Bytium®

Web Application Security Report

Reporting Date: Feb 8, 2025

Client Name: CLIENT_NAME
Testing Methodology: Black Box
Report Version: Final
Tested By: Bytium® Team

Confidentiality Notice

This report contains sensitive, privileged, and confidential information. Precautions should be taken to protect the confidentiality of the information in this document. Publication of this report may cause reputational damage to Vendor_name or facilitate attacks against the Smart School Management Application. Bytium Team shall not be held liable for special, incidental, collateral, or consequential damages arising out of the use of this information.

Disclaimer

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope of the engagement. This report is a summary of the findings from a “point-in-time” assessment made on Application_Name’s environment. Any changes made to the environment during the period of testing may affect the results of the assessment.

TABLE OF CONTENTS

Executive Summary

This report summarizes the security audit performed for Application_Name for 1 day. Our audit identified vulnerabilities ranging from low to high risk, with certain critical vulnerabilities requiring immediate attention for remediation.

Addressing these vulnerabilities will significantly decrease potential risks such as exploit use, regulatory penalties, service disruption, integrity breaches, system compromise, and data theft.

This report provides a detailed account of identified vulnerabilities, risk ratings, and suggested remediation. We recommend addressing these vulnerabilities systematically, prioritizing those posing the highest risk.

Regular audits and consistent security measures are crucial for maintaining and improving the security environment, given the constant evolution of technology and threats. We recommend ongoing security assessments to ensure robust defense mechanisms.

Objectives

It is expected that addressing vulnerabilities and diminishing risks within the system will significantly decrease the probability of the following scenarios:

- Usage of publicly available exploits due to insufficient patching.
- Monetary damages due to regulatory sanctions.
- Disruption of service availability owing to inadequate rate-limiting methods.
- Integrity breaches due to weak authorization verification.
- System compromise, data modification, or data destruction attacks.
- Data theft owing to weak or absent cryptographic measures.
- Damage to reputation due to exploitation of any of the aforementioned vulnerabilities.

Scope of Testing

The security assessment involves conducting tests for security gaps within the defined parameters detailed below. Beyond the given parameters, no additional information was furnished, and no assumptions were made at the commencement of the security assessment. The subsequent list delineates the specific areas covered in the scope of this security audit.

URL(testing purpose only)	IP Address	Note
company.com/webtest/	**ip_address**	Fresh install

Assessment Details

Stored XSS

Severity	High
Affected Application	https://company.com/webtest/user/apply_leave
Status	Open

Description

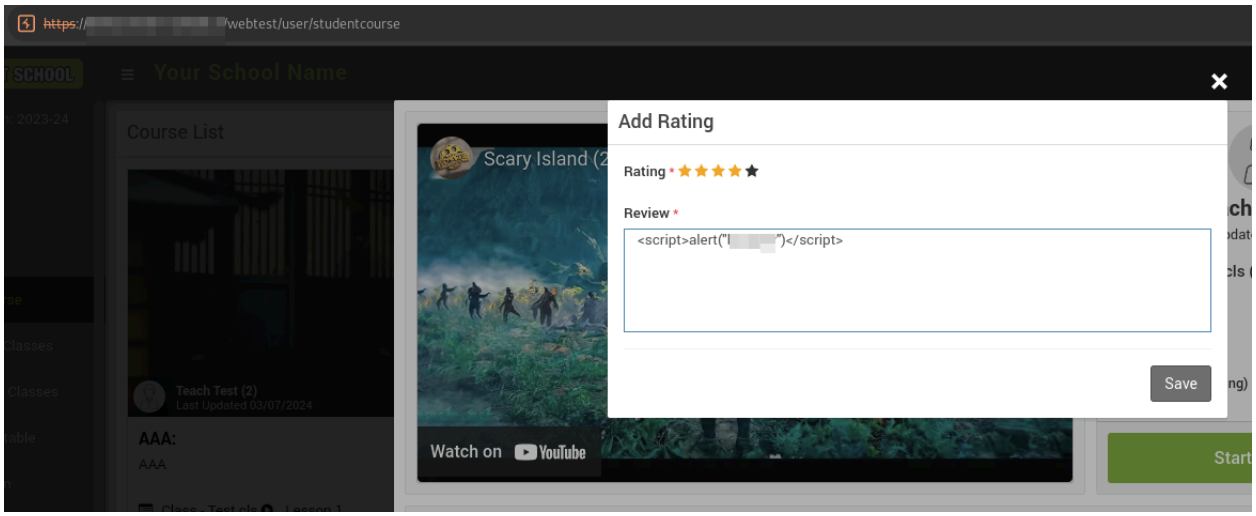
This is a high-risk client-side vulnerability found in sever. **test.com** is the main ip of test.com. When browsing **/wiki/api.php/** it takes value with parameter **format, it** takes unsanitized inputs. That leads to XSS.

