**Week 09**

**Penetration Testing Report**

**Introduction**

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **Week 09 Labs**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

**1. Objective**

The objective of the assessment was to uncover vulnerabilities in the **Week 09 Labs** and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

**2. Scope**

This section defines the scope and boundaries of the project.

| **Application Name** | Lab 1-SQL Injection Labs (SQLI) |
| --- | --- |

**3. Summary**

Outlined is a Black Box Application Security assessment for the **Week 09 Labs**.

**Total number of Sub-labs: 12 Sub-labs**

| **High** | **Medium** | **Low** |
| --- | --- | --- |
| **4** | **4** | **4** |

**High - Number of Sub-labs with hard difficulty level**

**Medium - Number of Sub-labs with Medium difficulty level**

**Low - Number of Sub-labs with Easy difficulty level**

# 1. SQL Injection Labs

# 1.1. Strings & Errors Part 1!

| **Reference** | **Risk Rating** |
| --- | --- |
| Strings & Errors Part 1! | **Low** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Error-based Vulnerability is SQL injection attack, an In-band injection technique where we utilize the error output from the database to manipulate the data inside the database. ... You can force data extraction by using a vulnerability in which the code will output a SQL error rather than the required data from the server. | |
| **How It Was Discovered** | |
| Manual Analysis by breaking out of SQL query statements. Put single-quote(‘), double-quote(“), backtick(`) or semicolon(;)in the identified input fields to interfere with the existing query. | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_1/index.php | |
| **Consequences of not Fixing the Issue** | |
| Error-Based SQL Injection technique forces the database to generate an error, giving the attacker or tester information upon login credentials of users to refine their injection. | |
| **Suggested Countermeasures** | |
| The application might be performing some filtering based on a checklist or whitelist. | |
| **References** | |
| <https://portswigger.net/web-security/sql-injection>  https://www.w3schools.com/sql/sql\_injection.asp  https://owasp.org/www-community/attacks/SQL\_Injection | |

# Proof of Concept

# 

# 

# 1.2. [Strings & Errors Part 2!](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_2/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Strings & Errors Part 2! | **Low** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Error-based Vulnerability is SQL injection attack, an In-band injection technique where we utilize the error output from the database to manipulate the data inside the database. ... You can force data extraction by using a vulnerability in which the code will output a SQL error rather than the required data from the server. | |
| **How It Was Discovered** | |
| Manual Analysis:by breaking out of SQL query statements. Put single-quote(‘), double-quote(“), backtick(`) or semicolon(;)in the identified input fields to interfere with the existing query. | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_2/index.php | |
| **Consequences of not Fixing the Issue** | |
| Error-Based SQL Injection technique forces the database to generate an error, giving the attacker or tester information upon login credentials of all users to refine their injection | |
| **Suggested Countermeasures** | |
| The application might be performing some filtering based on a checklist or whitelist. | |
| **References** | |
| <https://portswigger.net/web-security/sql-injection>  https://owasp.org/www-community/attacks/SQL\_Injection | |

