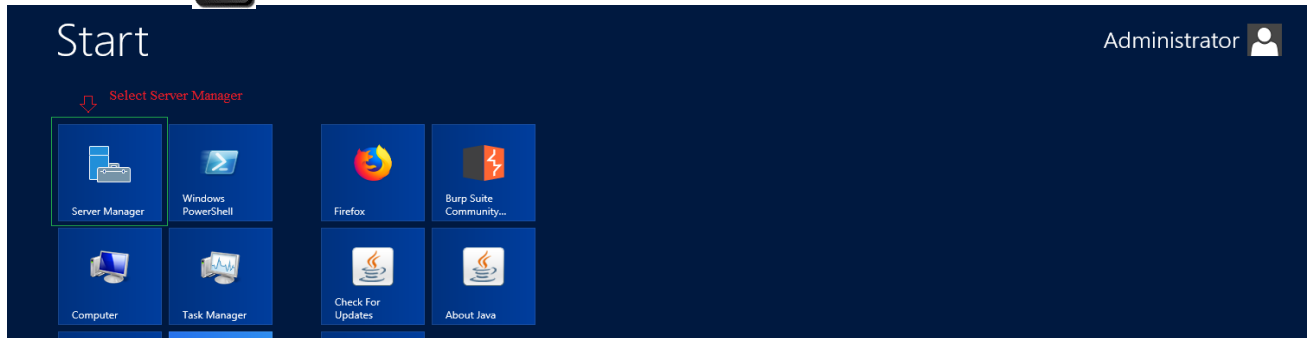


Follow the setup instruction and use the default setting don't make changes. After installation, you can access the application from your windows server startup screen or application shortcut from Desktop.

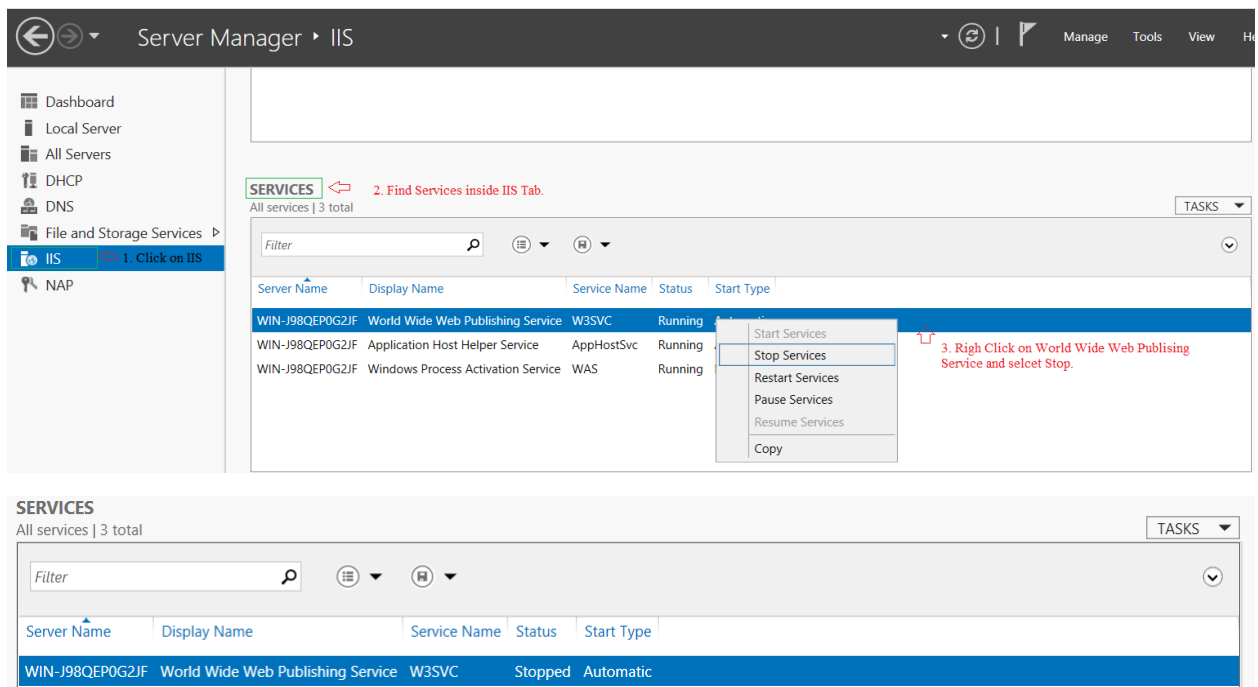
Exercise 1 Parameter Tampering

In this lab, we will use the WebGoat application from OWASP to do perform some attack on web application. It is good learning environment and allows us to exploit a number of different web application specific vulnerabilities.

Webgoat runs as a webserver. In order to start it up, you must first stop the IIS Webserver. To do so we can access our Server Manager Dashboard from our windows startup screen. Press Windows key  then select Server Manager from the start menu.

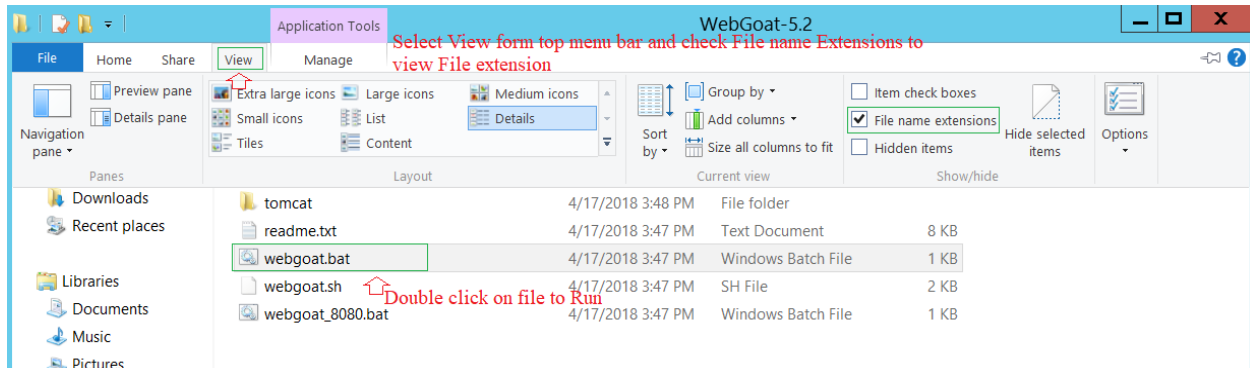


Select IIS service tab from you server manager dashboard on you left and scroll down to services list and find world wide web publishing service inside the list. You need to check the status of the service and if it is running you need to stop the service. To do so you need to select the world wide web publishing service, right click on it and select stop services. It might take few minutes wait until the status is updated with stopped sign.



Start up the Webgoat by opening the Webgoat-5.4 extracted folder on your Windows Server 2012 VM. In our case the Webgoat-5.4 is extracted inside our administrator download folder.

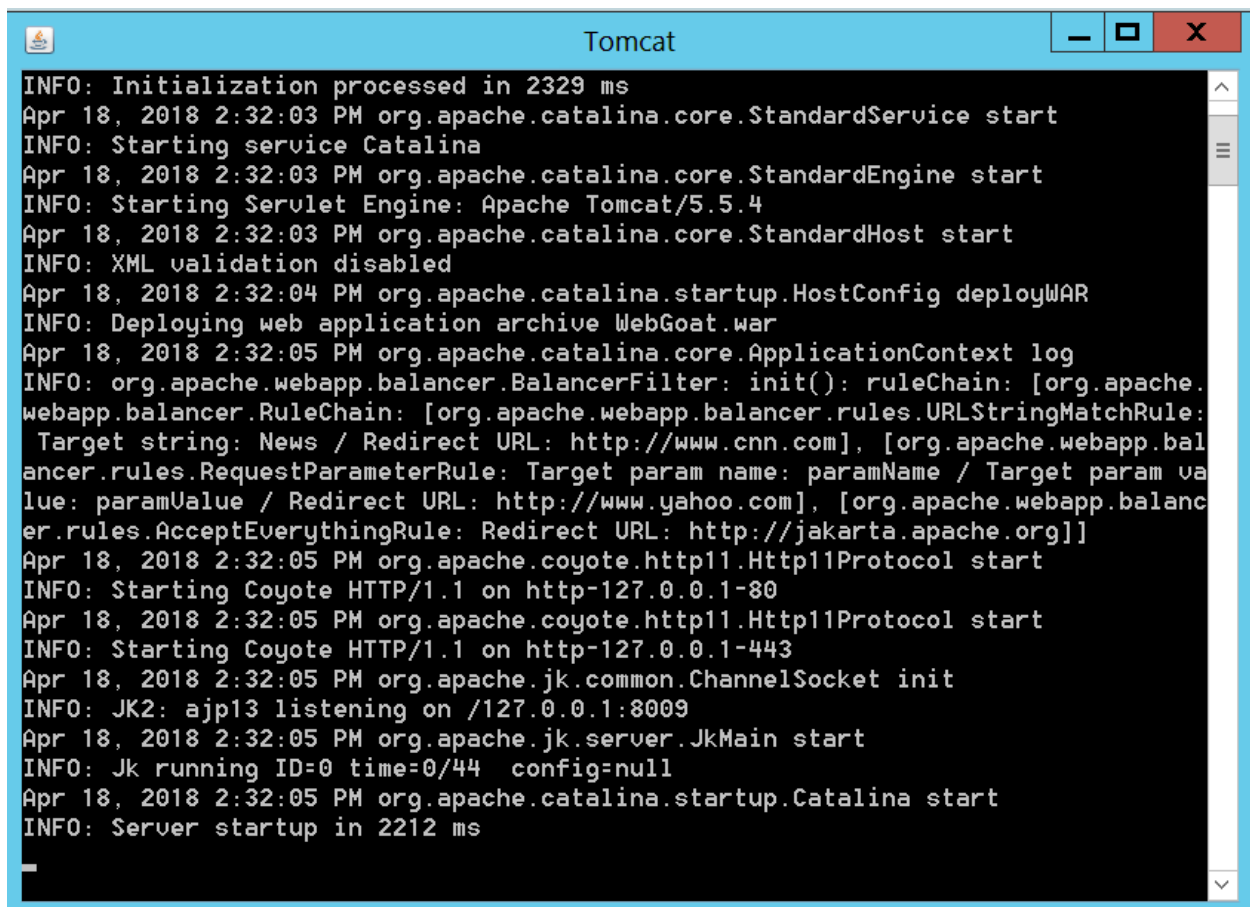
Navigate inside the extracted folder and double click on the WebGoat.bat (Don't use the WebGoat_8080.bat file). By default, a file extension is hidden we can enable from folder view menu. Click on the view menu inside the folder and check File Name Extensions on top right menu options.



You should see a status windows appear and Apache/Tomcat running:

Now, minimize this window.

Do not close this windows during the lab, as it will stop the Apache service as well as WebGoat!



After it starts, open firefox inside your Windows Server 2012 server, and navigate to.