Impact

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can Perform any action within the application that the user can perform.

Steps to Reproduce

1. Navigate to the affected URL
2. Fill the “Reason” Field with “<script>alert(1)</script>”



3. Save, and it will immediately pop up the payload

Recommendation

1. Any user input should be validated. If validation fails, the request should be rejected.
2. All HTML metacharacters such as <>= should be replaced by HTML entities such as (< >).
3. Additionally, a Web Application Firewall can be used.

Stored XSS

Severity	High
Affected Application	https://company.com/webtest/user/studentcourse
Status	Open

Description

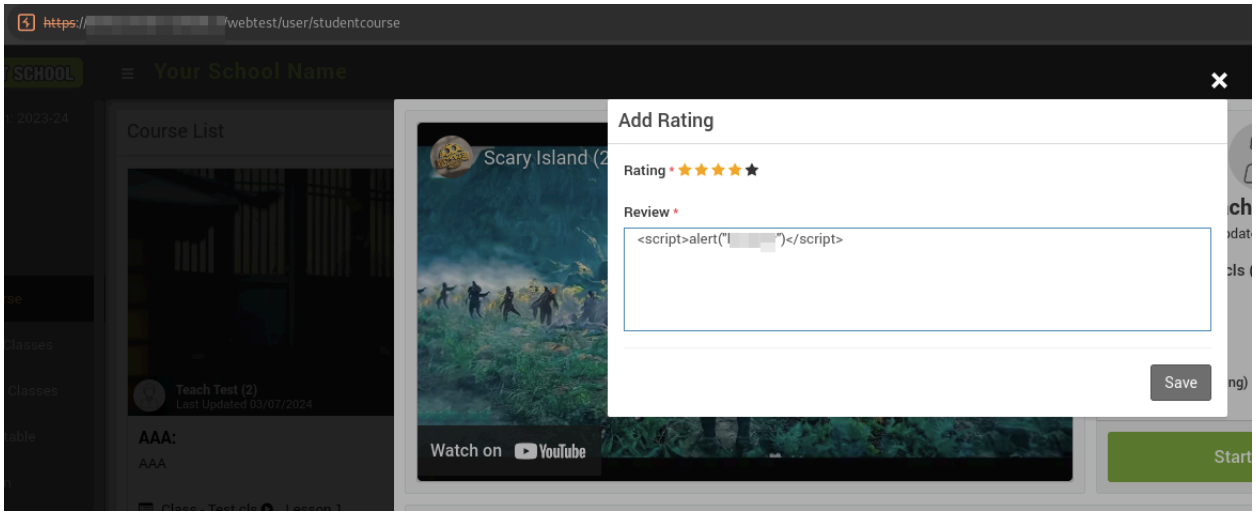
This is a high-risk client-side vulnerability found in sever. **test.com** is the main ip of test.com. When browsing **/wiki/api.php/** it takes value with parameter **format, it** takes unsanitized inputs. That leads to XSS.

Impact

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can Perform any action within the application that the user can perform.

Steps to Reproduce

1. Navigate to the affected URL
2. Click to rate, and Fill the “Review” Field with “<script>alert(1)</script>”



3. Save, and it will immediately pop up the payload

Recommendation

1. Any user input should be validated. If validation fails, the request should be rejected.
2. All HTML metacharacters such as <>= should be replaced by HTML entities such as (< >).
3. Additionally, a Web Application Firewall can be used.