# Proof of Concept

|  | |
| --- | --- |

# 

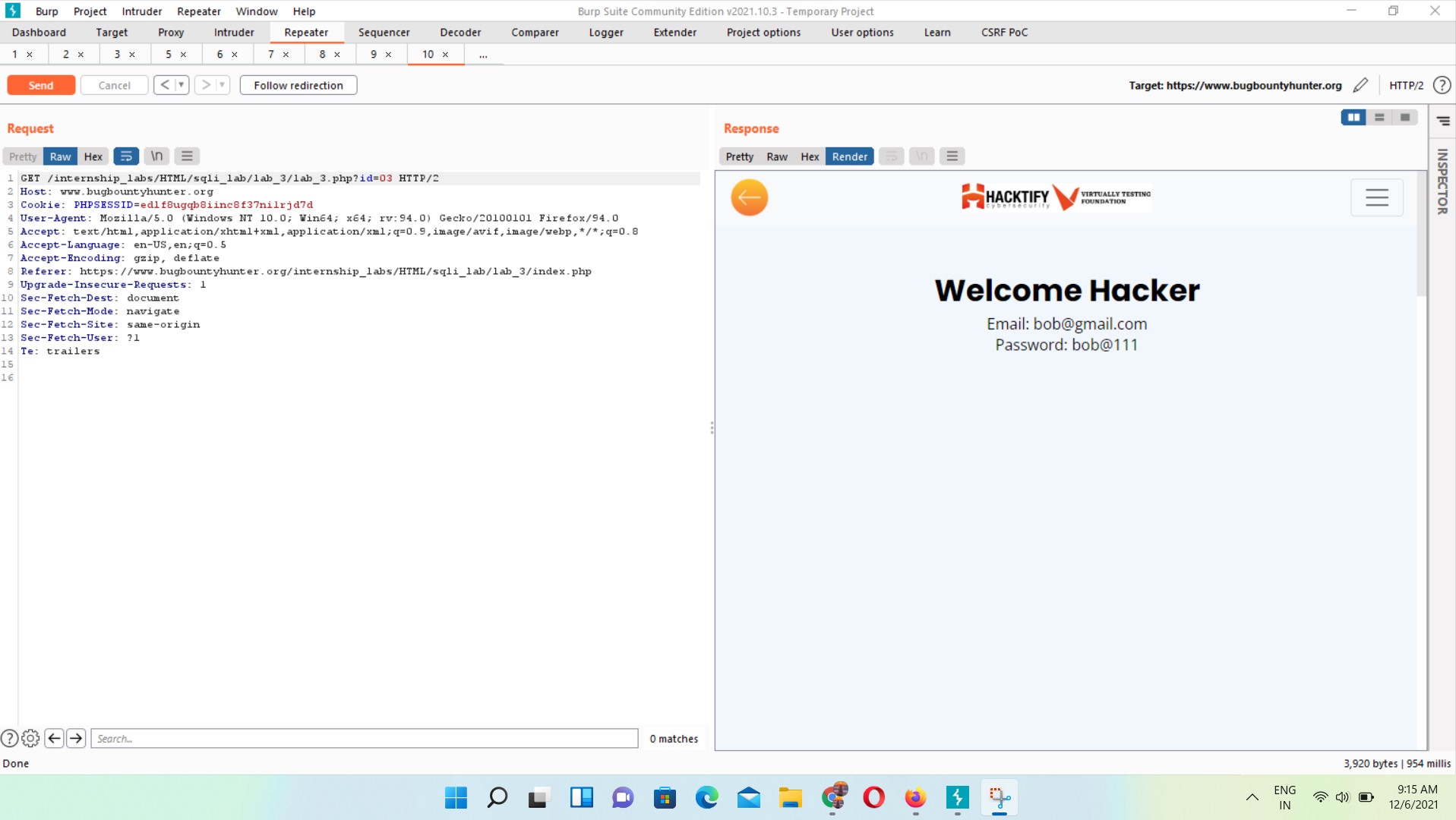
# 

# 

# 1.3. [Strings & Errors Part 3!](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_2/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Strings & Errors Part 3! | **Low** |
| **Tools Used** | |
| web browser,burp suite | |
| **Vulnerability Description** | |
| Error-based Vulnerability is SQL injection attack, an In-band injection technique where we utilize the error output from the database to manipulate the data inside the database. ... You can force data extraction by using a vulnerability in which the code will output a SQL error rather than the required data from the server. | |
| **How It Was Discovered** | |
| Manual Analysis :By breaking out of SQL query statements. Put single-quote(‘), double-quote(“), backtick(`) or semicolon(;)in the identified input fields to interfere with the existing query. | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_3/index.php | |
| **Consequences of not Fixing the Issue** | |
| Error-Based SQL Injection technique forces the database to generate an error, giving the attacker or tester information upon login credentials of all users to refine their injection | |
| **Suggested Countermeasures** | |
| The application might be performing some filtering based on a checklist or whitelist. | |
| **References** | |
| <https://portswigger.net/web-security/sql-injection>  https://owasp.org/www-community/attacks/SQL\_Injection | |