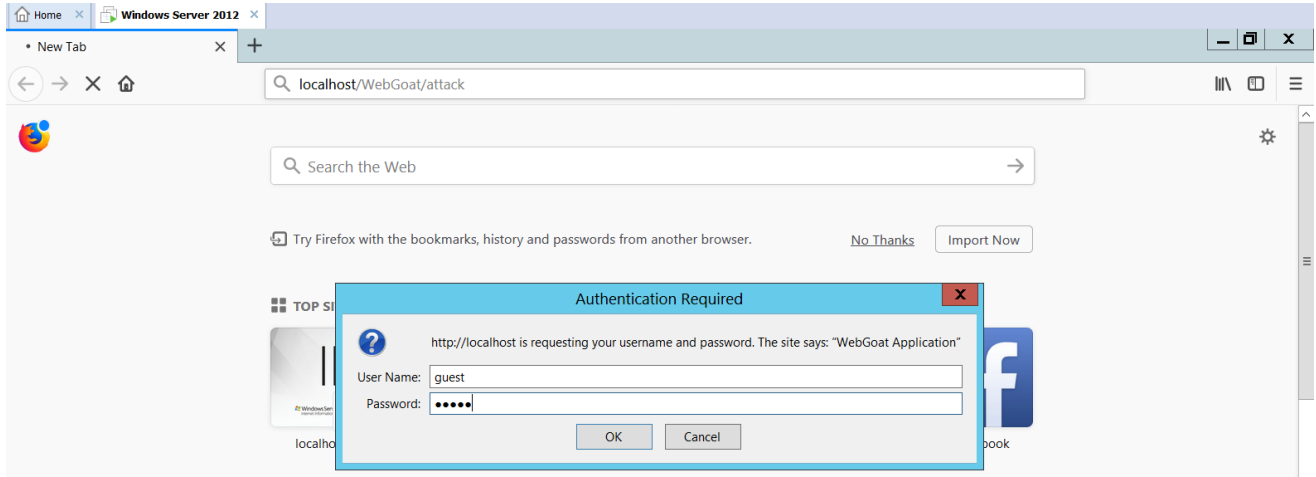
<http://localhost/WebGoat/attack>

Make sure the W and G on webgoat is capital.

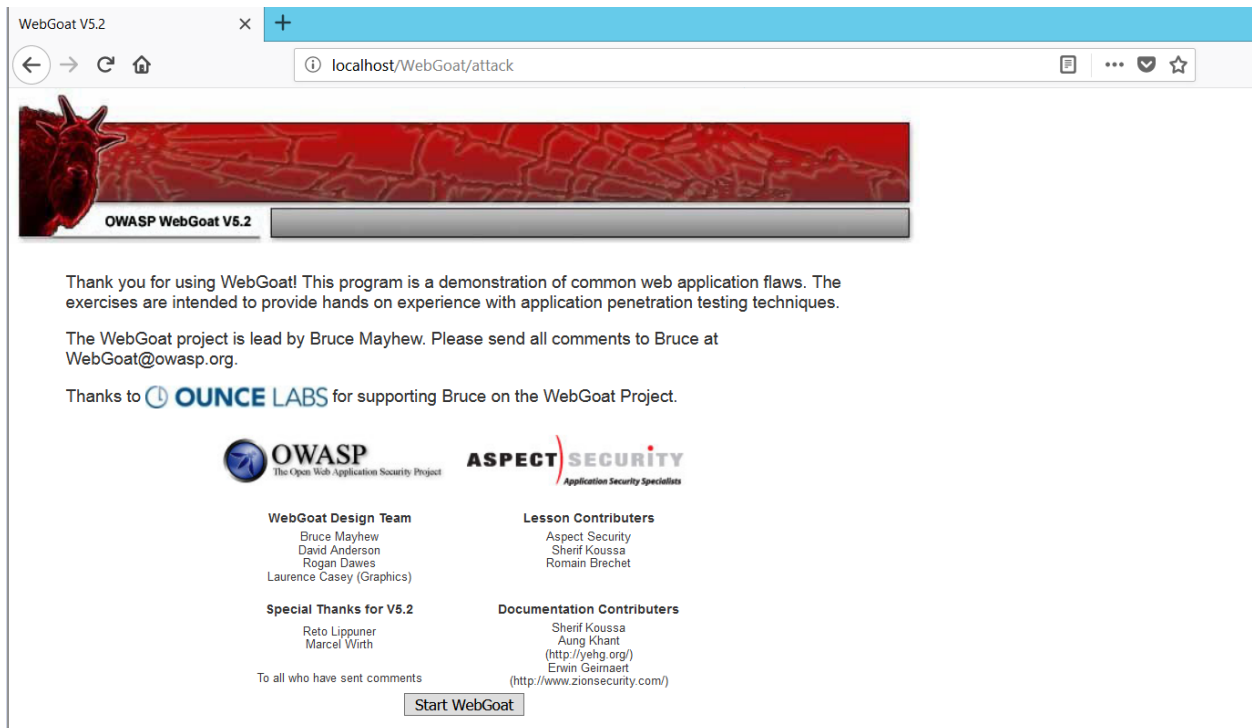
You will next be prompted with a login. Log in with account:

Username: guest

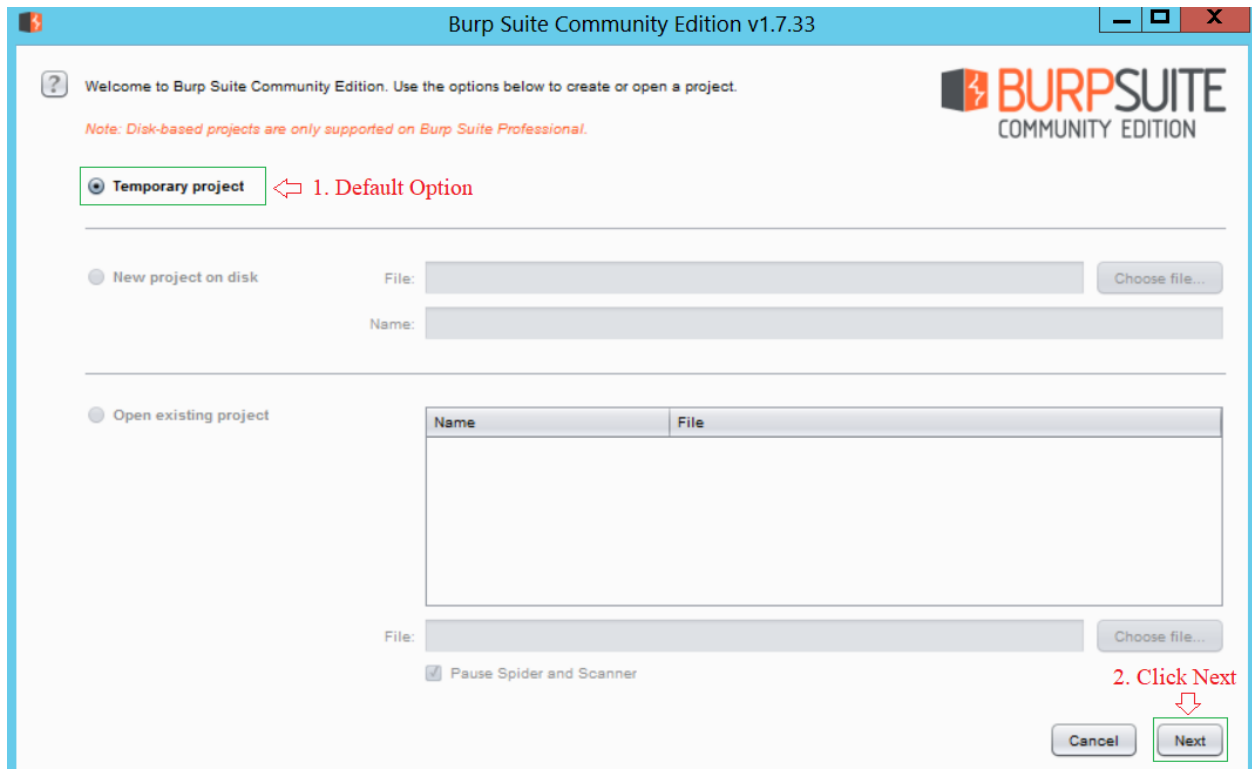
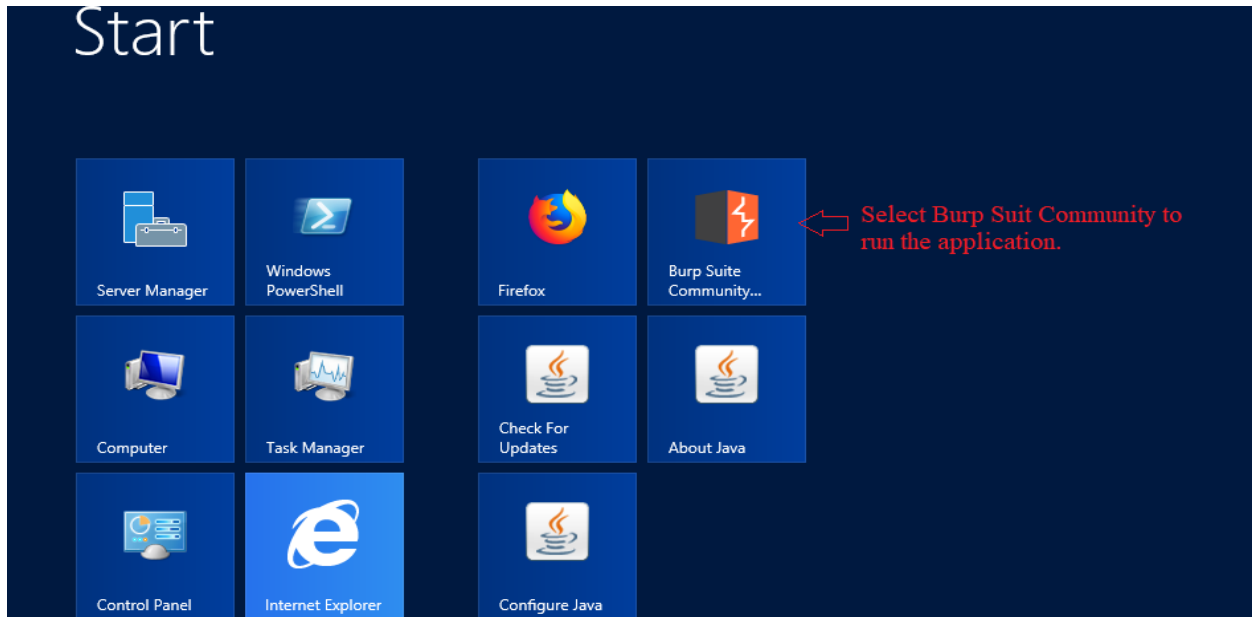
Password: guest

