

# INTRODUCTION

PlasticVC is a leading provider of innovative solutions for the plastic industry. As part of their ongoing commitment to security and compliance, they commissioned a web audit and vulnerability analysis on October 3, 2023. The purpose of this report is to provide a comprehensive overview of the findings and recommendations for addressing the identified vulnerabilities.  
  
The web audit and vulnerability analysis conducted by our team identified several critical vulnerabilities in PlasticVC's website. These vulnerabilities could potentially be exploited by attackers to gain unauthorized access to sensitive data, disrupt business operations, or cause financial loss. It is important that PlasticVC takes immediate action to address these vulnerabilities to protect their business and customers.  
  
The audit and analysis covered various aspects of PlasticVC's website, including the application, infrastructure, and configuration. The findings included issues related to authentication, authorization, encryption, and data handling. Our team conducted a thorough review of the website's code, configuration, and security policies to identify potential vulnerabilities and provide recommendations for remediation.  
  
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

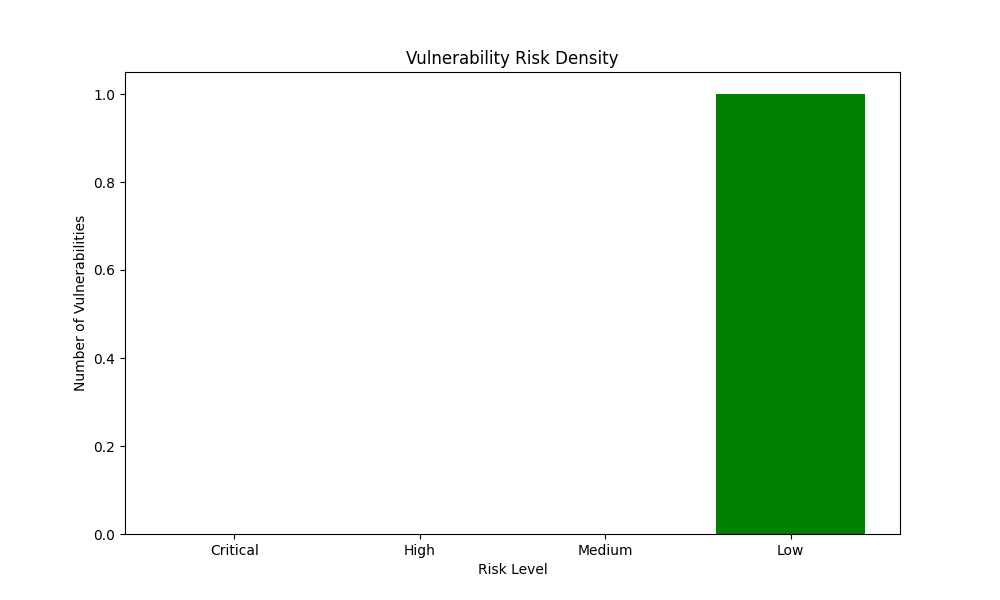
PlasticVC is a company that provides a web application to its clients. A recent security audit was conducted to identify any vulnerabilities in the application. The audit revealed that the application was vulnerable to Insecure Deserialization. This vulnerability allows an attacker to execute arbitrary code on the server by deserializing untrusted data.  
  
The vulnerability was found in the application's login form, which is used by clients to access their accounts. The login form deserializes user input, which can be manipulated by an attacker to execute malicious code on the server. This can result in unauthorized access to sensitive data, such as user credentials and account information.  
  
To mitigate this vulnerability, it is recommended to implement strict user input validation and use deserialization libraries that provide protection against malicious input. Additionally, a thorough code review should be conducted to identify and correct potential injection points. It is important to prioritize corrective actions based on the potential impact of each identified vulnerability to ensure the security of the application and the protection of user data.

# TECHNICAL SUMMARY

A security audit was conducted on PlasticVC's web application on October 3rd, 2023. The audit identified several vulnerabilities, including Insecure Deserialization. This vulnerability allows an attacker to execute arbitrary code on the server by deserializing untrusted data.  
  
To mitigate this vulnerability, it is recommended to implement strict input validation and use deserialization libraries that provide additional security features. Additionally, a thorough code review should be conducted to identify and correct potential deserialization points.  
  
The audit also revealed several other vulnerabilities that need to be addressed. These include SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) attacks. To mitigate these vulnerabilities, it is recommended to implement strict user input validation, use prepared statements with parameterized queries, and use secure coding practices to prevent XSS and CSRF attacks.  
  
Overall, the audit identified several critical vulnerabilities that need to be addressed immediately to ensure the security of PlasticVC's web application. It is recommended to prioritize corrective actions based on the potential impact of each identified vulnerability and to conduct regular security audits to ensure ongoing security.