# Proof of Concept



# 

# 

# 

# 

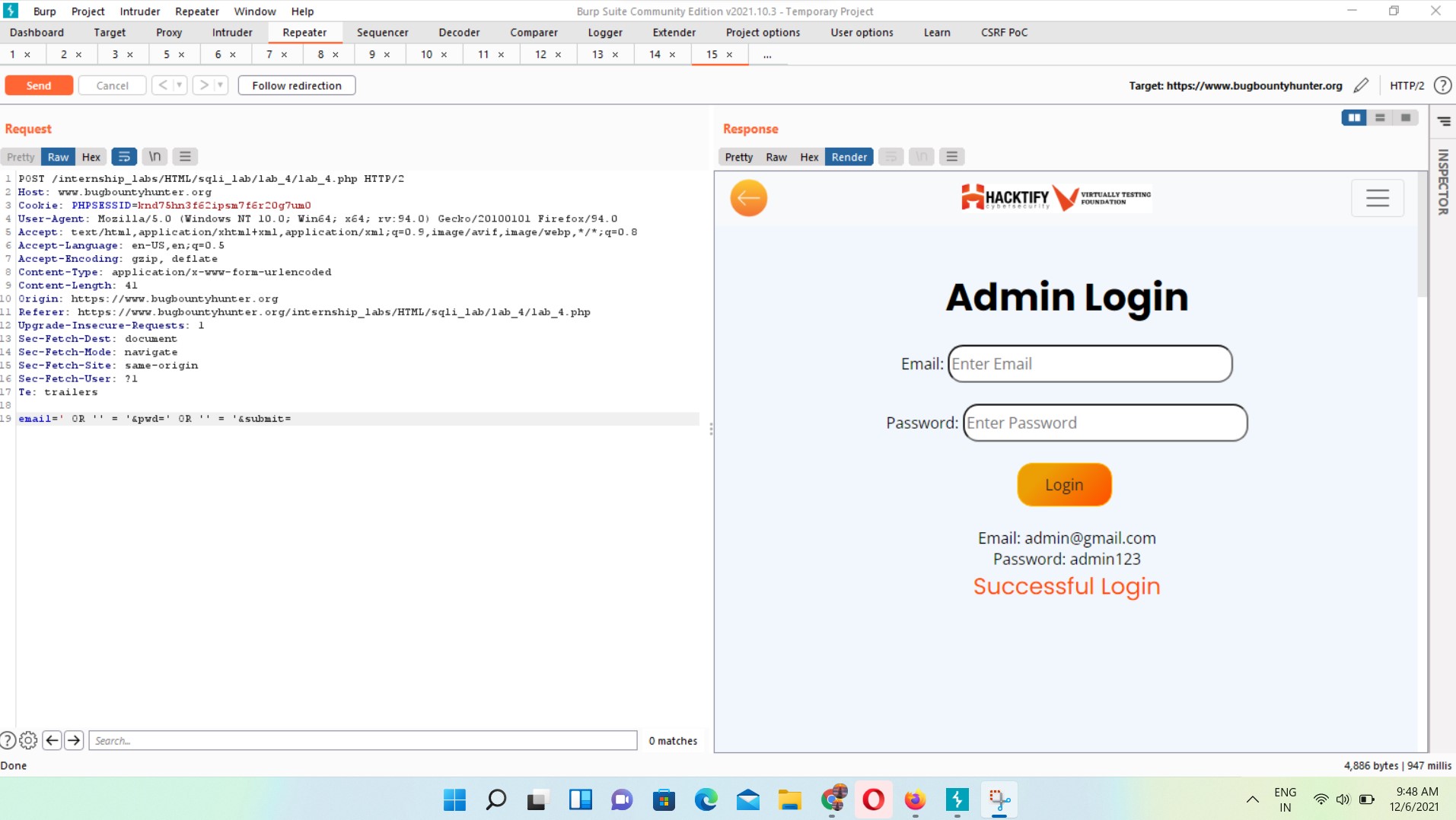
# 

# 

# 1.4. [Let's Trick 'em!](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_4/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Let’s Trick ‘em! | **Medium** |
| **Tools Used** | |
| Web Browser ,Burp Suite | |
| **Vulnerability Description** | |
| This vulnerability is based on SQL injection and allows an attacker to extract information from the database by extending the results returned by the original query. | |
| **How It Was Discovered** | |
| Manual Analysis :using union based sql queries | |
| **Vulnerable URLs** | |
| hhttps://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_4/index.php | |
| **Consequences of not Fixing the Issue** | |
| This technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response. This response may contain data that can be leveraged by the attacker. | |
| **Suggested Countermeasures** | |
| input validation,web application firewall (WAF) | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection> | |