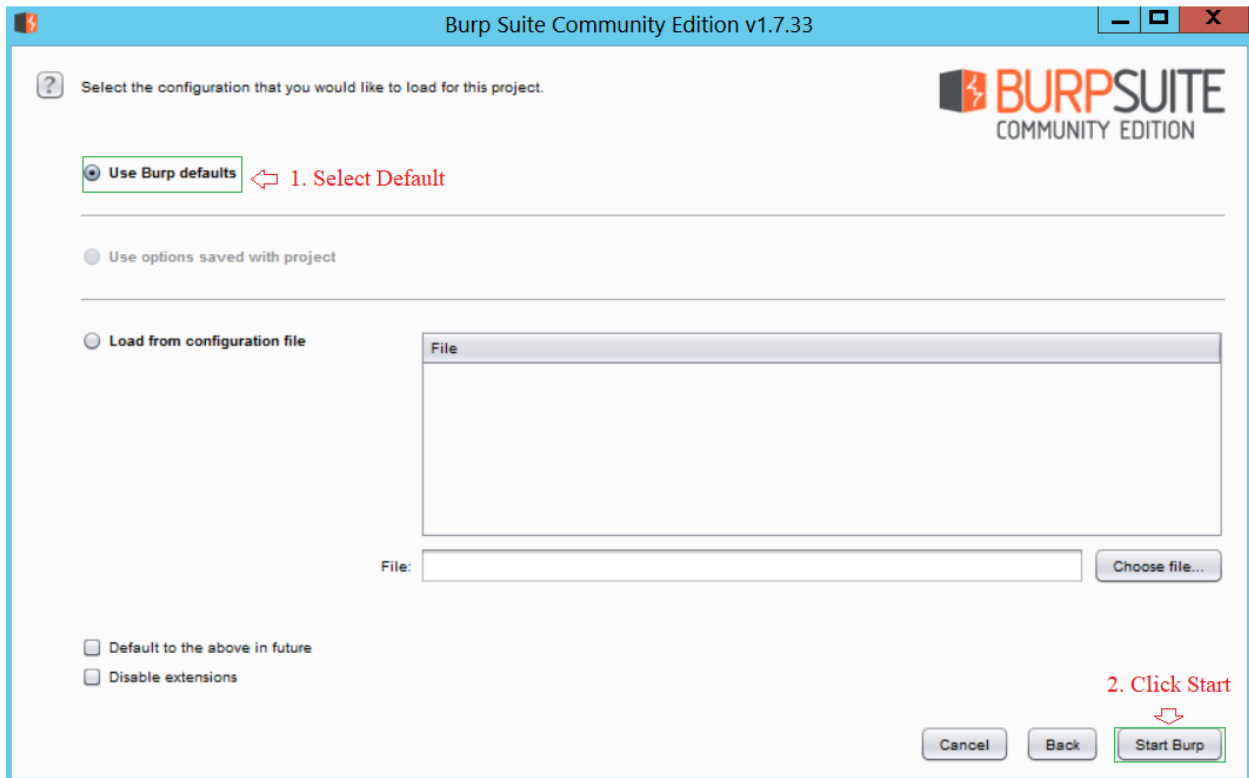


Webgoat welcome interface.

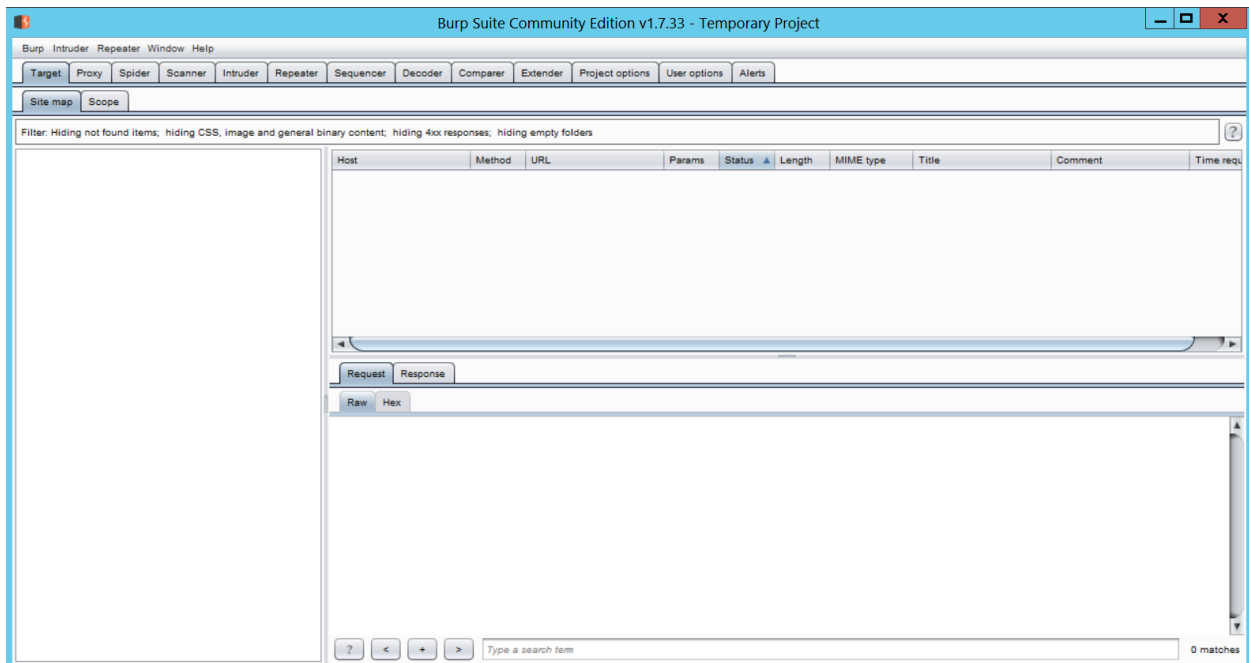


Most of these exercises will be done with Burp Suite, a full feature web proxy. Start burp from you windows server 2012 VM desktop or starup screen. The startup screen prompt you with an option to create or open an existing project. We will select a default option and continue with temporary project and select next and click start Burp on next window with the default option. It will take few minutes to load the dashboard.

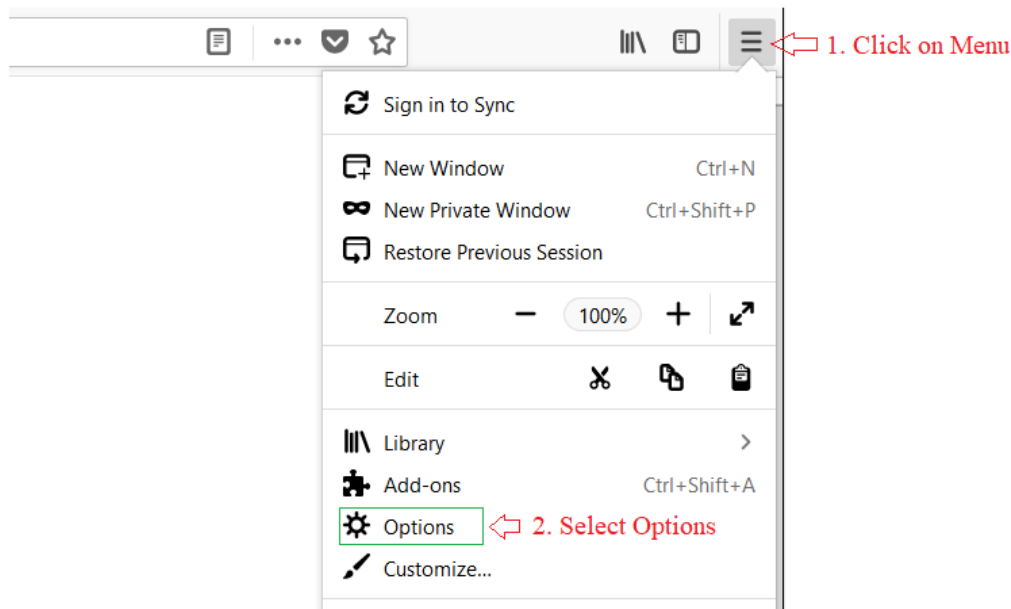




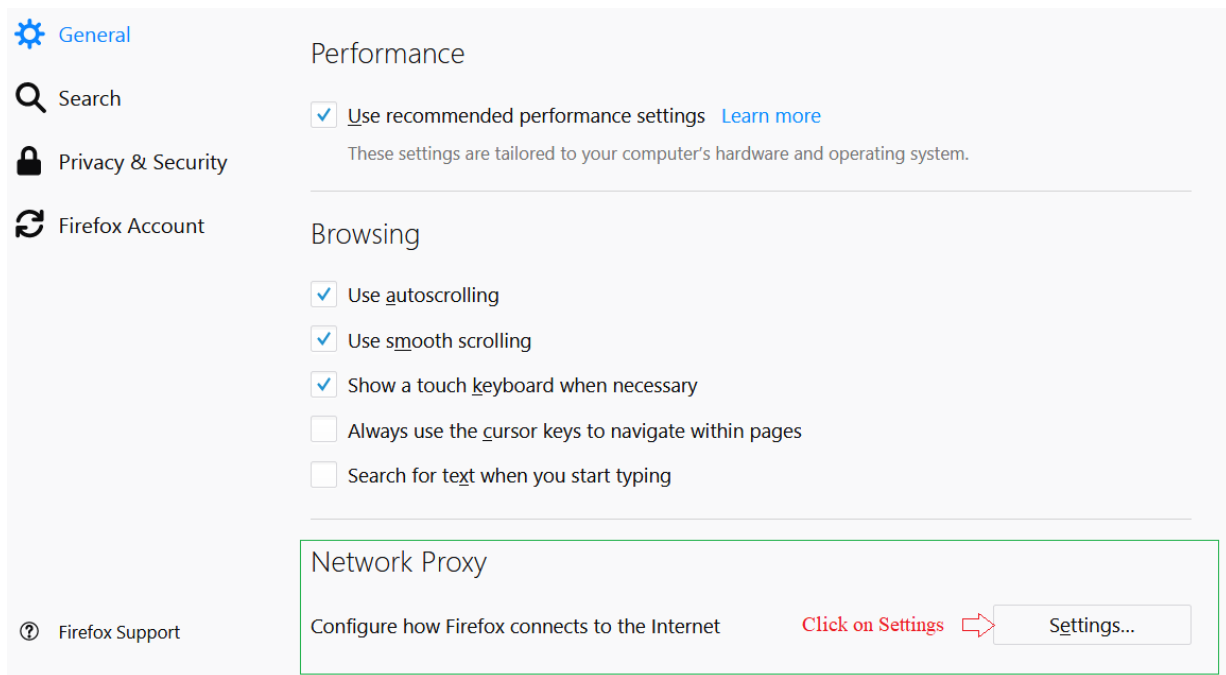
Burp Suit dashboard.



