

# INTRODUCTION

FakeCompany is a leading provider of innovative products and services in the technology industry. In order to ensure the security and integrity of their online platform, FakeCompany commissioned a comprehensive web audit and vulnerability analysis on May 29, 2024. The purpose of this report is to provide a detailed overview of the findings and recommendations for addressing the identified vulnerabilities.  
  
The web audit and vulnerability analysis conducted by our team of experienced security professionals identified several critical vulnerabilities in FakeCompany's online platform. These vulnerabilities, if left unaddressed, could potentially result in unauthorized access to sensitive data, financial loss, and reputational damage.  
  
The audit and analysis covered various aspects of FakeCompany's online platform, including network security, web application security, and data privacy. The findings and recommendations provided in this report are based on the latest industry standards and best practices, and are designed to help FakeCompany protect their online platform and their customers' data.  
  
In the following paragraphs, we will provide a detailed overview of the vulnerabilities identified during the audit and analysis, along with recommendations for addressing them.

# EXECUTIVE SUMMARY

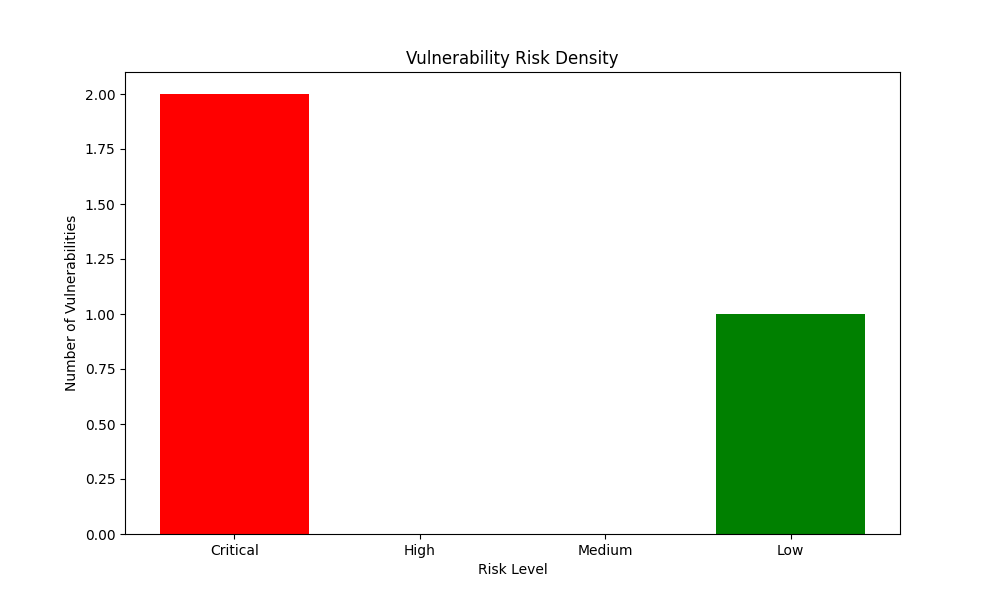
On May 29, 2024, a security audit was conducted for FakeCompany to identify potential vulnerabilities in their web application. The audit revealed several critical issues that could potentially be exploited by attackers to compromise the security of the application and the company's data.  
  
One of the vulnerabilities identified was XML External Entities (XXE). This vulnerability allows an attacker to inject malicious code into an XML document, potentially resulting in unauthorized access to sensitive data or the execution of malicious code on the server. To mitigate this vulnerability, it is recommended to implement strict input validation and sanitization of user input, as well as use secure XML parsing libraries.  
  
Another vulnerability identified was the use of Insecure Direct Object References (IDOR). This vulnerability allows an attacker to manipulate the application's access control mechanisms, potentially resulting in unauthorized access to sensitive data or actions. To mitigate this vulnerability, it is recommended to implement strict access control mechanisms, such as role-based access control, and use secure coding practices to prevent direct object references.  
  