# Proof of Concept



# 

# 

# 

# 

# 

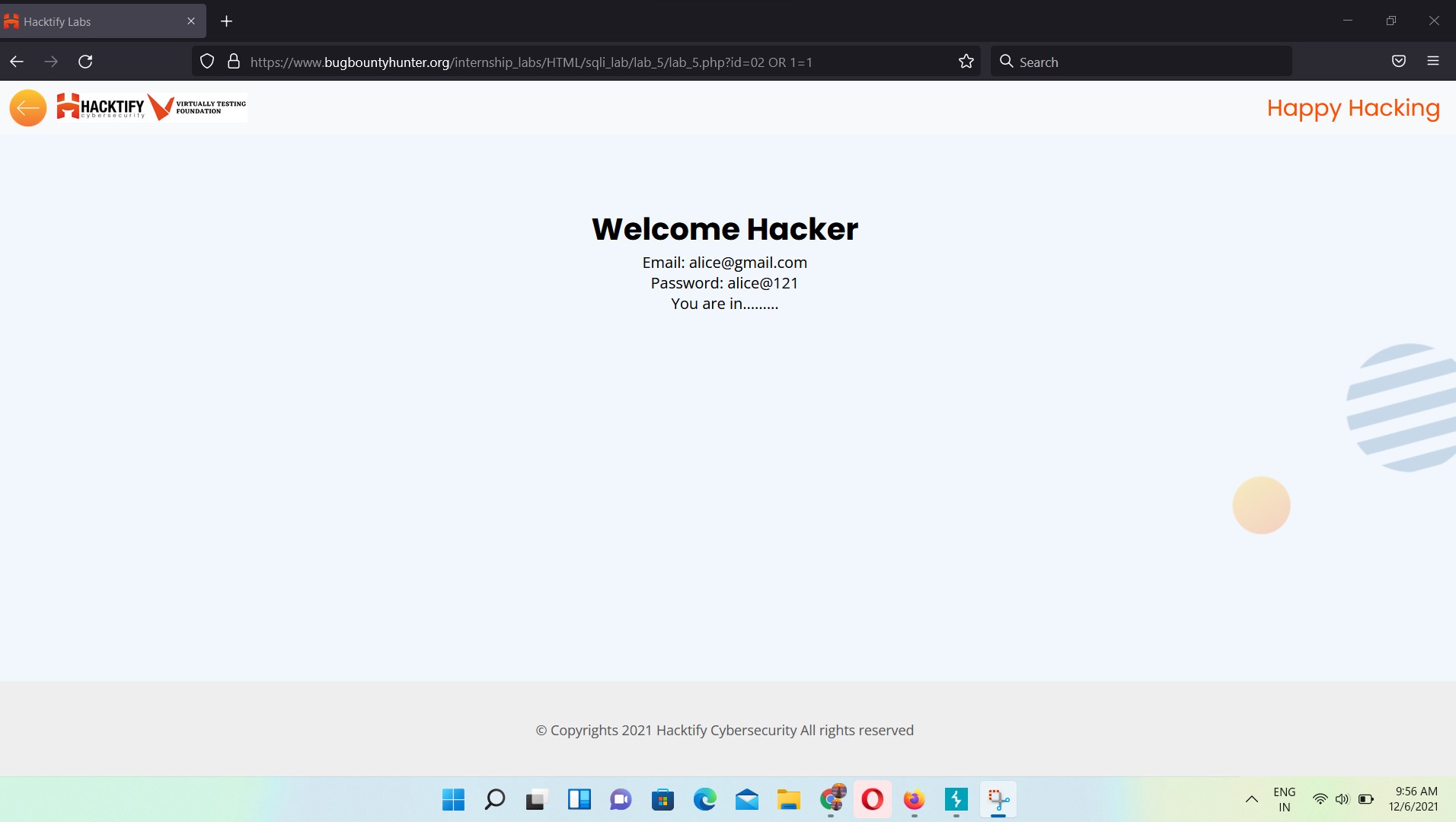
# 

# 

# 1.5. [Booleans and Blind!](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_5/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Booleans and Blind! | **High** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Boolean-based SQL injection is a technique which relies on sending an SQL query to the database. This injection technique forces the application to return a different result, depending on the query. Depending on the boolean result (TRUE or FALSE), the content within the HTTP response will change, or remain the same. The result allows an attacker to judge whether the payload used returns true or false, even though no data from the database are recovered. Also, it is a slow attack; this will help the attacker to enumerate the database. | |
| **How It Was Discovered** | |
| Manual Analysis if an application is vulnerable to SQL injection, it will not return anything, and the attacker will next inject a query with a true condition (1=1). If the content of the page is different than the page that returned during a false condition, then the attacker can infer that SQL injection is working. Now the attacker can verify it he all set to use other SQL Injection method | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_5/index.php | |
| **Consequences of not Fixing the Issue** | |
| it will result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.. | |
| **Suggested Countermeasures** | |
| lind SQL injection attacks can be prevented through the careful use of parameterized queries, which ensure that user input cannot interfere with the structure of the intended SQL query. | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection> | |