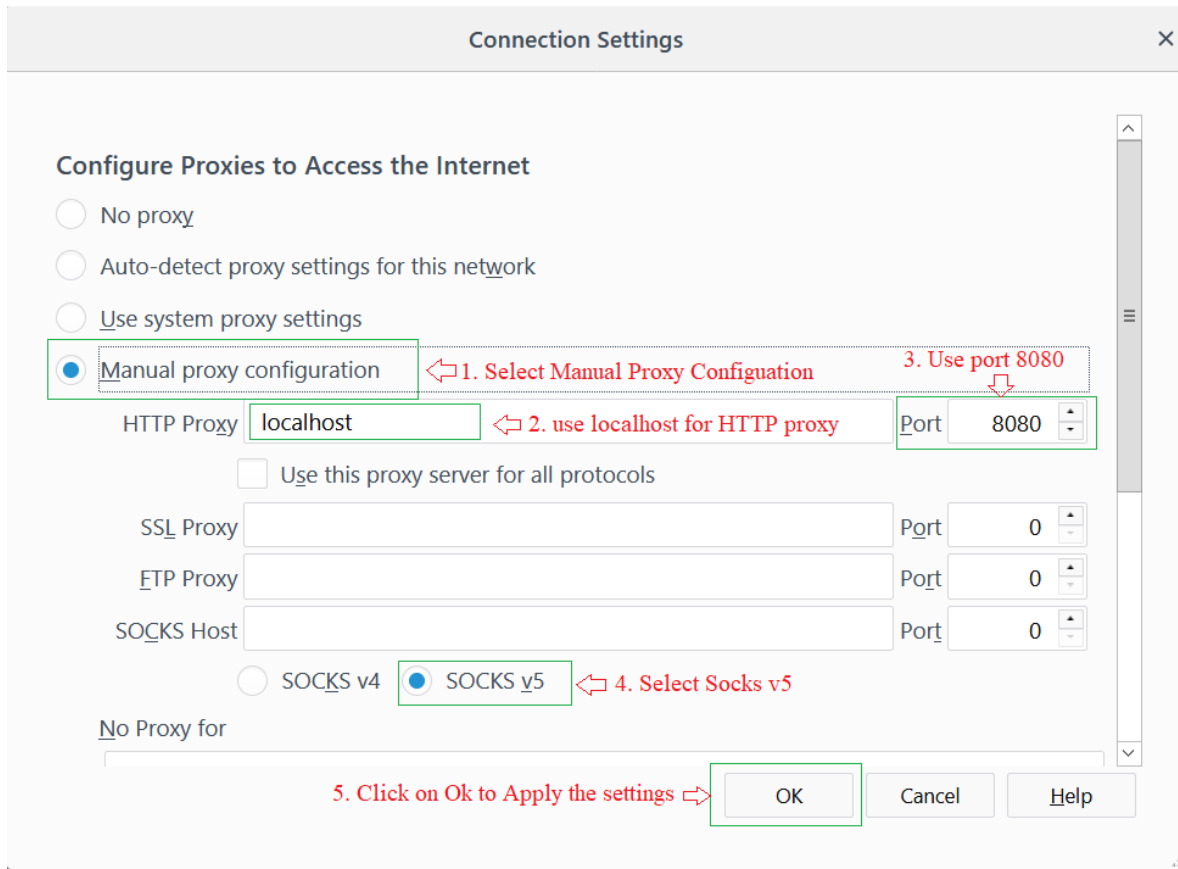
We now need to configure Firefox to route all web traffic the Burp suite. This can be done by selecting the menu button on the top right corner of your Firefox browser and select options.



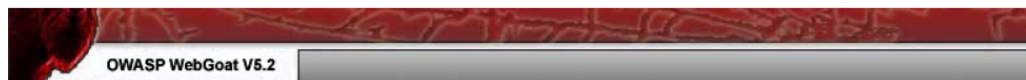
Inside General scroll down to the bottom for Network Proxy options and click on settings.



Use the following settings as shown on the screenshot. Select Manual proxy configuration use localhost as HTTP proxy with port number 8080 select Socky5 and click ok to apply the setting and close the options tab.



If you have set this up correctly, you can now click on the **Start WebGoat** button on the page open inside Firefox, and the **POST** request to the webserver from your Firefox browser should be captured by the Burp Suit as show (notice the intercept on button is in use):



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 lead by Bruce Mayhew. Please send all comments to Bruce at WebGoat@owasp.org.

Thanks to **OUNCE LABS** for supporting Bruce on the WebGoat Project.



WebGoat Design Team

Bruce Mayhew
David Anderson
Rogan Dawes
Laurence Casey (Graphics)

Lesson Contributors

Aspect Security
Sherif Koussa
Romain Brechet

Special Thanks for V5.2

Reto Lippuner
Marcel Wirth

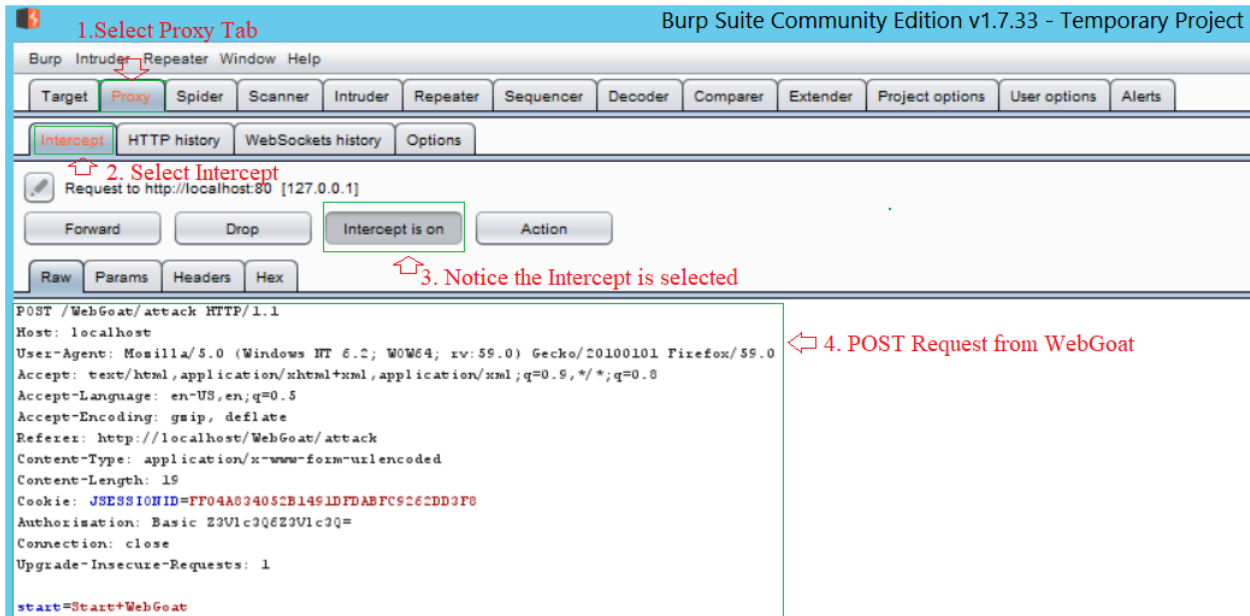
Documentation Contributors

Sherif Koussa
Aung Khant
(<http://yehg.org/>)
Erwin Geirnaert
(<http://www.zionsecurity.com/>)

To all who have sent comments

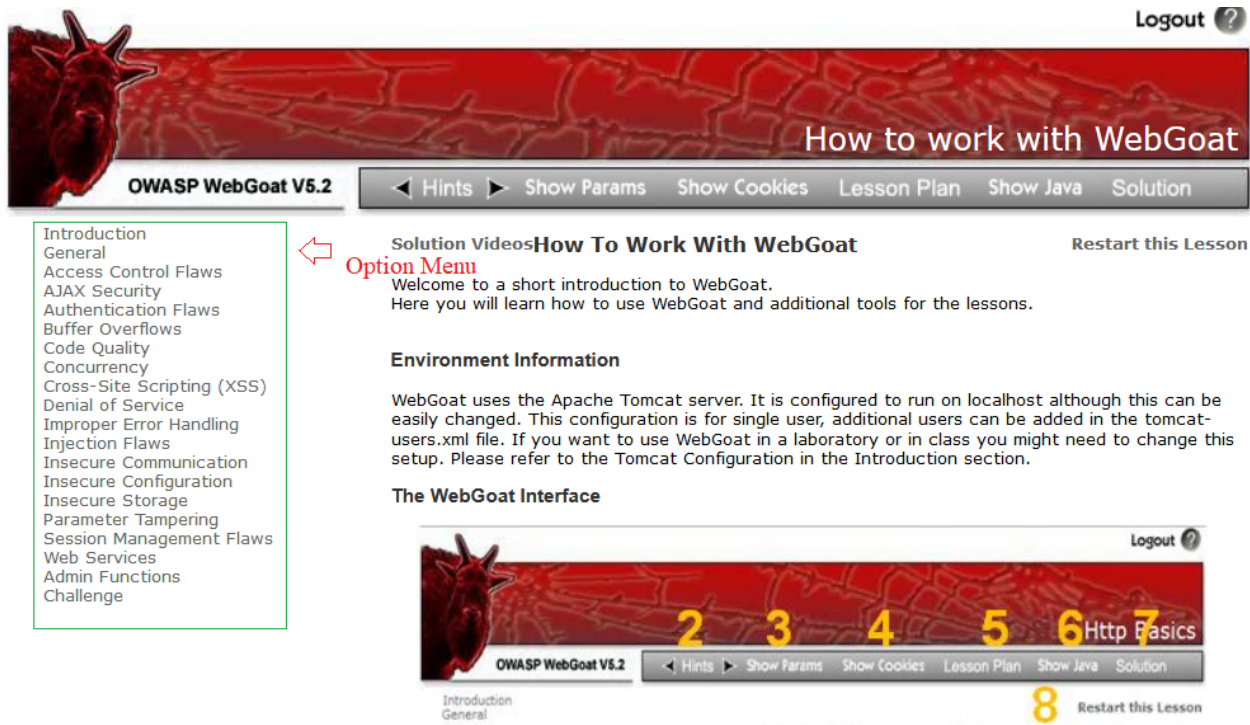
Start WebGoat

← Click Start WebGoat to POST Request



Click the **Forward** button as shown in the above screenshot to allow the request through the burp suit proxy.

Optionally, you can click on the **Intercept On** button in the Burp Suit to disable the holding of requests. This will allow all requests to flow through the burp suite proxy uninterrupted. After clicking on forward button, switch back to your Firefox and see a Webgoat Home page with different Options on Left panel.



Step 1

Go to the WebGoat application in your browser. Click on the **Parameter Tampering** link on the menu on the left, and select **Exploit Hidden Fields** as shown:

How to work with WebGoat

OWASP WebGoat V5.2

< Hints > Show Params Show Cookies Lesson Plan Show Java Solution

Restart this Lesson

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Denial of Service
Improper Error Handling
Injection Flaws
Insecure Communication
Insecure Configuration
Insecure Storage
Parameter Tampering
Exploit Hidden Fields
Exploit Unchecked Email
Bypass Client Side JavaScript Validation
Session Management Flaws
Web Services
Admin Functions
Challenge

Solution Videos **How To Work With WebGoat**

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

Environment Information

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

The WebGoat Interface

1. Click on parameter Tampering
2. Select Exploit Hidden Fields

Logout ?

2 3 4 5 6 Http Basics 7

OWASP WebGoat V5.2

< Hints > Show Params Show Cookies Lesson Plan Show Java Solution

8 Restart this Lesson

This page simulates a shopping cart feature on a website.

Purchase the TV normally to see how it work (it shows an amount charged to credit card of 2999)

Exploit Hidden Fields

< Hints > Show Params Show Cookies Lesson Plan Show Java Solution

Solution Videos Try to purchase the HDTV for less than the purchase price, if you have not done so already. **Restart this Lesson**

Shopping Cart

Shopping Cart Items -- To Buy Now	Price:	Quantity:	Total
56 inch HDTV (model KTV-551)	2999.99	1	\$2999.99

The total charged to your credit card: \$2999.99

Update Cart

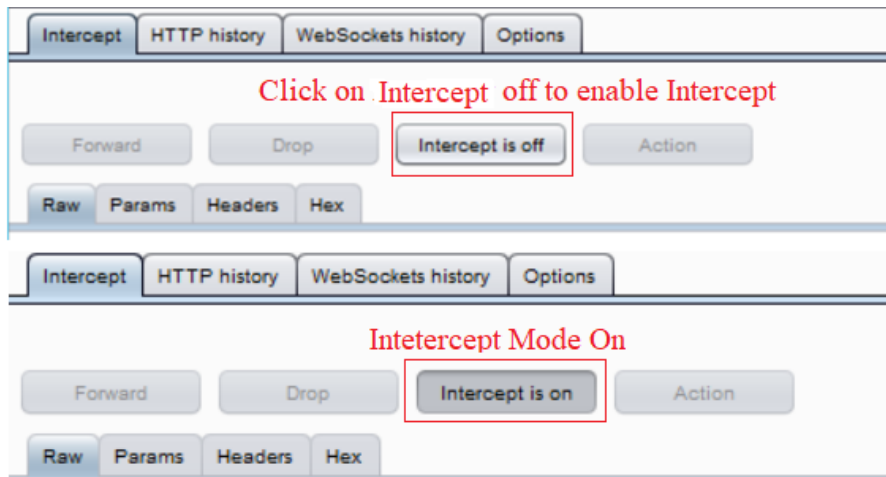
Purchase



POST request captured on Burp Suite for the request Exploit Hidden Field page.