Stored XSS

Severity	High
Affected Application	https://company.com/webtest/user/homework/dailyassignment
Status	Open

Description

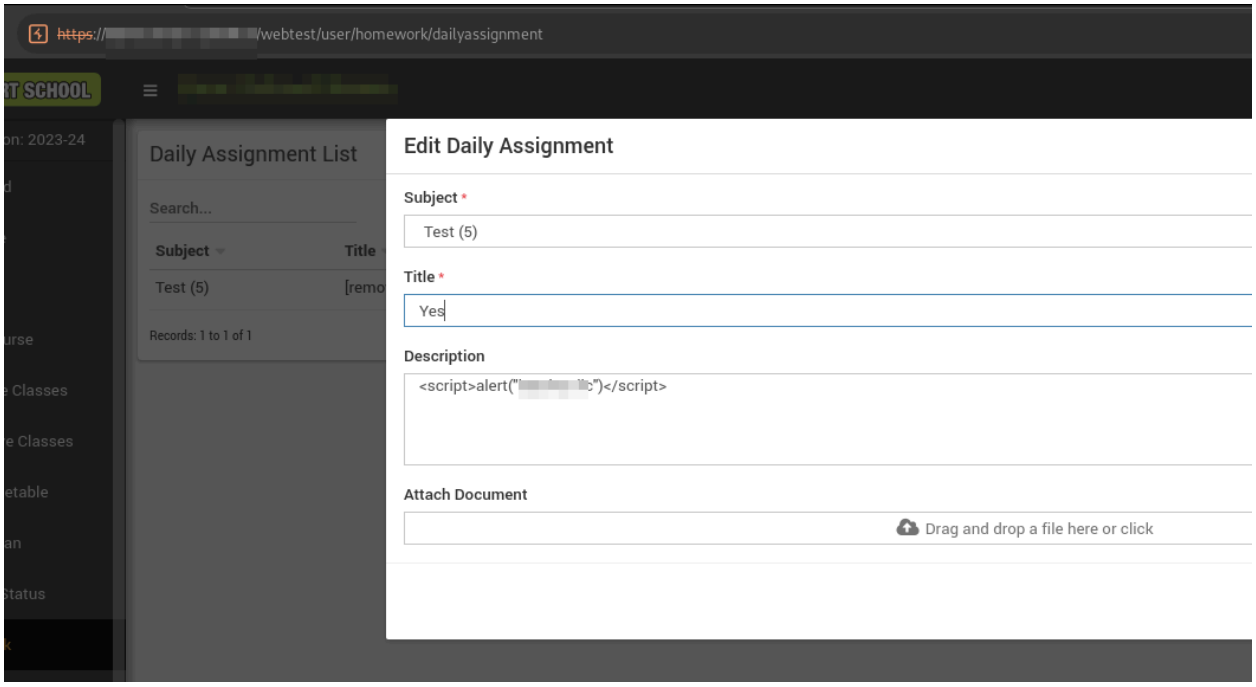
This is a high-risk client-side vulnerability found in sever. **test.com** is the main ip of test.com. When browsing **/wiki/api.php/** it takes value with parameter **format, it** takes unsanitized inputs. That leads to XSS.

Impact

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can Perform any action within the application that the user can perform.

Steps to Reproduce

1. Navigate to the affected URL
2. Click on the Daily Assignment button, and Fill the “Description” Field with “<script>alert(“byttium llc”)</script>”



3. Save, and it will immediately pop up the payload

Recommendation

1. Any user input should be validated. If validation fails, the request should be rejected.
2. All HTML metacharacters such as <>= should be replaced by HTML entities such as (< >).
3. Additionally, a Web Application Firewall can be used.

Blind SQL Injection

Severity	Critical
Affected Application	https://company.com/webtest/user/chat/mynewuser
Status	Open

Description

The parameter “users[]” is vulnerable to blind SQL injection. SQL injection (SQLi) is a vulnerability that lets attackers run harmful SQL code on a web application's database, potentially accessing or altering data unauthorizedly.