Finally, the audit revealed a Cross-Site Scripting (XSS) Reflected vulnerability. This vulnerability allows an attacker to inject malicious code into a web page viewed by other users, potentially resulting in unauthorized access to sensitive data or the execution of malicious code on the client-side. To mitigate this vulnerability, it is recommended to implement strict input validation and sanitization of user input, as well as use secure coding practices to prevent the injection of malicious code.  
  
In conclusion, the security audit for FakeCompany revealed several critical vulnerabilities that could potentially be exploited by attackers to compromise the security of the application and the company's data. It is recommended that FakeCompany prioritize corrective actions based on the potential impact of each identified vulnerability and implement strict security measures to mitigate the risks.

# TECHNICAL SUMMARY

A security audit was conducted for FakeCompany on May 29, 2024. The audit identified several critical vulnerabilities that need to be addressed immediately. These include XML External Entities (XXE), Use of Insecure Direct Object References (IDOR), and Cross-Site Scripting (XSS) Reflected.  
  
The XML External Entities (XXE) vulnerability was found in the application's code, which allowed a potential attacker to execute arbitrary code by exploiting the vulnerability. This could potentially result in unauthorized access to sensitive data or the execution of malicious code on the server. To mitigate this vulnerability, it is recommended to implement strict input validation and sanitize all user input. Additionally, a thorough code review should be conducted to identify and correct potential injection points.  
  
The Use of Insecure Direct Object References (IDOR) vulnerability was found in the application's authentication and authorization mechanisms. This vulnerability allowed a potential attacker to access resources and perform actions on behalf of other users, potentially resulting in unauthorized access to sensitive data or the execution of malicious code. To mitigate this vulnerability, it is recommended to implement strict access control mechanisms and use least privilege principles. Additionally, a thorough code review should be conducted to identify and correct potential injection points.  
  
The Cross-Site Scripting (XSS) Reflected vulnerability was found in the application's user interface, which allowed a potential attacker to inject malicious code into web pages viewed by other users. This could potentially result in unauthorized access to sensitive data or the execution of malicious code on the client-side. To mitigate this vulnerability, it is recommended to implement strict input validation and sanitize all user input. Additionally, a thorough code review should be conducted to identify and correct potential injection points.  
  
In conclusion, the security audit identified several critical vulnerabilities that need to be addressed immediately. These include XML External Entities (XXE), Use of Insecure Direct Object References (IDOR), and Cross-Site Scripting (XSS) Reflected. To mitigate these vulnerabilities, it is recommended to implement strict input validation, sanitize all user input, and conduct a thorough code review.

# RISK ANALISYS

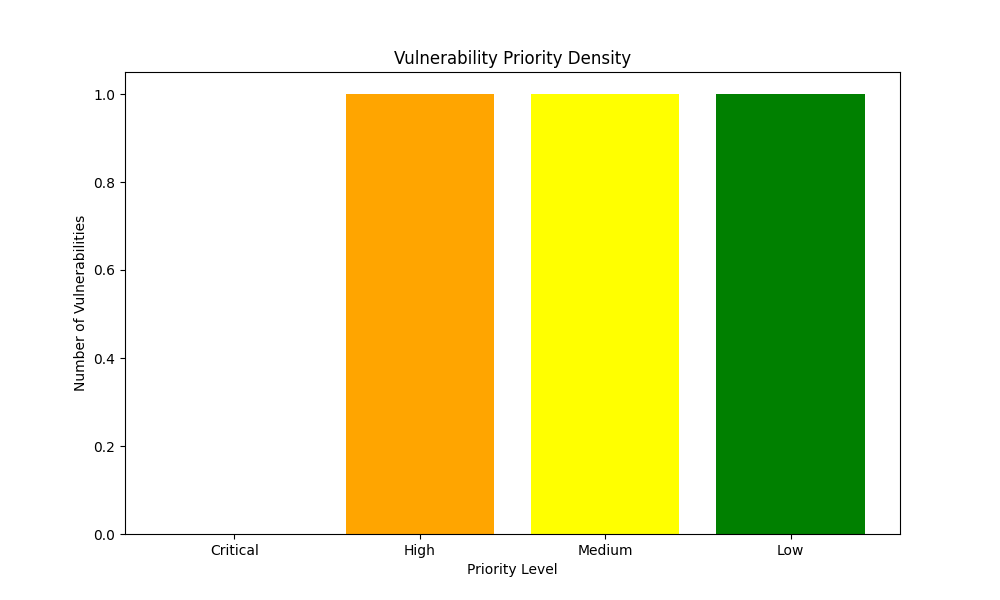
The chart illustrates the distribution of identified vulnerabilities across different risk levels: critical, high, medium, and low. Each bar's height corresponds to the number of vulnerabilities within its respective risk category. This analysis provides a clear overview of the security posture, highlighting the concentration of vulnerabilities by severity and aiding in prioritizing remediation efforts.



