# SUMMARY OF TECHNICAL FINDINGS

The following table is a summary listing of notable issues identified by ABC over the course of the security assessment.

## A - System A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref# | Issue Description | Risk | Likelihood | Consequence |
|  | Password Management: Empty Password in Configuration File | Critical | Probable | Severe |
|  | Weak Cryptographic Hash | High | Probable | Major |
|  | Cross Site Scripting: Persistent | Medium | Possible | Moderate |
|  | XML External Entity Injection | Medium | Possible | Moderate |

## B - System B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref# | Issue Description | Risk | Likelihood | Consequence |
|  | Password Management: Empty Password in Configuration File | Critical | Probable | Severe |
|  | Weak Cryptographic Hash | High | Probable | Major |
|  | Cross Site Scripting: Persistent | Medium | Possible | Moderate |
|  | XML External Entity Injection | Medium | Possible | Moderate |

# DETAILED FINDINGS & RECOMMENDATIONS

## A

This section details the risks that were identified for the A component of the security assessment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Password Management: Empty Password in Configuration File | Critical | Probable | Severe |
| It was observed that an empty string was used as password. The lack of password policy increases the chances of an attacker being able to guess valid user credentials.  **Recommendation:**  It is recommended to enforce strong password complexity. Application should enforce strong password complexity requirements whenever user passwords was set. The following guidelines are recommended when considering strong minimum passwords complexity requirements: - Minimum passwords length of 8 characters; - Require at least 1 of the following character types:  - Uppercase;  - Lowercase;  - Numeric; and   - Symbol. - Blacklist passwords based on common words (i.e. Welcome1!, P@ssword1)  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Weak Cryptographic Hash | High | Probable | Major |
| It was observed that the function uses weak cryptographic hashes which cannot guarantee data integrity and should not be used in security-critical contexts.  MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts.  Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.  **Recommendation:**  It is recommended to use SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3 as good alternatives for encryption.  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Cross Site Scripting: Persistent | Medium | Possible | Moderate |
| It was observed that several methods sends unvalidated data to the server which can allow results that was stored in the database to execute malicious code on browser when returned to the web browser.  XSS is a part of the injection class of vulnerabilities, where injected HTML can be used to render JavaScript submitted to the application that is returned to the user's browser without being safely encoded. Attackers can embed malicious payloads that perform a variety of actions including stealing credentials or other sensitive data such as customer information.  **Recommendation:**  It is recommended to validate user input and and perform HTML encoded output. Client supplied input should be limited to a whitelist of acceptable character sets. For example, for many user input fields this could instead be restricted to solely alphanumeric characters.  If non-alphanumeric characters are required, encode them as HTML entities before using them in an HTTP response, so that they cannot be used to modify the structure of the HTML document. The solution to XSS is to ensure that validation occurs in the correct places and checks are made for the correct properties.  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | XML External Entity Injection | Medium | Possible | Moderate |
| It was observed that the XML parser configured does not prevent nor limit external entities resolution. This can expose the parser to XML External Entities attack. XML Enternal Entities attacks leverage on the XML features that allow building of documents dynamically at the time of processing. This allows attacker to include data from external URI which will access resourced specified. This behavior exposes the application to XML External Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems.  **Recommendation:**  It is recommended to configure the XML parser securely such that it external entities will not be allowed to be executed as part of an incoming XML document.  XXE attacks can be prevented by disabling XML entity resolution through disabling inline DTD setting DtdProcessing to DtdProcessing.Prohibit or by disabling XML Entity resolution setting the XmlReaderSettings.XmlResolver property to null If external entities must be processed in the application, custom XmlResolver should be created with the following features: - Set a request timeout to prevent infinite delay attacks - Limit the amount of data that it will retrieve - Restrict the XmlResolver from retrieving resources on the localhost  **References:** | | | | |

## B

This section details the risks that were identified for the B component of the security assessment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Password Management: Empty Password in Configuration File | Critical | Probable | Severe |
| It was observed that an empty string was used as password. The lack of password policy increases the chances of an attacker being able to guess valid user credentials.  **Recommendation:**  It is recommended to enforce strong password complexity. Application should enforce strong password complexity requirements whenever user passwords was set. The following guidelines are recommended when considering strong minimum passwords complexity requirements: - Minimum passwords length of 8 characters; - Require at least 1 of the following character types:  - Uppercase;  - Lowercase;  - Numeric; and   - Symbol. - Blacklist passwords based on common words (i.e. Welcome1!, P@ssword1)  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Weak Cryptographic Hash | High | Probable | Major |
| It was observed that the function uses weak cryptographic hashes which cannot guarantee data integrity and should not be used in security-critical contexts.  MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts.  Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.  **Recommendation:**  It is recommended to use SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3 as good alternatives for encryption.  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | Cross Site Scripting: Persistent | Medium | Possible | Moderate |
| It was observed that several methods sends unvalidated data to the server which can allow results that was stored in the database to execute malicious code on browser when returned to the web browser.  XSS is a part of the injection class of vulnerabilities, where injected HTML can be used to render JavaScript submitted to the application that is returned to the user's browser without being safely encoded. Attackers can embed malicious payloads that perform a variety of actions including stealing credentials or other sensitive data such as customer information.  **Recommendation:**  It is recommended to validate user input and and perform HTML encoded output. Client supplied input should be limited to a whitelist of acceptable character sets. For example, for many user input fields this could instead be restricted to solely alphanumeric characters.  If non-alphanumeric characters are required, encode them as HTML entities before using them in an HTTP response, so that they cannot be used to modify the structure of the HTML document. The solution to XSS is to ensure that validation occurs in the correct places and checks are made for the correct properties.  **References:** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Security Assessment | | Risk | Likelihood | Consequence |
|  | XML External Entity Injection | Medium | Possible | Moderate |
| It was observed that the XML parser configured does not prevent nor limit external entities resolution. This can expose the parser to XML External Entities attack. XML Enternal Entities attacks leverage on the XML features that allow building of documents dynamically at the time of processing. This allows attacker to include data from external URI which will access resourced specified. This behavior exposes the application to XML External Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems.  **Recommendation:**  It is recommended to configure the XML parser securely such that it external entities will not be allowed to be executed as part of an incoming XML document.  XXE attacks can be prevented by disabling XML entity resolution through disabling inline DTD setting DtdProcessing to DtdProcessing.Prohibit or by disabling XML Entity resolution setting the XmlReaderSettings.XmlResolver property to null If external entities must be processed in the application, custom XmlResolver should be created with the following features: - Set a request timeout to prevent infinite delay attacks - Limit the amount of data that it will retrieve - Restrict the XmlResolver from retrieving resources on the localhost  **References:** | | | | |