Impact

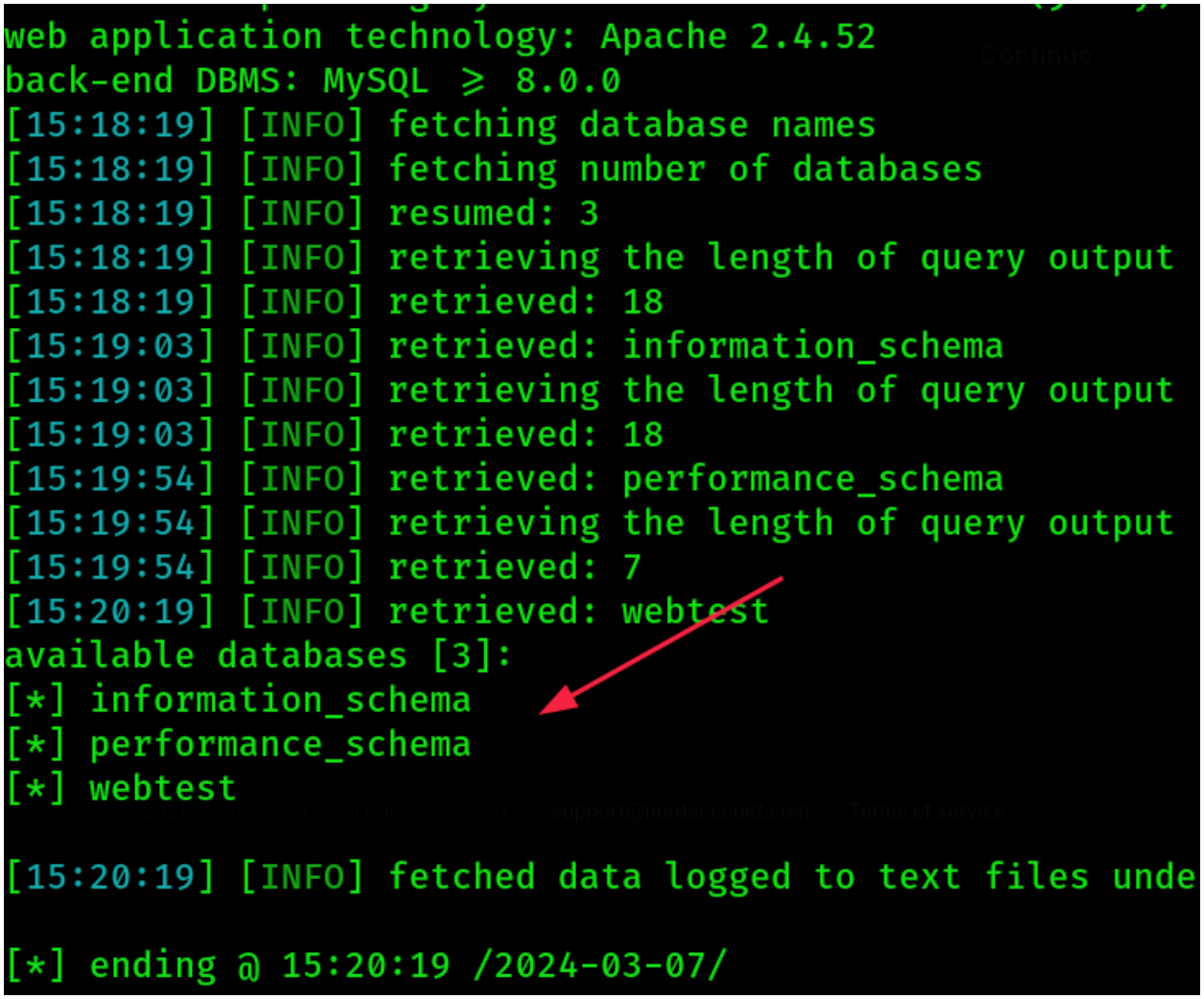
SQL injection allows attackers to circumvent a web application's security checks, accessing or altering the database's contents. This vulnerability can lead to unauthorized data retrieval, modification, or deletion, compromising data integrity. In certain scenarios, SQLi might enable attackers to run operating system commands, potentially escalating the attack further.

Steps to Reproduce

1. Login as a student. Intercept with burp suite to get the authenticated session.
2. Install sqlmap, copy this command:

```
sqlmap -u "https://company.com/webtest/user/chat/mynewuser" --data="users[]=10"
--cookie="sitecookies=1; ci_session=q6eos1m1b6ss4h7p8imkqh940ndjimt2"
--referer="https://demo.smart-school.in/webtest/" --user-agent="Mozilla/5.0 (Windows NT 6.1;
WOW64; Trident/7.0; rv:11.0) like Gecko" --headers="X-Requested-With: XMLHttpRequest"
--headers="Accept: application/json, text/javascript, */*; q=0.01" --headers="Content-Type:
application/x-www-form-urlencoded" --headers="Connection: Keep-alive" --risk=3 --level=5 -p
"users[]" --dbms mysql --technique B
```

3. Make sure to change the cookie value of the above command. And run the sqlmap command.



Recommendation

1. **Input Validation:** Ensure that all user inputs are strictly validated. If the validation fails, the request should be immediately rejected.
2. **Block Special Characters:** Single quotation marks, which are commonly used in SQL queries, should be blocked or properly escaped to prevent malicious queries.
3. **Parameterized Queries:** Use parameterized queries or prepared statements to ensure that user input is always treated as data and not executable code.