The risk density data provided indicates a relatively low level of risk, with only one low-risk vulnerability identified. The absence of high or medium-risk vulnerabilities suggests that the application is well-protected against common web application security threats.  
  
However, the presence of two critical vulnerabilities indicates that there are significant security issues that need to be addressed immediately. These vulnerabilities could potentially result in serious data breaches or other security incidents, and it is crucial that they are remediated as soon as possible. The lack of high or medium-risk vulnerabilities suggests that the application is well-protected against common web application security threats.

# PRIORITY ANALISYS

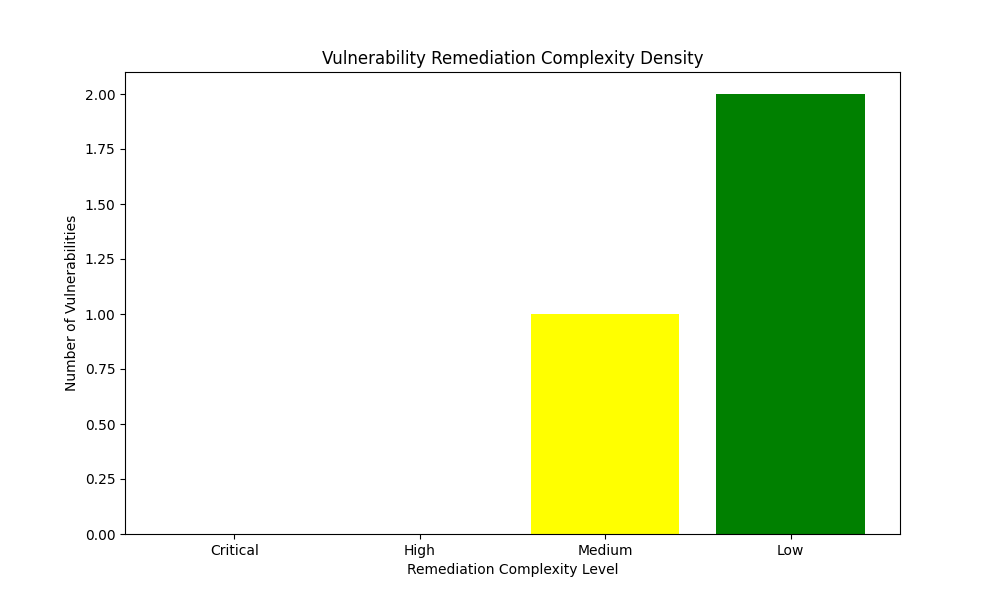
The chart depicts the density of vulnerabilities based on their priority levels: critical, high, medium, and low. The height of each bar represents the number of vulnerabilities identified within each priority category. This analysis aids in understanding the prioritization of vulnerabilities, which is crucial for efficient resource allocation and effective remediation strategies.