Now, we want to enable Burp Suite to capture requests. Click the Intercept Off button in burp suite if it is set to Intercept Off. If it already shows Intercept On, do not click it.



Now, go back to the WebGoat application and click the purchase Button.

Solution Videos Try to purchase the HDTV for less than the purchase price, if you have not done so already. [Restart this Lesson](#)

Shopping Cart

Shopping Cart Items -- To Buy Now	Price:	Quantity:	Total
56 inch HDTV (model KTV-551)	2999.99	1	\$2999.99

The total charged to your credit card: \$2999.99

[Update Cart](#)

[Purchase](#)

↔ Click on Purchase



Change the contents of the Price field to 5.00 and click Forward to complete the attack

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=53&menu=1600 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=53&menu=1600
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Cookie: JSESSIONID=FF04A834052B1491DFDABFC9262DD3F8
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

QTY=1&SUBMIT=Purchase&Price=2999.99 ⇐ Update the Price to 5.00

QTY=1&SUBMIT=Purchase&Price=5.00
```

You will see the new price you added. If this were a real application, it would get submitted to the company. If there is no manual checking of the prices, and automated fulfillment, the item would be send out.

Solution Videos Try to purchase the HDTV for less than the purchase price, if you have not done so already. [Restart this Lesson](#)

*** Congratulations. You have successfully completed this lesson.**

Your total price is: **\$5.0**

This amount will be charged to your credit card immediately.



Exercise 2 Cross Site Scripting (XSS)

In this lab we will be looking at how to perform a Cross Site Scripting attack against a “stored” Cross Site Scripting vulnerability. The general goal of a Cross Site Scripting attack is to have a user other than yourself execute a script/function on your behalf using their permission. One common thing is executing password change, where a user unknowingly changes their password to something picked by the attacker via a script. Another example and probably the most common demonstration is having a user send the attacker his/her session cookies or tokens. This is precisely the attack we’ll demonstrate in this exercise.

To begin, go back to the WebGoat start page. Remember if you've still got the *intercept on* enabled in Burp, you'll need to make sure you go there and hit the *forward* button for each browsing action.

Once you're back at the WebGoat start page, Select Cross Site Scripting > Stage 1 Stored XSS.

The screenshot shows the OWASP WebGoat V5.2 interface. At the top, there is a navigation bar with buttons for Hints, Show Params, Show Cookies, Lesson Plan, Show Java, and Solution. Below this, a list of topics is shown on the left, with 'Cross-Site Scripting (XSS)' highlighted. Underneath, 'Stage 1: Stored XSS' is selected. The main content area displays a message: 'Solution Videos Try to purchase the HDTV for less than the purchase price, if you have not done so already.' followed by a red notification: '* Congratulations. You have successfully completed this lesson.' Below this, it says 'Your total price is: \$5.0' and 'This amount will be charged to your credit card immediately.' At the bottom right, there is a logo for ASPECT SECURITY Application Security Specialists and a footer with 'OWASP Foundation | Project WebGoat | Report Bug'.

Just another friendly reminder, make sure you hit the forward button on the Burp Suite proxy if you have it enabled, otherwise it'll appear nothing is happening when you browse from step to step.

The instructions say the following;

As Tom, execute a stored XSS attack against the stree field on the edit profile page. Verify that jerry is affected by the attack.

So, we'll actually be doing this as Tom. But we'll want our script injection to affect Jerry. First step is to login as Tom. As the instructions tell us, each user's password is simply the all lower-case version of the username. To login as Tom, click the drop-down button and select Tom's name form the list.

The screenshot shows the 'LAB: Cross Site Scripting' page. A dropdown menu is open, listing various users: Larry Stoooge (employee), Moe Stoooge (manager), Curly Stoooge (employee), Eric Walker (employee), Tom Cat (employee), Jerry Mouse (hr), David Giambi (manager), Bruce McGuirre (employee), Sean Livingston (employee), Joanne McDougal (hr), John Wayne (admin), Neville Bartholomew (admin), and Larry Stoooge (employee). The 'Tom Cat (employee)' option is highlighted. Below the dropdown, there is a 'Password' input field and a 'Login' button.

Enter “tom” as the password for the Tom Cat account, then select the login button. After logged in, select the “View profile” option.

3. We need to Forward the POST request to successfully login.

1. Make note of Intercept is enable or off

2. View the Post method detail and note the login details.

```
Request to http://localhost:80 [127.0.0.1]
Forward Drop Intercept is on Action
Raw Params Headers Hex
POST /WebGoat/attack?Screen=33&menu=900 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Cookie: JSESSIONID=FF04A834052B1491DFDABFC9262DD3F8
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1
employee_id=105&password=tom&action=Login
```

Next, you'll want to select the employee form the list and click Edit Profile button.

Solution Videos Stage 1: Execute a Stored Cross Site Scripting (XSS) attack. **Restart this Lesson**
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack. The passwords for the accounts are the prenames.

Goat Hills Financial
Human Resources

Welcome Back Tom - Staff Listing Page

Select from the list below

Tom Cat (employee)

1. Select the employee from the list

SearchStaff

ViewProfile

2. Click on ViewProfile

Logout

Next you'll want to select the Edit Profile button.

Solution Videos Stage 1: Execute a Stored Cross Site Scripting (XSS) attack. **Restart this Lesson**
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack. The passwords for the accounts are the prenames.

Goat Hills Financial
Human Resources

Welcome Back Tom

First Name:	Tom	Last Name:	Cat
Street:	2211 HyperThread Rd.	City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	106
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

Employee Information

ListStaff **EditProfile** Logout

Click on EditProfile to make changes on Profile Details.

Here we'll see all the editable fields for this profile. In the street field we will attempt to add a script that would be executed by any other user viewing Tom's Profile.

Goat Hills Financial
Human Resources

Welcome Back Tom

1. Add Script Here →

First Name:	<input type="text" value="Tom"/>	Last Name:	<input type="text" value="Cat"/>
Street:	<input)<="" script>"="" type="text" value="id by infosec"/>	City/State:	<input type="text" value="New York, NY"/>
Phone:	<input type="text" value="443-599-0762"/>	Start Date:	<input type="text" value="1011999"/>
SSN:	<input type="text" value="792-14-6364"/>	Salary:	<input type="text" value="80000"/>
Credit Card:	<input type="text" value="5481360857968521"/>	Credit Card Limit:	<input type="text" value="30000"/>
Comments:	<input type="text" value="Co-Owner."/>	Manager:	<input type="text" value="Tom Cat"/>
Disciplinary Explanation:	<input type="text" value="NA"/>	Disciplinary Action Dates:	<input type="text" value="0"/>

ViewProfile **UpdateProfile** Logout

2. Click Update to add Script.

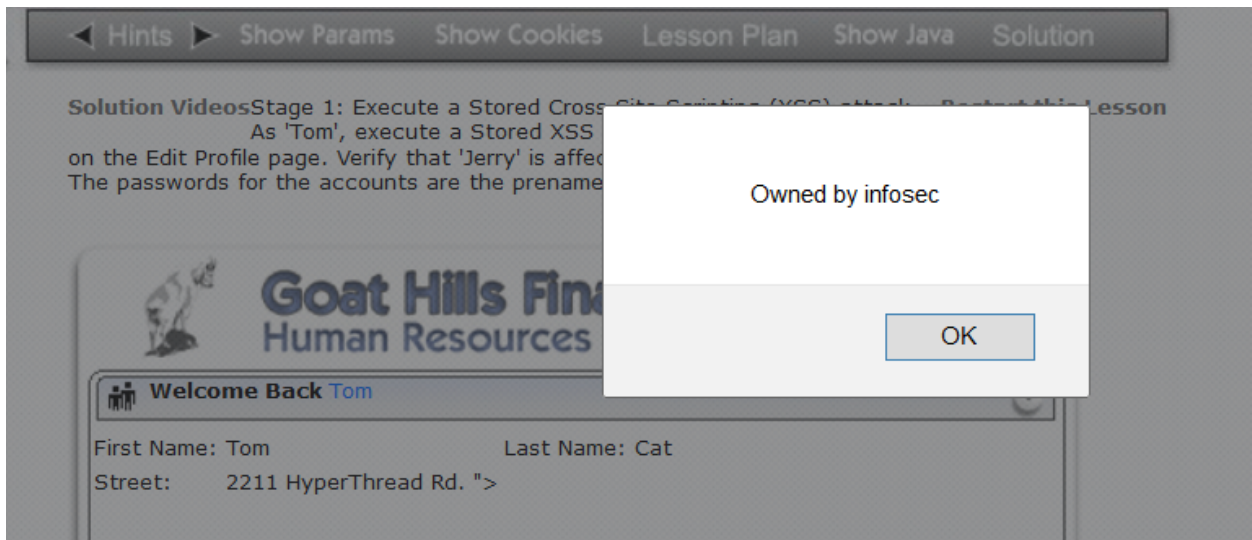
Next, you'll want to make the street field read exactly as it is shown below. You'll be adding the string.

```
"><script>alert("Owned by infosec")</script>
```

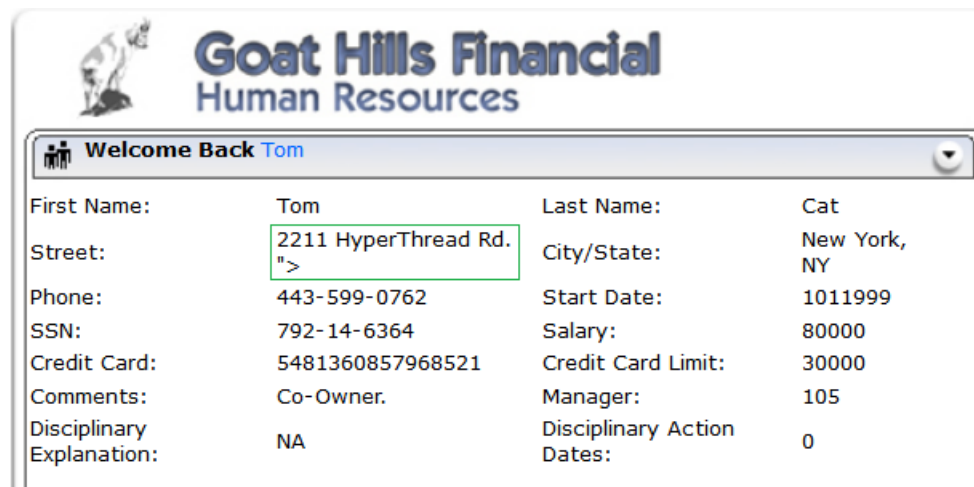
To what's already there. So the complete entry in the street field should read as follows.