# Proof of Concept



**1.6** [**Error Based: Tricked**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_7/index.php)

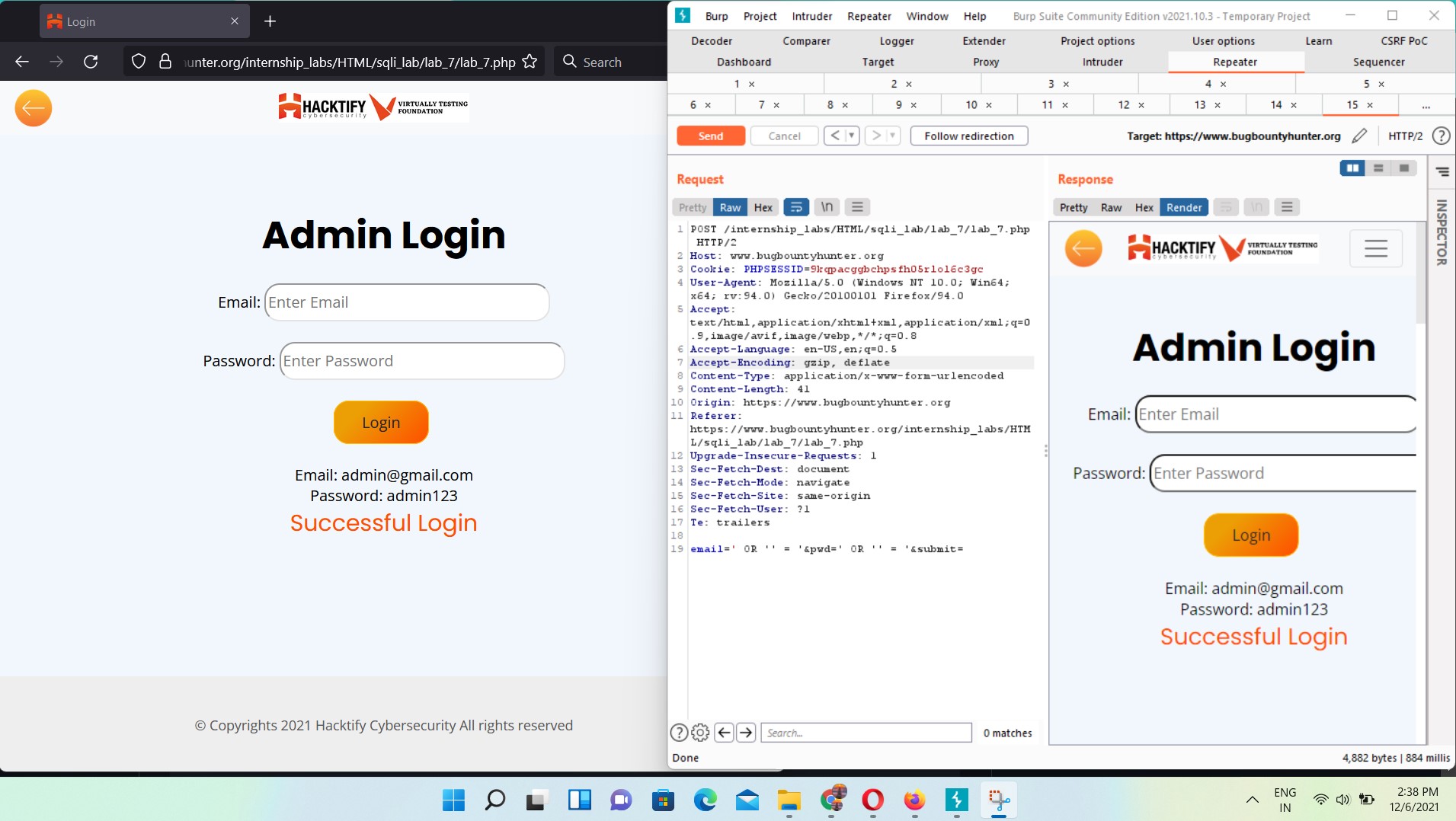
| **Reference** | **Risk Rating** |
| --- | --- |
| Errors Based: Tricked | **Medium** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Error-based SQL injection attack is an In-band injection technique where we utilize the error output from the database to manipulate the data inside the database. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_6/index.php | |
| **Consequences of not Fixing the Issue** | |
| it will result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.Error-Based SQL Injection technique forces the database to generate an error, giving the attacker or tester information upon which to refine their injection | |
| **Suggested Countermeasures** | |
| BySQL injection is usually a URL and its parameters, but here the attacker puts the SQL query hidden in the HTTP header into the field. This technique is commonly used in a variety of scanners, for example, the SqlMap with -p parameters will try the HTTP request header field for injection. | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection> | |

# Proof of Concept

**1.7** [**Errors and Post!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_7/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Error and Post! | **Low** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Error-based SQL injection attack is an In-band injection technique where we utilize the error output from the database to manipulate the data inside the database. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_7/index.php | |
| **Consequences of not Fixing the Issue** | |
| it will result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.Error-Based SQL Injection technique forces the database to generate an error, giving the attacker or tester information upon which to refine their injection | |
| **Suggested Countermeasures** | |
| By performing some filtering based on a checklist or whitelist | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection> | |

