|  |
| --- |
| DASH, Denver aerospace hub |
| Arranged By: Brooklynn Silva & Jonathan Rumley |
| MALI Security Requirements Machine Learning AI Web Application |

Table of Contents

[Introduction 3](#_Toc53218376)

[Background 3](#_Toc53218377)

[Mission Statement 3](#_Toc53218378)

[Assessment Breakdown 3](#_Toc53218379)

[Residual Risk Factors 4](#_Toc53218380)

[Likelihood of Risk 4](#_Toc53218381)

[Security Requirements 5](#_Toc53218382)

[Requirements 5](#_Toc53218383)

[Security Requirements Matrix 6](#_Toc53218394)

[Assessments 8](#_Toc53218395)

[Security Costs 8](#_Toc53218396)

[Threats 8](#_Toc53218397)

[Vulnerabilities 9](#_Toc53218398)

[Exploits 9](#_Toc53218399)

[Mitigation 9](#_Toc53218400)

[Design 10](#_Toc53218401)

[Implementation 10](#_Toc53218402)

[Verification 11](#_Toc53218403)

[Conclusion 11](#_Toc53218404)

[References 12](#_Toc53218405)

# Introduction

## Background

DASH, Denver Aerospace Hub, was originated to strictly focus on aerospace and defense industry system technologies. The MALI web application will strengthen the security frameworks to avoid future exploitable vulnerabilites in the future by applying machine learning to implement and improve:

* Information Confidentiality
* Intelligence & Surveillance
* Preventive maintenance
* Research and development
* Threat detection

## Mission Statement

The determination of the MALI web application security requirements is to ensure that any of the aerospace or defense technologies mounted with MALI would be secure from outside threats including the following fields:

* Aerospace and Defense Parts Wholesalers
* Aerospace and Defense Products & Parts Manufacturing
* Guided Missile & Space Vehicle Manufacturing
* Industrial & Military Computer System Manufacturing

Integrating the MALI web application is an essential way to implement security into technologies that could pose serious threats to the globalization of the aerospace and defense industry that are without AI machine learning competences.

## Assessment Breakdown

The assessment outlines the security requirements required for the MALI web application to appropriately analyze the enterprise’s security. All requirements have been created with the intentions to deploy the most secure machine learning web application in the industry which entails routine security requirement reviews prior to finalization.

## Residual Risk Factors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4 Urgent** | **3 Elevated** | **2 Rising** | **1 Low** | **0 Informative** |

Overall risk factors for web applications vary by industry, but as for the aerospace and defense industry, web application security is top priority. Below provides’ the risk factors for the MALI web application security requirements for potential threats, exploits, and vulnerabilities in the future:

**4 Urgent** – Immanent security risk that must be addressed immediately resulting in very serious threat detections strategies, remediation efforts, and mitigations applied appropriately

**3 Elevated** – Serious security risk as threats could become inherent resulting in mass damage that requires attention to mitigations

**2 Rising** – Enough security risk to be called a potential threat that needs required attention to consider mitigations

**1 Low** – Security risks are beginning to increase as some attention is required

**0 Informative** – Lowest security risk with no required attention, intended to be informational

## Likelihood of Risk

The likelihood of risk varies upon the type of threat or vulnerability as potential attackers are continuing to learn how to hack technologies and systems just as fast as anyone else learning how to defend against any type of cyberthreat. Below indicates the likelihood of risk scale for potential threats:

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

**High** – Highest likelihood types of threats happen most frequently resulting in an urgent risk factor compared to a threat that doesn’t get executed as frequently

**Medium** – Medium likelihood types of threats happen less frequently, but can still range from urgent to informative on the risk factor scale resulting in a variety of

**Low** – Lowest likelihood type threats happen the most rarely, but still need to be taking just as serious as a threat occurs more frequently

# Security Requirements

## Requirements

**MALI01 Injection** – The MALI system shall comply with the OWASP requirements to recognize, manage and prevent SQL Injection attacks

**MALI02 Security Misconfiguration** – The MALI system shall comply with the OWASP requirements to ensure preventative maintenance will routinely find security flaws related to configurations

**MALI03 Broken Access Control** - The MALI system shall comply with the OWASP requirements to prevent broken access control from attackers breaking into the system allowing modifications to be made by acting like an administrator or user

**MALI04 Data Encryption** - The MALI system shall comply with the OWASP requirements to encrypt all sensitive data that should always be protected to avoid valuable data to be compromised

**MALI05 Broken Authentication** – The MALI web application shall comply with the OWASP requirements to ensure application functions related to authentication and session management are implemented correctly

**MALI06 Cross-Site Scripting** – The MALI web application shall comply with the OWASP requirements to require proper validation or escape when displaying untrusted data, and securely update existing web pages

**MALI07 Insecure Deserialization** - The MALI system shall comply with the OWASP requirements to encrypt all sensitive data that should always be protected to avoid valuable data to be compromised

**MALI08 XML External Entities** – The MALI system XML processors shall comply with the OWASP requirements, be kept up to date and properly configured in order to keep data, servers, and internal systems secure from attacks