```
221 HyperThread Rd. "><script>alert("Owned by infosec")</script>
```

After making the changes, go ahead and click the "Update Profile" button at the bottom to save these changes. You should immediately see a popup alert that says, "Owned by Infosec". See below.

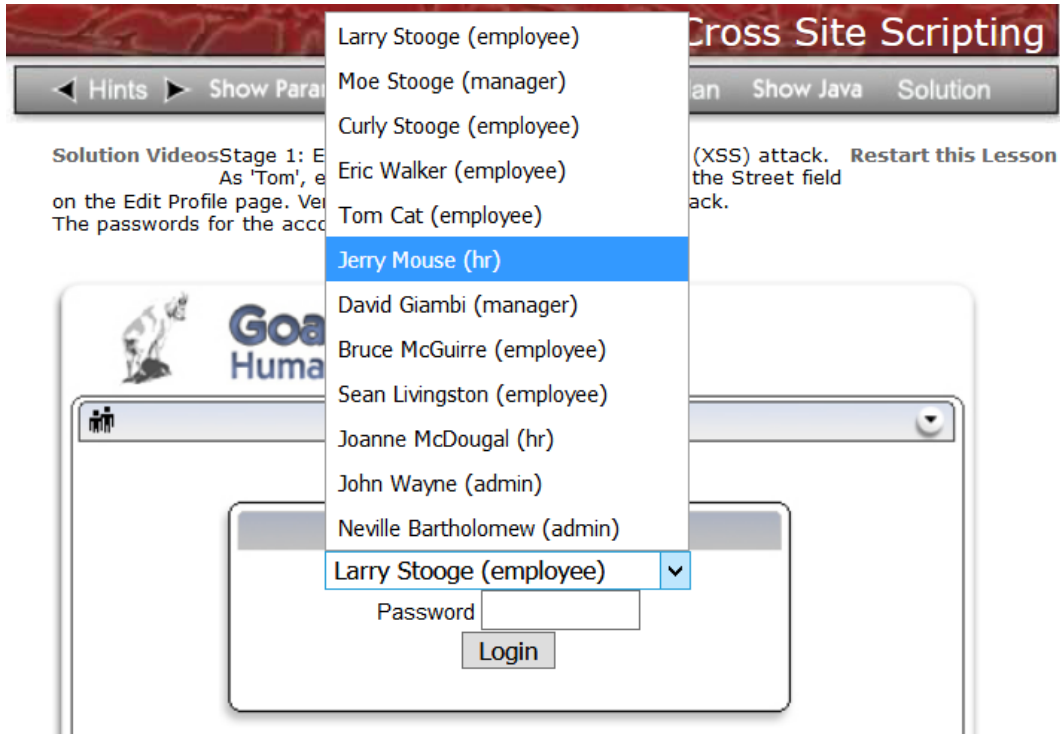


Click Ok and notice the change on Tom Street address field.

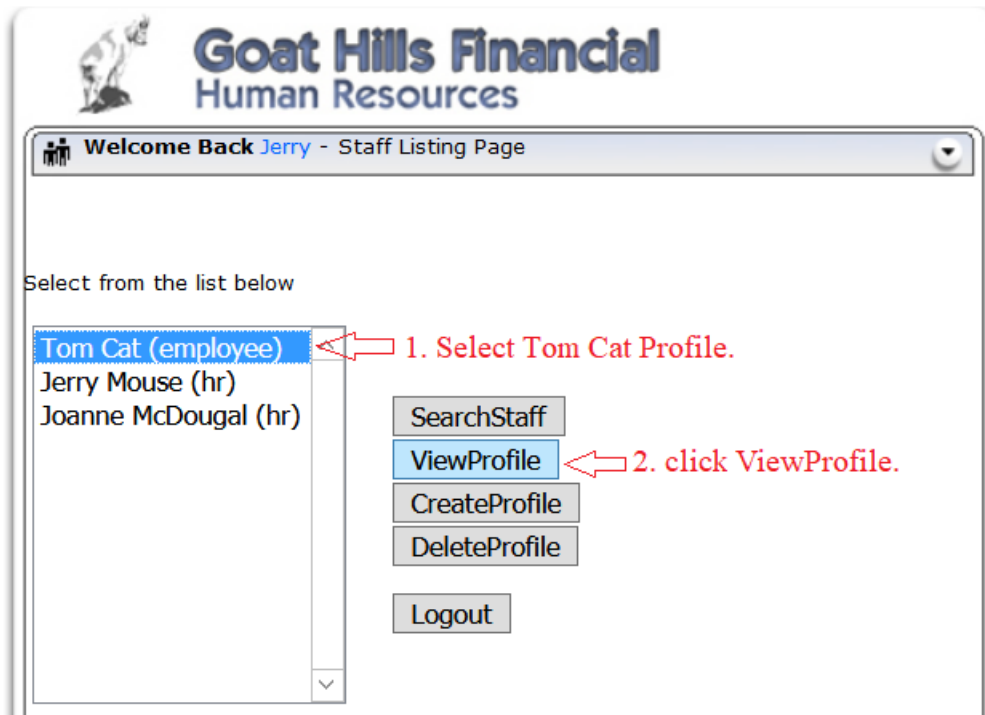


So the goal is Jerry Should basically see this same popup when he views our (Tom's) profile. If he see the script, then that means Tom has been able to successfully "store" a script on this website that is transparently executed by Jerry upon Jerry viewing Tom's profile. This would essentially mean that anyone viewing Tom's profile would execute his script in their browser.

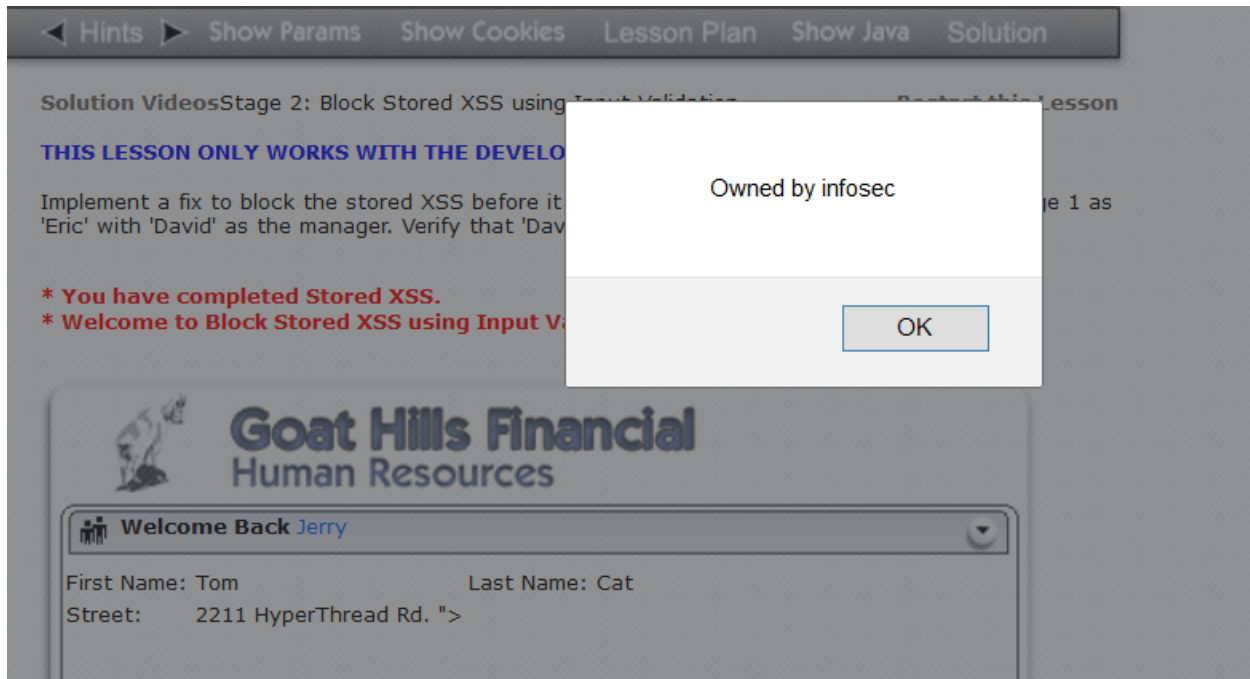
Let's test it out and see. You'll need to hit the **logout** button and the button right. Then login again as Jerry.



Remember, Jerry's password is simply Jerry in all lowercase format. Once you're logged in as Jerry, select Tom's account, then select the "View Profile" button on the right.



Selecting the ViewProfile button should cause the Owned by Infosec popup to now show up in Jerry's browser session. See below.



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

1. Select Cross-Site Scripting

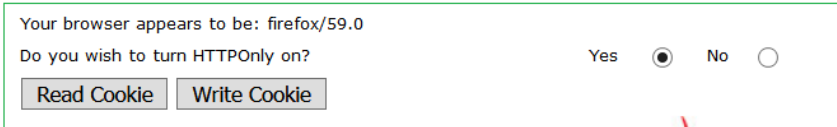
Solution Videos To help mitigate the cross site scripting threat, Microsoft has introduced a new cookie attribute entitled 'HttpOnly.' If this flag is set, then the browser should not allow client-side script to access the cookie. Since the attribute is relatively new, several browsers neglect to handle the new attribute properly.

Restart this Lesson
For a list of supported browsers see: OWASP HTTPOnly Support

General Goal(s):

The purpose of this lesson is to test whether your browser supports the HTTPOnly cookie flag. Note the value of the **unique2u** cookie. If your browser supports HTTPOnly, and you enable it for a cookie, client side code should NOT be able to read OR write to that cookie, but the browser can still send its value to the server. Some browsers only prevent client side read access, but don't prevent write access.

With the HTTPOnly attribute turned on, type "javascript:alert(document.cookie)" in the browser address bar. Notice all cookies are displayed except the unique2u cookie.



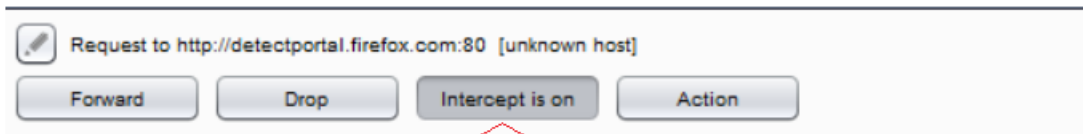
Make Sure HTTPOnly is turn On before you turn on Intercepted Mode back On at BurpSuite.

2. Click on HTTPOnly Test



OWASP Foundation | Project WebGoat | Report Bug

You'll want to make sure you've enabled intercept in the Burpsuite proxy.



Turn Intercept Mode back On

Once you've made sure intercept is on, you'll want to go back to the WebGoat page and select the radio button turn off HTTPOnly. See below.

With the HTTPOnly attribute turned on, type "javascript:alert(document.cookie)" in the browser address bar. Notice all cookies are displayed except the unique2u cookie.

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?



Turn Off HTTPOnly
Yes No



OWASP Foundation | Project WebGoat | Report Bug

Once you make this change you should see your Burpsuite icon at the bottom of the screen flashing to let you know it's intercepted some data. Go ahead and switch to the Burpsuite screen. Notice the HttpOnly variable is set to false.

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=148&menu=500 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=148&menu=500
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Cookie: unique2u=B/bZmC9LxsnL9PsYLWFKM1pGYXI=; JSESSIONID=E087D2AF4463A2D8ED40354A4ABDC069
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

httponly=False&read_result=
```

Go ahead and hit the Forward button to pass the change.