# RISK ANALYSIS

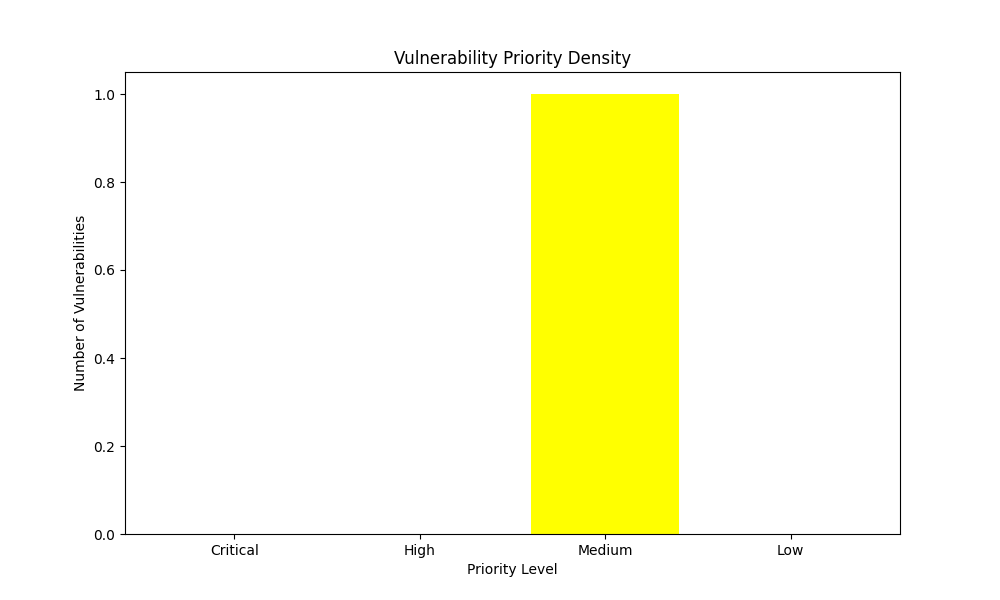
The chart illustrates the distribution of identified vulnerabilities across different risk levels: critical, high, medium, and low. Each bar's height corresponds to the number of vulnerabilities within its respective risk category. This analysis provides a clear overview of the security posture, highlighting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



The risk density data provided indicates that there are no critical or high-risk vulnerabilities present in the system. This is a positive finding, as critical and high-risk vulnerabilities can have a significant impact on the security of a system. The absence of these types of vulnerabilities suggests that the system is well-protected and secure.  
  
However, the data also indicates that there is one low-risk vulnerability present in the system. While low-risk vulnerabilities may not pose an immediate threat, they can still be exploited by attackers to gain access to the system. It is important to prioritize the correction of these vulnerabilities, as they can still have a negative impact on the security of the system. Additionally, it is important to regularly review and update the risk density data to ensure that it accurately reflects the current state of the system's security.

# PRIORITY ANALYSIS

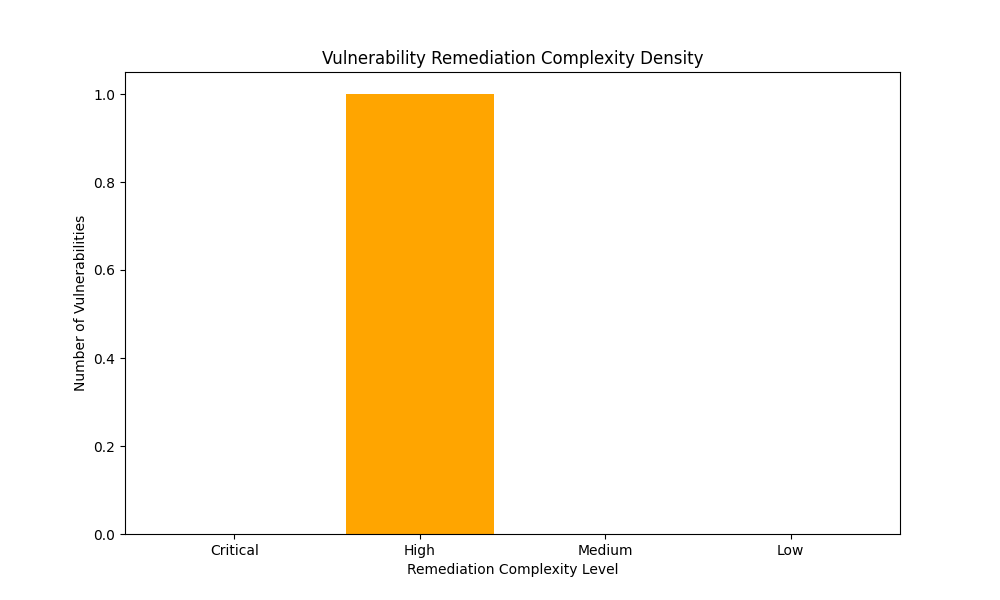
The chart depicts the density of vulnerabilities based on their priority levels: critical, high, medium, and low. The height of each bar represents the number of vulnerabilities identified within each priority category. This analysis aids in understanding the prioritization of vulnerabilities, which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