**MALI09 Components** – The MALI system shall comply with the OWASP requirements to not comply with any type of unknown component or hardware that is known to have any type of vulnerability

**MALI10 Insufficient Logging & Monitoring**– The MALI web application shall be review frequently for any patching and updating that is required to maintain the prevention of security flaws that could lead to potential vulnerabilities

## Security Requirements Matrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Security Concern** | **Threat** | **Vulnerability** | **Mitigation** | **Damages** | **Risk** | **Likelihood** |
| **MALI01** | SQL Injections –  Unauthorized account access to data | Data breach resulting in data modified or lost and/or system take over | Execution of modified SQL queries resulting in unauthorized code commands | Input validation, dynamic analysis, & sanitization of data | Data corruption or loss, Permissions altered | 4  Urgent | HIGH |
| **MALI02** | Security Misconfiguration – Unsecure default configurations | Unpatched flaws, access default accounts, unused pages, unprotected data | Missing important security hardening, misconfigurations, or unnecessary services | Automated scanners or security hardening | Unauthorized access to some system data or functionality, and complete system compromise | 4  Urgent | MED |
| **MALI03** | Broken Access Control – Unauthorized user access | Unauthorized user is able to access data and application function | Lack of access control functionality or implementation | Access and permission standards—enforced with trusted code | Unauthorized access to application and data | 3  Elevated | MED |
| **MALI04** | Sensitive Data Exposure –  Data that isn’t encrypted over networks | Stolen clear text data from server side, transmitting, or client side | Lack of data encryption prior to transmitting via networks | Implementation of encryption standards | Data subjection | 4  Urgent | HIGH |
| **MALI05** | Broken Authentication –  User privileges are accessed by unauthorized user | Access to username & password combinations for credential stuffing, brute force, and dictionary attack tools | Weakness of design and implementation of most identity and access controls | Multi-factor authentication, weak password checks, comply with NIST 800-63 B’s password policy | Money laundering, social secuirty fraud, and identity theft or disclosure of legally protected highly sensitive data | 4  Urgent | MED |
| **MALI06** | Cross-Site Scripting – Exploitable frameworks | Session stealing, account takeover, node modifications | Malicious software, key logging, and other client-side attacks | Separation of untrusted data from active browser content | Stealing credentials, sessions, or delivering malware | 4  Urgent | HIGH |
| **MALI07** | Insecure Deserialization –  User-controllable data is deserialized by a website | Remote code execution attacks, privilege escalation, and arbitrary file access | Deserialization of user input leads to manipulated data | Implementation of integrity checks, enforcing strict type constraints, and monitoring | Remote code execution – malicious attacks or denial of entry | 3  Elevated | MED |
| **MALI08** | XML External Entities – Exploiting storage and transport data | Parsing XML input, accessing remote content via declared system identifier | Weakly configured SML parsers lead to the disclosure of confidential data, denial of service, and server-side request forgery | Use less complex data formats, avoid serialization of sensitive data, patch and/or upgrade all XML processors and libraries | Interference with application’s processing of XML data; disclosing local files | 3  Elevated | MED |
| **MALI09** | Using Components with Known Vulnerabilities –  Exploiting components without addressing vulnerabilities | Attacks on out of date and unpatched software and libraries | Failure to update, scan and patch vulnerable software components | Perform manual updates, scans, and patches to ensure all components/ libraries are up to date; maintain APIs, microservices, libraries and legacy code | Remote code execution – invoking software with full permission | 3  Elevated | MED |
| **MALI10** | Insufficient Logging & Monitoring –  Lack of logging issues, events, and solutions | Loss of internal source code, unauthorized access to accounts, fraud | Failure to create and share logs with team; failure to manually analyze application | Compliance with all log regulations that mandate assembly, storage, and evaluation of the logs | Loss of future updates and the inability to conduct event traceback; financial fraud | 2  Rising | MED |

# Assessments

## Security Costs

According to IBM’s “2020 Cost of a Data Breach Report”, the average total cost to recover from a data breach is $3.86 million. At DASH, we run the risk of having our intellectual property and intelligence stolen on a daily basis. The last thing we want to do is spend millions of dollars recovering these properties -- this is where MALI comes into play. Secure MALI developers work on the early design stages of this technology, as the mitigation of security issues is much less expensive during this timeframe. Even with these professionals’ salaries starting at $80k and ranging up to $250k, we are saving money by having a strong security team that has the ability to find and correct vulnerabilities before they reach the surface, potentially creating a detrimental problem. Without our Secure MALI Developers, we would face a much higher chance of MALI vulnerabilities slipping through the cracks, costing DASH not only millions, but our credibility.

## Threats

Threatening groups are often in pursuit of DASH’s valuable data in order to monitor our actions, advance opposing aerospace and defense capabilities, develop counter-measures, and produce copied technologies to make a profit.