Next, we're going to see if we can read a cookie while the HttpOnly flag is turned off. Click the Read Cookie button. You should see the following cookie pop up.

unique2u=DG2NyNtp9/3pqRrHBkPaCL8PeNE=; JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D

OK

Do you wish to turn HTTPOnly on? Yes No

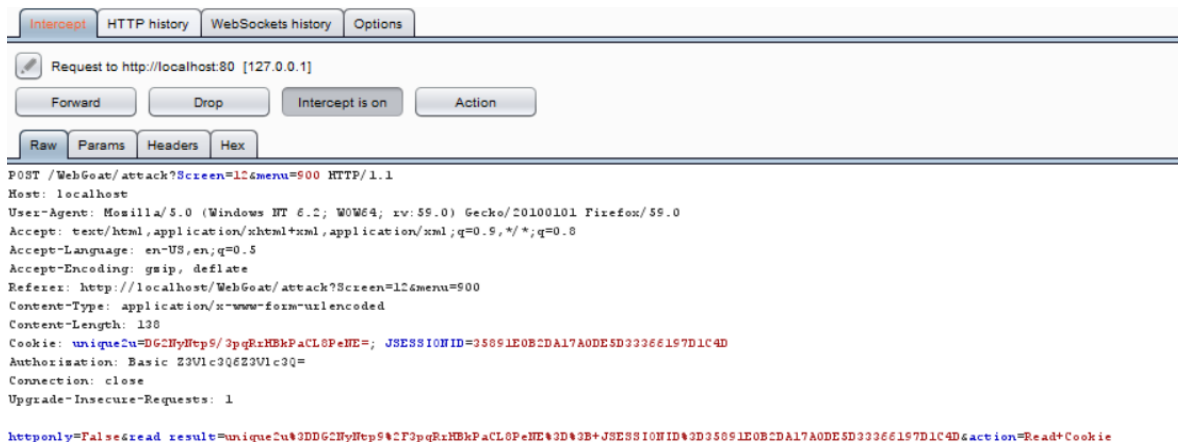
Read Cookie Write Cookie

ASPECT SECURITY
Application Security Specialists

OWASP Foundation | Project WebGoat | Report Bug

Go ahead and click Ok to the popup. You should now see Burpsuite intercept more traffic.

We can see that the httponly flag is set to false. So therefore the script that's being executed to read the cookie is able to run. This exercise is simply doing the script for you that we did in the earlier XSS lab where caused the Infosec popup. Go ahead and hit the Forward button and return to WebGoat. You should see the following message in red.



With the HTTPOnly attribute turned on, type "javascript:alert(document.cookie)" in the browser address bar. Notice all cookies are displayed except the unique2u cookie.

*** Since HTTPOnly was not enabled, the 'unique2u' cookie was displayed in the alert dialog.**

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?

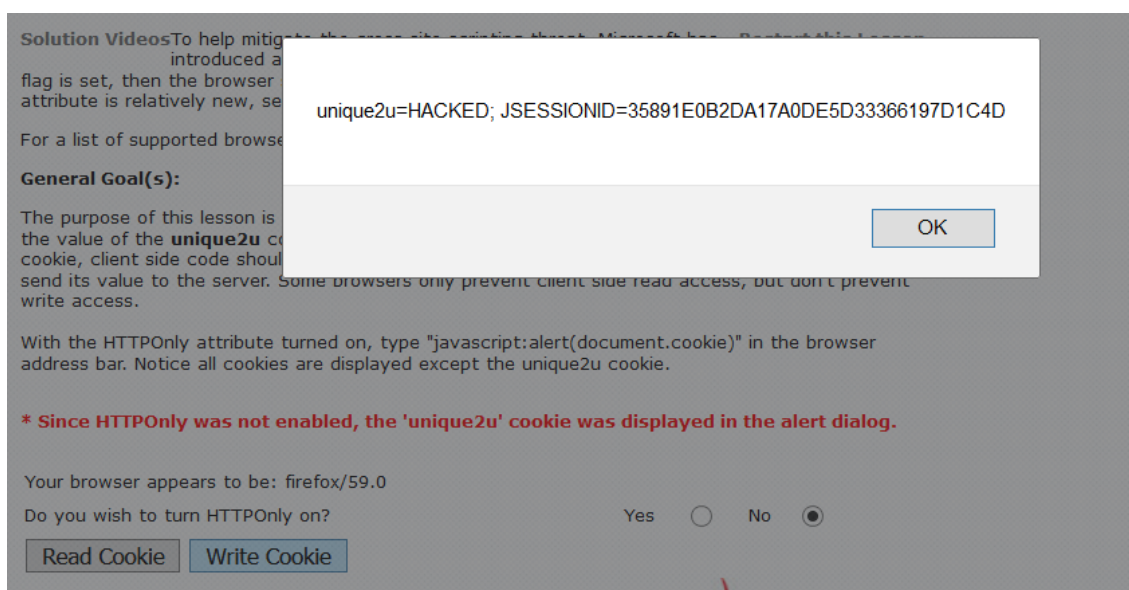
Yes No

Read Cookie

Write Cookie



Next Select the Write Cookie button. Again you'll see a cookie popup and upon hitting OK. You should see traffic in Burpsuite again. This time you'll see the Burpsuite has intercepted a "write cookie" action.



Intercept HTTP history WebSockets history Options

Request to http://localhost:80 [127.0.0.1]

Forward Drop Intercept is on Action

Raw Params Headers Hex

```

POST /WebGoat/attack?Screen=12&menu=900 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.0; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=12&menu=900
Content-Type: application/x-www-form-urlencoded
Content-Length: 47
Cookie: unique2u=HACKED; JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

httponly=False&read_result=&action=Write+Cookie

```

Click forward to allow the request back to web browser.

With the HTTPOnly attribute turned on, type "javascript:alert(document.cookie)" in the browser address bar. Notice all cookies are displayed except the unique2u cookie.

*** Since HTTPOnly was not enabled, the browser allowed the 'unique2u' cookie to be modified on the client side.**

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?

Yes No

Read Cookie

Write Cookie



Now go back to WebGoat and this time we're going to turn on the HTTPOnly functionality. Do this by selecting the "Yes" radio button to the right. See below.

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?

Yes No

Read Cookie

Write Cookie

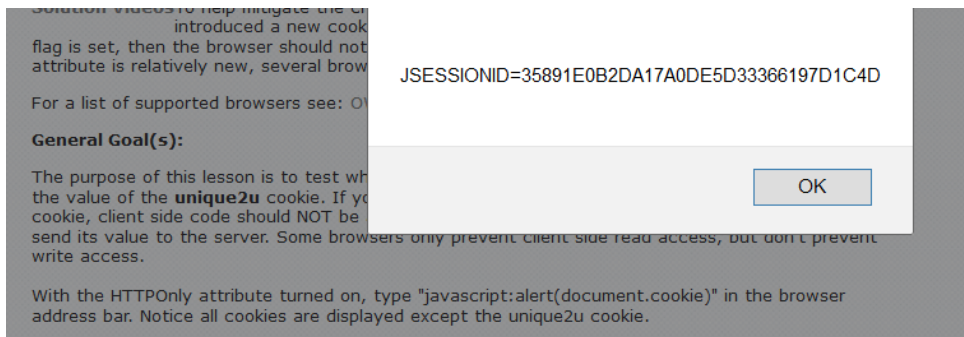


Of course, after making this change, you will see alert flashes from Burpsuite yet again. This is because selecting the “Yes” radio button has dynamically changed what’s displayed and changed the functionality of the html to an extent. Go ahead and switch to Burpsuite and hit the forward Button.

```
POST /WebGoat/attack?Screen=12&menu=900 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=12&menu=900
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Cookie: unique2u=Gp+XB+uaIPg2dD0s/ZHyR96ZRMc=; JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

httponly=True&read_result=
```

Now select the Read Cookie button as we did previously. You should see that the popup contains only JSESSIONID.



Go ahead and hit the Ok button. As was before, this will cause another alert in Burpsuite.

Switch the Burpsuite and notice that even though the cookie is there the script was not able to read all the information like unique2u information on cookie in the popup.

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=12&menu=900 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=12&menu=900
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Cookie: unique2u=Gp+XB+uaIPg2dD0s/ZHyR96ZRMc=; JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

httponly=True&read_result=JSESSIONID*3D35891E0B2DA17A0DE5D33366197D1C4D&action=Read+Cookie
```

Hit the forward button and return to WebGoat. You should see that it tells you you've successfully prevented the script from reading the cookie because HttpOnly was enabled.

- * **SUCCESS: Your browser enforced the HTTPOnly flag properly for the 'unique2u' cookie by preventing direct client side read access to this cookie.**
- * **Now try to see if your browser protects write access to this cookie.**

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?

Yes No



OWASP Foundation | Project WebGoat | Report Bug

Repeat these steps by hitting the “write Cookie” button. Once you get back to Burpsuite, the results should look like the following.

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=12&menu=900 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=12&menu=900
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Cookie: unique2u=Gp+XB+uaIPg2dD0s/ZHyR96ZRMc=; JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorisation: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

httponly=True&read_result=&action=Write+Cookie
```

Go ahead and hit the “Forward” button.

Now return to WebGoat to see that you're completed the lesson.

General Goal(s):

The purpose of this lesson is to test whether your browser supports the HTTPOnly cookie flag. Note the value of the **unique2u** cookie. If your browser supports HTTPOnly, and you enable it for a cookie, client side code should NOT be able to read OR write to that cookie, but the browser can still send its value to the server. Some browsers only prevent client side read access, but don't prevent write access.

With the HTTPOnly attribute turned on, type "javascript:alert(document.cookie)" in the browser address bar. Notice all cookies are displayed except the unique2u cookie.

- * **SUCCESS: Your browser enforced the write protection property of the HTTPOnly flag for the 'unique2u' cookie by preventing client side modification.**
- * **Congratulations. You have successfully completed this lesson.**

Your browser appears to be: firefox/59.0

Do you wish to turn HTTPOnly on?

Yes No

Exercise 4 Basic SQL Injection

In this WebGoat exercise, we'll be looking at how to perform basic SQL Injection. We'll first do the string injection exercise, then later follow that up with doing a "blind" injection attack. Here's a short definition of what SQL injection is;

"A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operation on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system". -Source OWASP

Let's take a username and login field on a web form for example. Typically when you enter a username and password to login to a database via a web form (such as an online bank login), the username and password are combined to form part of a sql query that basically says "If the username and password I enter match what's in the database (or form a true statement), then do some action on this users behalf." As it turns out there are many ways to make the username and password part of the query equal "true". This is precisely what we'll be doing in this exercise.

Go back to your WebGoat start page and click on Injection Flaws> LAB: Sql Injection> Stage 1: String SQL Injection.

OWASP WebGoat V5.2 LAB: SQL Injection

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Denial of Service
Improper Error Handling
Injection Flaws
Command Injection
Blind SQL Injection
Numeric SQL Injection
Log Spoofing
XPath Injection
LAB: SQL Injection
Stage 1: String SQL Injection
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection

Solution Videos Stage 1: Use String SQL Injection to bypass authentication. **Restart this Lesson**
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).

Goat Hills Financial
Human Resources

Please Login
Larry Stooge (employee) ▾
Password
Login

1. click on Injection Flaws
2. Select SQL Injection
3. Select String SQL Injection

After clicking each link, remember you'll have to go to Burpsuite and hit the "Forward" button to pass the request through the Burpsuite proxy.