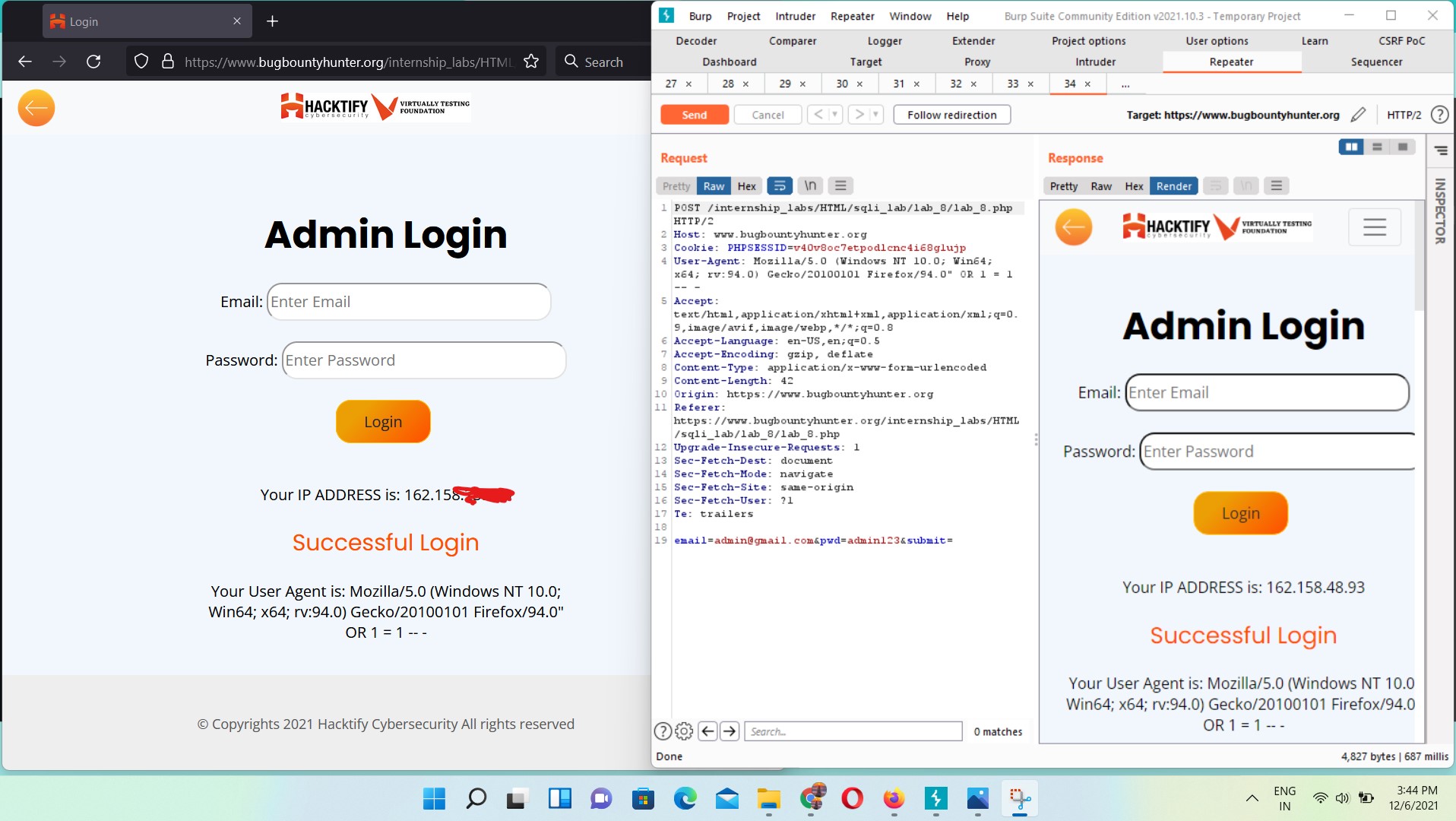
# Proof of Concept



**1.8** [**User Agents lead us!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_8/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| User Agents Lead Us! | **High** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. | |
| **How It Was Discovered** | |
| Manual Analysis ,SQL query hidden in the HTTP header into the field. | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_8/index.php | |
| **Consequences of not Fixing the Issue** | |
| Web application is vulnerable to SQL injection, allowing access to data | |
| **Suggested Countermeasures** | |
| By performing some filtering based on a checklist or whitelist | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection> <https://portswigger.net/web-security/sql-injection> | |