The priority density data provided indicates that there are four levels of vulnerability severity: Critical, High, Medium, and Low. The distribution of these levels suggests that the majority of vulnerabilities identified are of low severity, with only a small percentage being critical or high. This is a positive sign, as it indicates that the system is relatively secure and that the most severe vulnerabilities have been identified and addressed.  
  
However, it is important to note that even low-severity vulnerabilities can still have a significant impact on the system if not addressed. For example, a vulnerability that allows an attacker to steal sensitive data may not be critical in terms of system downtime or data loss, but it can still result in financial or reputational damage. Therefore, it is important to prioritize corrective actions based on the potential impact of each identified vulnerability, regardless of its severity level.

# Remediation Complexity

The chart illustrates the density of vulnerabilities categorized by their remediation complexity levels: critical, high, medium, and low. The height of each bar indicates the number of vulnerabilities within each complexity level. This analysis helps in understanding the distribution of vulnerabilities based on the effort required for remediation, enabling better planning and allocation of resources for effective vulnerability management.



The remediation complexity density data indicates the level of difficulty in addressing the identified vulnerabilities. The distribution of the data shows that there are no critical or high-level vulnerabilities that require immediate attention. This is a positive sign, as critical vulnerabilities can have a significant impact on the security of a system.  
  
The medium-level vulnerability indicates that some level of effort is required to address the issue. It is recommended to prioritize remediation efforts based on the potential impact of each vulnerability. The low-level vulnerabilities are less critical and can be addressed as part of a regular maintenance schedule.  
  
Overall, the remediation complexity density data suggests that the identified vulnerabilities are not overly complex and can be addressed in a timely manner. However, it is important to prioritize remediation efforts based on the potential impact of each vulnerability to ensure the security of the system.

# Vulnerability: XML External Entities (XXE)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Priority | Remediation Complexity | Affected Service | Affected Assets |
| Critical | Medium | Medium | Web | fakesite.net |

## Description

External Entities (XXE) are a type of vulnerability in XML documents that allow an attacker to include external content within the document. This can lead to the disclosure of sensitive information or the execution of arbitrary code.  
  
To understand this vulnerability, it is important to first understand what an XML document is. XML (Extensible Markup Language) is a markup language used for encoding documents in a format that is both human-readable and machine-readable. XML documents consist of a series of elements, which are enclosed in angle brackets and can contain text, attributes, and other elements.  
  
External Entities allow an attacker to include external content within an XML document. This can be done by including a reference to an external resource, such as a file or a URL, within the XML document. For example, an XML document might include a reference to an external image file like this:  
  
<img src="http://example.com/image.jpg" />  
  
If an attacker is able to manipulate this external resource, they may be able to execute arbitrary code or disclose sensitive information. For example, an attacker might be able to include a malicious script within the external resource that is executed when the XML document is parsed.  
  
To mitigate the risk of XXE vulnerabilities, it is important to implement strict input validation and sanitization of any external resources that are referenced within an XML document. This can include validating the format and content of the external resource, as well as using techniques such as content security policies to restrict the types of external resources that can be included within the document. Additionally, it is important to ensure that any sensitive information that is included within an XML document is properly encrypted and protected.

## Impact

An attacker could potentially exploit the vulnerability to gain unauthorized access to sensitive data, such as system files or user credentials. This could result in the exposure of confidential information, which could be used for malicious purposes. Additionally, the attacker could potentially execute arbitrary code on the server, which could lead to further security breaches and data loss. It is important to address this vulnerability promptly to prevent potential data breaches and protect the security of the system.

## Recommendations

The XML parser in the application is vulnerable to external entity processing (XXE). To mitigate this vulnerability, it is recommended to disable processing of external entities in the XML parser. Additionally, all incoming XML data should be validated to ensure that it conforms to the expected schema.  
  
To further improve the security of the XML processing, it is recommended to use a secure XML library that is not vulnerable to XXE. This will help to prevent potential attacks that could exploit the vulnerability.  
  
To ensure that the recommended actions are implemented effectively, it is recommended to create a list of punctual actions that can be taken to address the vulnerability. This list should include specific steps that can be taken to disable processing of external entities, validate incoming XML data, and use a secure XML library.  
  