The instructions say that we are to try and use the Neville account which has admin privileges to get to the point that we're able perform an admin function. We'll go to that, but first let's look at some basic way to see what's being sent to the application, how it's being sent, and whether or not there are any apparent client-side controls pushed to the browser.

First, we'll use the first account in the list to try and see what we can get passed to the Post request that goes to the application.

You should see activity in the Burpsuite proxy. Locate the where the username and password strings are being passed in the proxy. See below.



```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=54&menu=1200 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=54&menu=1200&stage=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Cookie: JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=101&password=infosec&action=Login
```

Notice we can see the infosec string we entered. It's also important to note that the username has been converted to a numeric value and it's being passed using a field identified as employee_id. If we were doing more advanced injection attack such as using SQL functions like INSERT, UNION, etc. We would have to tell SQL to perform the action to a specific field in the database. We now know there's field named "password" and one named "employee_id".

Let's try some basic tests to begin with. In the request that we have paused, let's change the password string of infosec to a single quote ('). See the change below.

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=54&menu=1200 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=54&menu=1200&stage=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Cookie: JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=101&password='&action=Login
```

After making the change, go ahead and hit the “Forward” button. If you go back to the WebGoat page, you should see that the login simply failed.

- * Error logging in
- * Login failed



You should be back at the login field. Now click the drop down and change the login name to the Neville admin account. Repeat the process, using the password of infosec. After you’ve captured the request in Burpsuite. Make the following change;

After the password identifier, enter the following ‘OR 1=1--

That is single quote, OR space, one. Equal sign, one, dash dash. See below.



Goat Hills Financial Human Resources

Please Login

Neville Bartholomew (admin) ▾

Password ●●●●●●

Login

Password: infosec

```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=54&menu=1200 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=54&menu=1200
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Cookie: JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=112&password=infosec&action=Login
```

Click Forward to pass the modified password request.

Request to http://localhost:8080/

Forward Drop Intercept is on Action

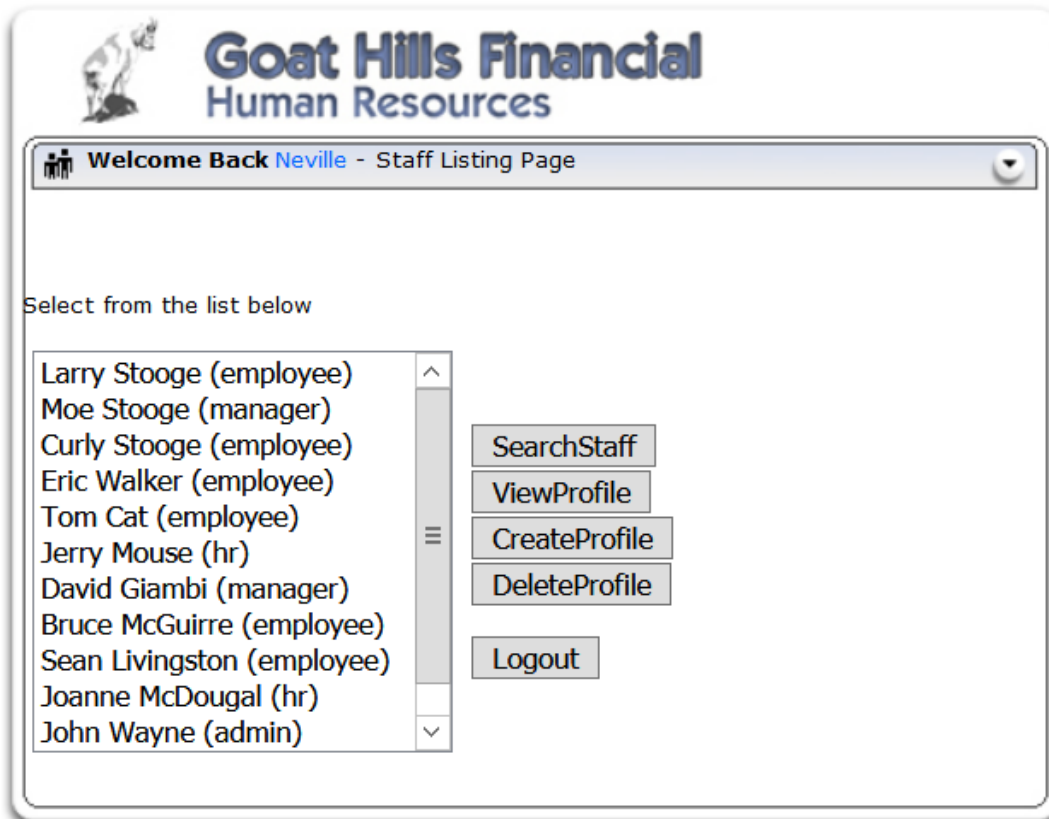
```
Raw Params Headers Hex
POST /WebGoat/attack?Screen=54&menu=1200 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/WebGoat/attack?Screen=54&menu=1200
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Cookie: JSESSIONID=35891E0B2DA17A0DE5D33366197D1C4D
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=112&password='OR 1=1--&action=Login
'OR 1=1--
```

Next go ahead and hit the “Forward” button to pass the request. If you typed everything exactly right. You should notice now that on the WebGoat page, you’re informed that you’ve completed this lesson.

Solution Videos Stage 2: Block SQL Injection using a Parameterized Query. **Restart this Lesson**
Implement a fix to block SQL injection into the fields in question on the Login page. Repeat stage 1. Verify that the attack is no longer effective.

- * **You have completed String SQL Injection.**
- * **Welcome to Parameterized Query #1**



At this point, if we wanted to, we could delete profiles, view profiles and do other functions under the privilege of the admin Neville admin account. Let's move on to the next exercise and do Blind SQL Injection using numeric values and operators.

Exercise 5 Blind SQL Injection

NOTE: You need to have WebGoat Version 5.4 to complete Blind SQL Injection. You can download the zip file like the earlier version, but you need to make sure you must only run one WebGoat version at a time. WebGoat version 5.4.zip will be provide by the instructor along with the lab contents.

In this exercise we're going to work on the premise that we've got a stolen credit card or at least, we know the credit card number. We're then going to try and do Blind SQL Injection to a credit card database and extract the card's PIN. Basically, we'll use a comparison operator to try and get SQL to tell us when we've gotten close to the correct PIN, then eventually narrow it down to

the extract PIN associated with the given credit card number. We're given a few of the table names, which are generally created using standard naming conventions.

Start by going back to the WebGoat start page and selecting Injection Flaws > Lab:SQL Injection > Blind Numeric SQL Injection.

See Below.

OWASP WebGoat v5.4

Blind Numeric SQL Injection

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Command Injection
Numeric SQL Injection
Log Spoofing
XPath Injection
String SQL Injection
LAB: SQL Injection
Stage 1: String SQL Injection
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection
Stage 4: Parameterized Query #2
Modify Data with SQL Injection
Add Data with SQL Injection
Database Backdoors
Blind Numeric SQL Injection
Blind String SQL Injection

Solution Videos [Restart this Lesson](#)

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

1. Select Injection Flaws

Enter your Account Number:

Account number is valid.

Created by Chuck Willis **MANDIANT**
INTELLIGENT INFORMATION SECURITY

OWASP Foundation | Project WebGoat | Report Bug

2. SQL Injection Lab

3. Blind Numeric SQL Injection

Based on the instructions, we are to find the value in the field *field pin* which is part of the *table pins* for the *row* with the **cc_number** of **1111222233334444**.

So behind this form there is a database with a table named pins, which has a field name pin, which also has a row named cc_number. And we have a known credit card number. The form we're presented with is basically created to check a number to see if it's a valid account number. There is probably a field in the database named "account_number" or something like that. But we don't know that for sure, and right now it's not required that we know it, as we're simply using this part of the form to create a query to try and access other parts of the database.

You'll want to make sure you turn the intercept function off in the Burpsuite proxy before starting this exercise.

First let's do a simple test and find out if we can quickly discern the approximate number of valid accounts. Change the 101 in the field to 102 and hit the Go button.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

It should have returned the message to inform you that 102 is a valid account. Repeat this with 103, 104, etc.

Notice how will the 104 number being entered the message changes to say the account number is not valid. Now got down from 101. Try 100, then 99 and continue down until you get the invalid account number message again. What you should have figured out is that the valid account number appear to range from 101 to 103. Now for the blind part. Since we know some valid account numbers, we can try to use them to pass other commands to the database on the backend.

Enter the following query in the form field;

101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 100);

Now go ahead and hit the Go button. Immediately you should see that the message that confirms a valid account number returns again. This means that what we've just said is "TRUE" as far as SQL is concerned.

OWASP WebGoat v5.4

[Hints](#) [Show Params](#) [Show Cookies](#) [Lesson Plan](#) [Show Java](#) [Solution](#)

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
[Command Injection](#)
[Numeric SQL Injection](#)
[Log Spoofing](#)
[XPath Injection](#)
[String SQL Injection](#)
[LAB: SQL Injection](#)
[Stage 1: String SQL Injection](#)

Solution Videos [Restart this Lesson](#)

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 100);

Enter your Account Number: **Hit Go!**

Account number is valid.

Created by Chuck Willis **MANDIANT**
INTELLIGENT INFORMATION SECURITY

So, what we know now is that pin for the given credit card number is greater than 100. Awesome! That really narrows it, down right?

Now let's repeat that with a higher number. Change the number at the end of your query from 100 to 1000. Then hit enter again.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.



When you hit the Go button, the “Account number is valid” response remains. So that means we now know the credit card PIN is more than 1000.

Now change the value to 10000. Hit Go again.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Invalid account number.



OWASP Foundation | Project WebGoat | Report Bug

Whoa! We got a different response. This means that we just said something to SQL that’s not true. Essentially, the PIN we’re looking for is not greater than 10000, but it is more than 100.

Let’s try to narrow it down some more. Try changing the value to 5000. What response do you get?

Response should have been Invalid account number. Which means the PIN we’re searching for is not greater than 5000.

Let’s cut that in half. Change the value to 2500. Hit Go. The response we get back is still Invalid account. So we know it’s less than (or equal to) 2500.

Next try the value of 2200. You should see that it now say Valid Account number again.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.



OWASP Foundation | Project WebGoat | Report Bug

Before looking below at the answer. Go ahead and keep incrementing and decrementing the value until you find out what the PIN is. If you can't figure it out after a few minutes, keep following this lab.

Now change the value to 2250. Hit Go again. You should that we get a valid account number response again. Now try 2260. It should return valid again. Next try 2265. This brings us another valid account response. Try 2275.

Ok. I think you get the point here! Let me help you speed up the process. Just understand that you might have to spend hours doing this in the real world to eventually find what you're looking for. Try the value of 2363. It should return that it's a valid account number.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

Now try 2364

We get an invalid account number response.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Invalid account number.



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

So if the PIN for the credit card number is greater than 2363, but not greater than 2364, then we can assume the actual pin is 2364. Since we know that pins are not typically used in decimal notation, it has to be a whole number that is more than 2363 but not than 2364. There's only one whole number that meets that criteria. 2364! Problem Solved. Blind Injection accomplished.

Go ahead and claim your prize by entering the string 2364 into the form field. See below.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

*** Congratulations. You have successfully completed this lesson.**

Enter your Account Number:

Created by Chuck Willis  **MANDIANT**[®]
INTELLIGENT INFORMATION SECURITY

OWASP Foundation | Project WebGoat | Report Bug