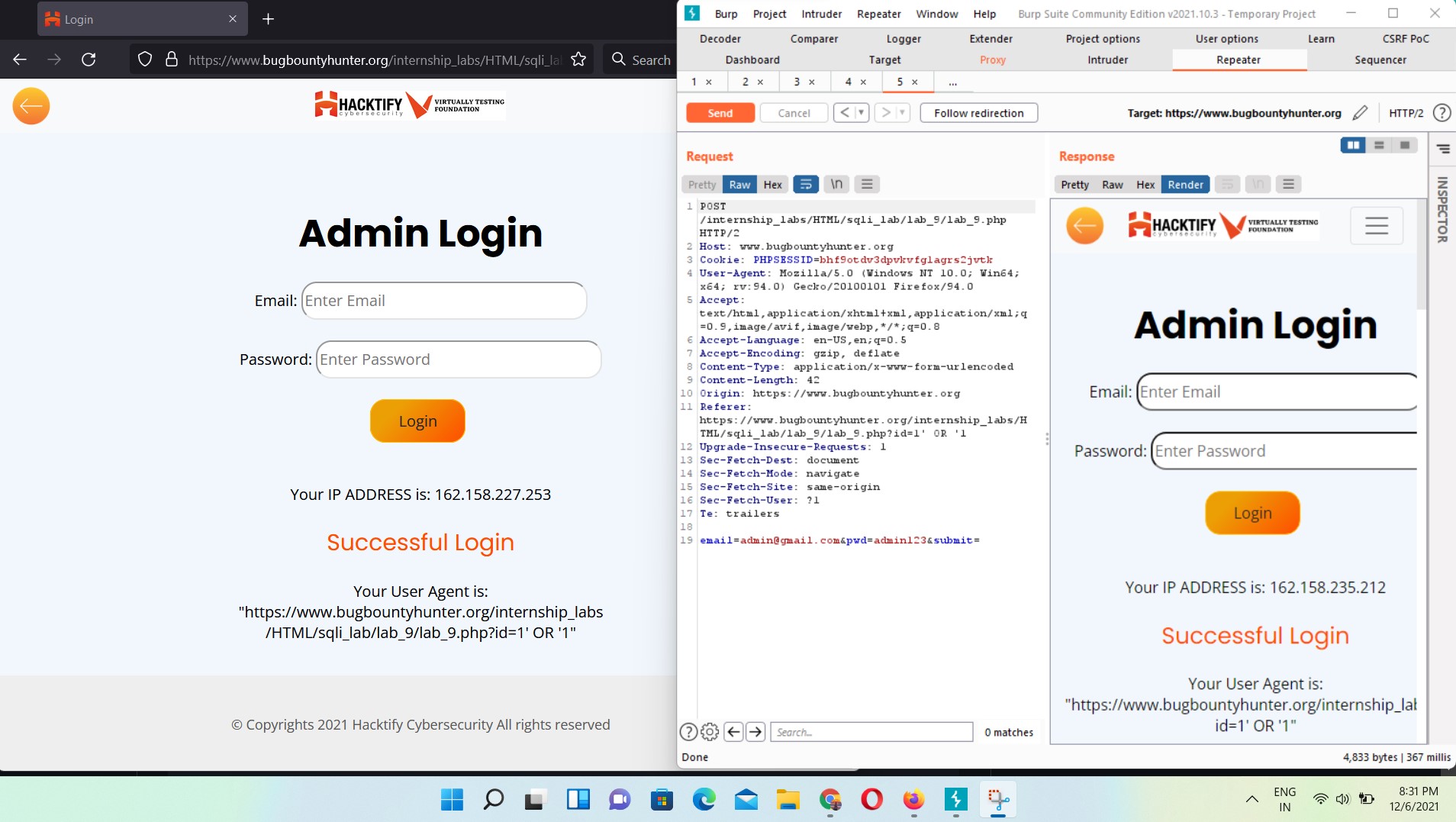
# Proof of Concept



**1.9** [**Referer lead us!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_9/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Referer lead us! | **Medium** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_9/index.php | |
| **Consequences of not Fixing the Issue** | |
| Web application is vulnerable to SQL injection, allowing access to data | |
| **Suggested Countermeasures** | |
| By performing some filtering based on a checklist or whitelist | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection> <https://portswigger.net/web-security/sql-injection> | |