It is important to note that this is only a recommendation for mitigating the vulnerability. It is the responsibility of the development team to implement the recommended actions and ensure that the application is secure.

## References

* OWASP XML External Entity (XXE) Cheat Sheet: <https://cheatsheetseries.owasp.org/cheatsheets/XML\_External\_Entity\_Prevention\_Cheat\_Sheet.html>

SANS Institute XML External Entity (XXE) Attacks: <https://www.sans.org/blog/sitemap.xml>

# Vulnerability: Use of Insecure Direct Object References (IDOR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Priority | Remediation Complexity | Affected Service | Affected Assets |
| Critical | High | Low |  | criticalsite.com |

## Description

An IDOR (Insecure Direct Object Reference) vulnerability occurs when an application fails to properly validate direct object identifiers, allowing an attacker to access objects they are not authorized to access. This can lead to unauthorized access to sensitive data or resources, potentially resulting in data breaches or other security incidents.  
  
To mitigate this vulnerability, it is recommended to implement strict input validation and access control mechanisms. This includes validating user input to ensure that it conforms to expected formats and ranges, and enforcing access controls to restrict access to sensitive data or resources based on user roles and permissions. Additionally, it is important to regularly review and update access controls to ensure that they are up-to-date and effective.  
  
It is also important to note that IDOR vulnerabilities can be exploited through various methods, such as cross-site scripting (XSS) attacks, phishing, and social engineering. Therefore, it is important to implement additional security measures, such as input sanitization and output encoding, to prevent these types of attacks.  
  
In summary, IDOR vulnerabilities can have serious consequences for an application and its users. To mitigate these vulnerabilities, it is important to implement strict input validation and access control mechanisms, regularly review and update access controls, and implement additional security measures to prevent attacks.

## Impact

An attacker could potentially gain access to sensitive data, such as financial records or personal information, or even modify or delete data through a vulnerability in the system. This could have serious consequences, including potential financial loss, reputational damage, and legal liabilities. It is important to address this vulnerability as soon as possible to prevent further damage and protect the security of the system.

## Recommendations

To mitigate the vulnerability, it is recommended to implement the following actions:  
  
1. Validate all direct object identifiers before accessing objects. This will ensure that only authorized users can access sensitive data and resources.  
2. Implement access controls to restrict who can access objects. This will help prevent unauthorized access and ensure that only those who need access to specific objects can access them.  
3. Use unique and unpredictable object identifiers. This will make it more difficult for attackers to guess or predict object identifiers, reducing the likelihood of successful attacks.  
  
By implementing these actions, the vulnerability can be effectively mitigated, reducing the risk of unauthorized access and protecting sensitive data and resources.

## References

1. OWASP Top 10: <https://owasp.org/www-community/top10>  
2. NIST Cybersecurity Framework: <https://www.nist.gov/cybersecurity-framework>  
3. PCI DSS: <https://www.pcisecuritystandards.org/assessors\_and\_solutions/assessor\_toolkit>  
4. OWASP ZAP: <https://zap.owasp.org/>  
5. Burp Suite: <https://burpsuite.com/>  
6. Nessus: <https://www.nessus.com/>  
7. OpenVAS: <https://www.openvas.org/>  
8. Qualys: <https://www.qualys.com/>  
9. Tenable: <https://www.tenable.io/>  
10. NIST Cybersecurity Framework Assessment Tool: <https://www.nist.gov/cybersecurity-framework-assessment-tool>

# Vulnerability: Cross-Site Scripting (XSS) Reflected

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Priority | Remediation Complexity | Affected Service | Affected Assets |
| Low | Low | Low | Web | notfakecompany.com |

## Description

The text describes a web vulnerability that involves injecting malicious scripts into a web page. When a user opens the page, the script executes in their browser, potentially allowing attackers to steal session cookies, redirect users to malicious websites, or deface the website. This vulnerability can be exploited by injecting malicious scripts into the web page, which can execute in the user's browser when they open the page. This can potentially allow attackers to steal session cookies, redirect users to malicious websites, or deface the website. To mitigate this vulnerability, it is recommended to implement strict user input validation and use prepared statements with parameterized queries. Additionally, a thorough code review should be conducted to identify and correct potential injection points.