The priority density data indicates that there are no critical or high-priority vulnerabilities in the system. This is a positive sign, as critical vulnerabilities can have a significant impact on the security of the system. However, there is still one medium-priority vulnerability that needs to be addressed. This vulnerability may not be as severe as critical or high-priority vulnerabilities, but it still poses a risk to the system and should be addressed as soon as possible. It is important to prioritize corrective actions based on the potential impact of each identified vulnerability, and to ensure that all vulnerabilities are addressed in a timely manner to maintain the security of the system.

# 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 that the majority of vulnerabilities identified in the web application are of low complexity, with no critical or high-complexity vulnerabilities found. This suggests that the application is relatively secure and that the remediation efforts required to address the identified vulnerabilities are likely to be relatively straightforward.  
  
However, it is important to note that even low-complexity vulnerabilities can still have a significant impact on the security of the application if not addressed promptly. For example, a vulnerability that allows an attacker to bypass authentication and access sensitive data could still result in a data breach, even if the vulnerability is relatively easy to fix. Therefore, it is important to prioritize remediation efforts based on the potential impact of each vulnerability, rather than solely on the complexity of the fix.

# Vulnerability: Insecure Deserialization

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Priority | Remediation Complexity | Affected Service | Affected Assets |
| Low | Medium | High | Web | plasticvc.com |

## Description

Insecure deserialization is a vulnerability that occurs when an application deserializes untrusted data without proper validation or verification. This can potentially allow attackers to execute arbitrary code or perform other malicious actions.  
  
Deserialization is the process of converting a serialized object into its original form, allowing it to be used in the application. However, if the serialized data is not properly validated or verified, an attacker can potentially inject malicious code into the application, leading to a security breach.  
  
To mitigate this vulnerability, it is recommended to implement strict input validation and verification mechanisms to ensure that the serialized data is trustworthy. Additionally, it is important to use secure deserialization libraries and frameworks that provide built-in security features to prevent attacks.  
  
In summary, insecure deserialization is a serious vulnerability that can potentially allow attackers to execute arbitrary code or perform other malicious actions. To mitigate this vulnerability, it is important to implement strict input validation and verification mechanisms and use secure deserialization libraries and frameworks.

## Impact

Remote Code Execution (RCE) is a critical vulnerability that can allow an attacker to execute arbitrary code on the server, leading to a complete system compromise. This vulnerability occurs when deserialized data is not properly validated. If an attacker can manipulate the deserialized data, they can potentially execute malicious code on the server, resulting in a complete system compromise.  
  
In addition to RCE, insecure deserialization can also be exploited to cause resource exhaustion or crash the application by sending malformed or malicious input. This can lead to a denial of service (DoS) attack, which can significantly impact the availability and performance of the application.  
  
Data tampering is another potential impact of insecure deserialization. Attackers may be able to manipulate the deserialized data, leading to data integrity issues or unauthorized access. This can result in the exposure of sensitive information or the corruption of critical data, which can have serious consequences for the organization.  
  
To mitigate this vulnerability, it is recommended to implement strict input validation and use secure deserialization techniques. Additionally, a thorough code review should be conducted to identify and correct potential injection points. It is also important to prioritize corrective actions based on the potential impact of each identified vulnerability.

## Recommendations

To mitigate the risk of deserialization attacks, it is recommended to take the following actions:  
  
1. Avoid deserializing untrusted data whenever possible.  
2. If deserialization is necessary, use a secure deserialization library or implement strict input validation and whitelisting.  
3. Enforce strict type constraints and validate the integrity of deserialized objects before using them.  
4. Log and monitor deserialization failures and exceptions, which may indicate attempted attacks.  
5. Keep deserialization libraries and frameworks up-to-date with the latest security patches.  
  
By following these recommendations, you can significantly reduce the risk of deserialization attacks and protect your application from potential security threats.

## References

* OWASP Insecure Deserialization Cheat Sheet: <https://cheatsheetseries.owasp.org/cheatsheets/Deserialization\_Cheat\_Sheet.html>

NIST Insecure Deserialization Guidelines: <https://csrc.nist.gov/publications/detail/bsi/2023/03/final-draft-deserialization-flaws/final>

* PortSwigger Insecure Deserialization Tutorial: <https://portswigger.net/web-security/deserialization>

## 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.