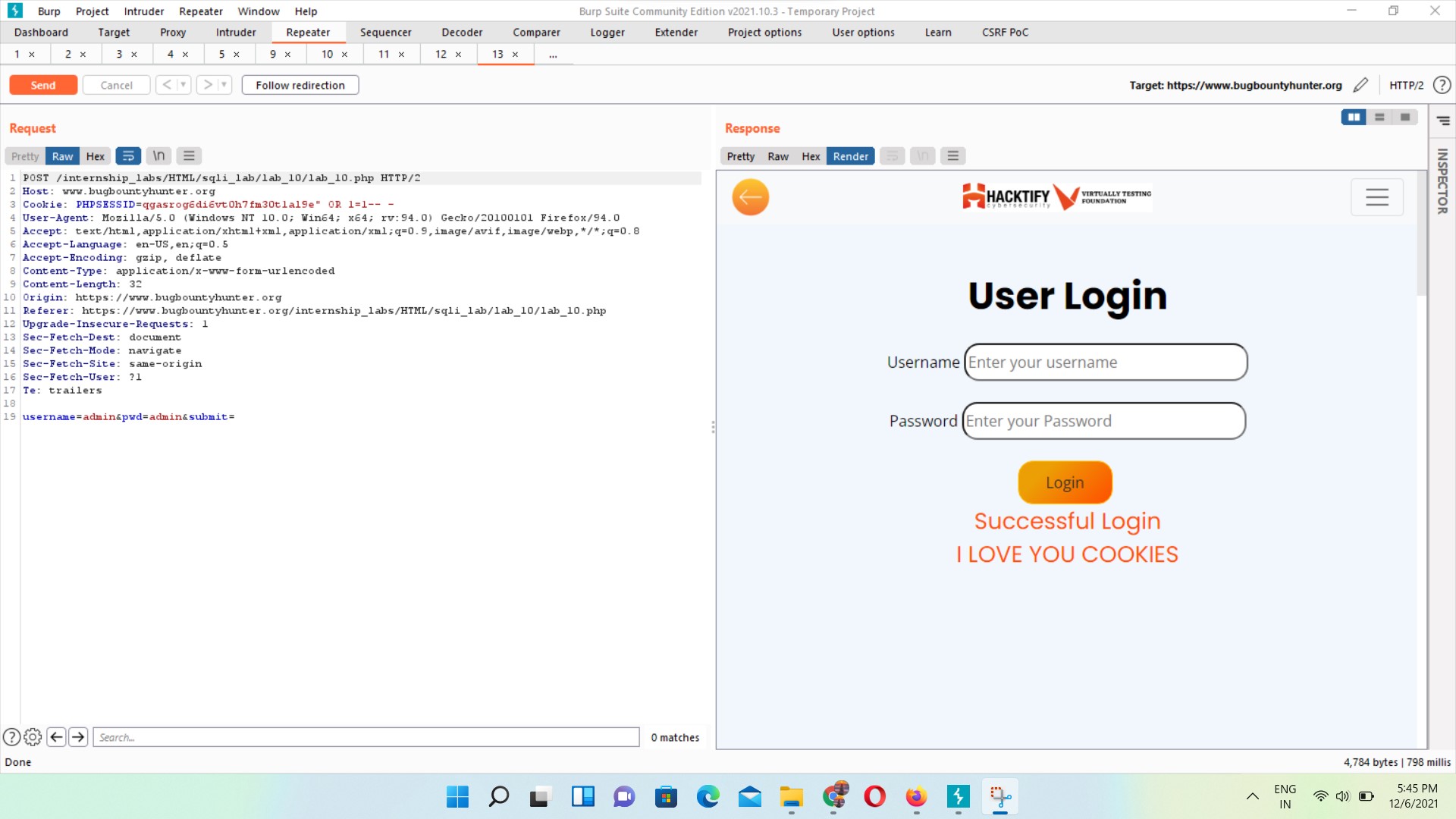
# Proof of Concept



**1.10** [**Oh Cookies!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_10/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| Oh Cookies! | **High** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| Cookie Poisoning attacks involve the modification of the contents of a cookie (personal information stored in a Web user’s computer) in order to bypass security mechanisms. Using cookie poisoning attacks, attackers can gain unauthorized information about another user and steal their identity. | |
| **How It Was Discovered** | |
| Manual Analysis ,It’s like a get/post based SQL Injection, except that certain characters can’t be used. For example, ‘;‘ and ‘,‘ are typically treated as delimiters, so they end the injection if they aren’t URL-encoded.malicious SQL statements implemented in the cookie field. | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_10/index.php | |
| **Consequences of not Fixing the Issue** | |
| Web application is vulnerable to SQL injection, allowing access to data | |
| **Suggested Countermeasures** | |
| Cookie variables should be properly sanitized | |
| **References** | |
| <https://resources.infosecinstitute.com/topic/cookie-based-sql-injection/>  <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection> | |

# Proof of Concept



**1.11** [**WAF's are injected!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_11/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| WAF’s are injected! | **High** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_11/index.php | |
| **Consequences of not Fixing the Issue** | |
| Web application is vulnerable to SQL injection, allowing access to data | |
| **Suggested Countermeasures** | |
| By performing some filtering based on a checklist or whitelist | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection>  https://owasp.org/www-community/attacks/SQL\_Injection\_Bypassing\_WAF | |

# Proof of Concept

**1.12** [**WAF's are injected Part 2!**](https://www.bugbountyhunter.org/internship_labs/HTML/sqli_lab/lab_11/index.php)

| **Reference** | **Risk Rating** |
| --- | --- |
| WAF’s are Injected Part 2! | **Medium** |
| **Tools Used** | |
| Web Browser,Burp Suite | |
| **Vulnerability Description** | |
| SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/sqli\_lab/lab\_12/index.php | |
| **Consequences of not Fixing the Issue** | |
| Web application is vulnerable to SQL injection, allowing access to data | |
| **Suggested Countermeasures** | |
| By performing some filtering based on a checklist or whitelist | |
| **References** | |
| <https://owasp.org/www-community/attacks/SQL_Injection>  <https://portswigger.net/web-security/sql-injection>  https://owasp.org/www-community/attacks/SQL\_Injection\_Bypassing\_WAF | |

# Proof of Concept