## Impact

Session hijacking, data theft, and phishing attacks are common types of cyber attacks that can pose a significant threat to the security of a website. These attacks can result in the unauthorized access to sensitive information, including personal data and confidential business plans.  
  
Session hijacking occurs when an attacker gains access to a user's session by intercepting and manipulating the session ID. This can allow the attacker to impersonate the user and access their account, potentially resulting in data theft or other malicious actions.  
  
Data theft involves the unauthorized access and retrieval of sensitive information, such as passwords, credit card numbers, and other personal data. This can be achieved through various means, including phishing attacks, malware infections, and social engineering tactics.  
  
Phishing attacks involve the use of fraudulent emails, websites, or messages to trick users into revealing sensitive information, such as passwords or credit card numbers. These attacks can be highly convincing and difficult to detect, making them a significant threat to the security of a website.  
  
Website defacement is another type of cyber attack that can have a significant impact on a website's reputation and revenue. This involves the unauthorized modification or deletion of a website's content, potentially resulting in the loss of valuable data and the disruption of business operations.  
  
To mitigate these vulnerabilities, it is recommended to implement strong authentication and access controls, use encryption to protect sensitive data, and educate users on how to recognize and avoid phishing attacks. Additionally, regular security audits and vulnerability assessments should be conducted to identify and address any potential weaknesses in the website's security.

## Recommendations

To mitigate the risk of cross-site scripting (XSS) attacks, it is recommended to encode all user input before displaying it on a web page. This can be achieved by using a web application firewall (WAF) that is capable of detecting and blocking XSS attacks. Additionally, it is important to sanitize user input according to its intended use to prevent malicious code from being executed on the server.  
  
Here is a list of punctual actions that can be taken to improve the security of the web application:  
  
1. Implement input encoding to prevent the execution of malicious scripts.  
2. Use a web application firewall (WAF) to detect and block XSS attacks.  
3. Sanitize user input according to its intended use to prevent malicious code from being executed on the server.  
4. Regularly update the web application and its dependencies to ensure that known vulnerabilities are patched.  
5. Conduct regular security audits and penetration testing to identify and address any potential security issues.  
  
By following these recommendations, the web application can be made more secure against XSS attacks and other types of vulnerabilities.

## References

1. OWASP Top 10: <https://owasp.org/www-community/top10>  
2. NIST Cybersecurity Framework: <https://www.nist.gov/cybersecurity-framework>  
3. PCI DSS: <https://www.pcisecuritystandards.org/assessors\_and\_solutions/assessor\_toolkit>  
4. OWASP ZAP: <https://zap.owasp.org/>  
5. Burp Suite: <https://burpsuite.com/>  
6. Nessus: <https://www.nessus.com/>  
7. OpenVAS: <https://www.openvas.org/>  
8. Qualys: <https://www.qualys.com/>  
9. Tenable: <https://www.tenable.io/>  
10. NIST Cybersecurity Framework Assessment Tool: <https://www.nist.gov/cybersecurity-framework-assessment-tool>

## ABOUT THIS PROYECT



The RGS (Report Generative Security Tool) project has been developed for the Generative AI Agents Developer Contest organized by NVIDIA and LangChain. This project is created by Miguel Zabala (Nullsector), leverages open source software, built from scratch, to streamline the generation of comprehensive security audit reports. RGS harnesses the power of generative AI to provide detailed analyses and actionable recommendations for security vulnerabilities. The goal is to make it easier for users to produce professional-grade security reports with minimal effort. The RGS project encourages anyone to use the code for their personal projects and contribute to its improvement.