Threats can come from many different directions, making it crucial to address any potential threats early on in the design stage of software development in order to keep critical data secure. Things to consider when gauging the threat:

* Historic risks
* Potential for future attacks
* Emerging threats
* Security of existing software systems
* Possible threat actors
  + Hackers
  + Insiders
  + Competing intelligence professionals
  + Terrorists
  + Criminal organizations

Threat modeling is the practice of identifying and prioritizing potential threats and security mitigations in order to protect our valued data. MALI is designed to aid in these threat modeling steps:

* Identify the vulnerable asset(s)
* Create an overview of the software architecture
* Break down the application
* Identify the threat
* Document the threat
* Rate the threat

## Vulnerabilities

Vulnerabilities arise as a result of imprecise engineering practices, and in this industry, can be catastrophic if foes are given the opportunity to take advantage of them. These weaknesses can open the software to exploits at the network, host, or application levels. In order to avoid potential vulnerabilities, it is imperative to keep an eye open for mishaps, or sloppy work. The following mistakes are common causes for vulnerabilities:

* Lack of clear security goals
* Developers having unclear roles
* Inadequate planning and tracking
* Failing to recognize security risks early
* Ignoring security and quality until the end of the Software Development Lifecycle (SDL)

## Exploits

If any of the above vulnerabilities are missed or not addressed, exploits may occur. At DASH, these attacks can happen if a threat actor were to take advantage of any weaknesses in MALI, which can lead to a disastrous security breach.

\*Refer to MALI Security Requirements matrix\*

## Mitigation

Mitigations are methods and/or tools used to prevent or limit exploits against vulnerabilities. After performing root cause analysis, any vulnerabilities must be isolated in order to correct them. At DASH, we use active mitigation, constantly working towards security in the beginning stages of MALI design by:

* Characterizing the root causes of vulnerabilities that have been identified in earlier phases of MALI analysis
* Evaluate these vulnerabilities; what causes them and how they interact with one another
* Identify alternative mitigation strategies, methods, and tools for each specific vulnerability
* Select and commit the resources required for the specific alternatives
* Communicate planning results to all project participants for implementation

## Design

As mentioned earlier, we see the design phase of MALI as the most influential time to establish the software’s trustworthiness. Here at DASH, we follow standard software development principles when designing MALI:

* Economy of mechanism
  + Keep the design as simple and small as possible
* Fail-safe defaults
  + Base access decisions on permission rather than exclusion
* Complete mediation
  + Every access to every object must be checked for authority
* Separation of privilege
  + A protection mechanism that requires two keys to unlock it is more robust and flexible that one that allows access to the presenter of only a single key
* Least privilege
  + Every program and every user of the system should operate using the least set of privileges necessary to complete the job
* Least common mechanism
  + Minimize the amount of mechanisms common to more than one user and depended on by all users
* Psychological acceptability
  + It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly

## Implementation

While developing MALI, DASH follows these sequential steps from the SDLC in order to properly implement our security requirements and create safe, secure, and functional software:

* Establish the scope and boundaries; assess the landscape
* Incorporate the previously described security requirements into MALI design
* Educate the MALI Developers on all security requirements
* Assign responsibility of security to the Secure MALI Developers
* Hold security-focused meetings to perform risk analyses
* Establish a mitigation plan to ensure proper security design
* Perform architecture reviews and threat modeling
* Carry out code reviews as a condition to passing a release gate
* Execute test plans and perform penetration tests
* Launch MALI
* Maintain MALI

![Diagram

Description automatically generated]()\*Image A--Learntek

## Verification

At DASH, we use the Level 3 OWASP Application Security Verification Standard (ASVS) in order to verify that the latest version of MALI is ready to be securely released. ASVS Level 3 is designed for the most critical applications, securing sensitive data and establishing the highest level of trust. This community-driven verification standard is aligned with the MALI security requirements and focuses on defining the security controls required when designing, developing, and testing the software. ASVS Level 3 includes:

* High assurance
* Security architecture and reviews
* Secure coding
* Standards and checklists
* Secure and peer code review
* Unit and integration tests
* Hybrid reviews

## Conclusion

When following all of DASH’s security requirements, MALI works as an Artificial Intelligence web application that uses standard best practices to keep intruders and attackers out of our critical, private data. This protects our products, technologies, and studies. Without MALI, our security costs and chance of exploits would increase greatly, so it is imperative to follow the OWASP requirements along with our established DASH requirements that go along with them to uphold our strong image and credibility.

# References

Collasius, Erik. (2020). “6 Ways to Improve Defense and Aerospace Security Strategies”.

*JABIL.* jabil.com/blog/defense-and-aerospace-security.html

Comiter, Marcus. (2019, August). “Attacking Artificial Intelligence: AI’s Security Vulnerability

and What Policymakers Can Do About It”. *Belfer Center for Science and*

*International Affairs.* belfercenter.org/publication/AttackingAI

“Cost of a Data Breach Report 2020”. (2020). *IBM.* ibm.com/security/data-breach

Foster, Stuart. (2020, July 27). “Top 10 Software Vulnerabilities + Vulnerabilities Definition”.

*Security & Compliance | Static Analysis.* perforce.com/blog/kw/common-software-

vulnerabilities

“OWASP Application Security Verification Standard 4.0”. (2019, March). *OWASP.*

owasp.org/www-pdf-archive/OWASP\_Application\_Security\_Verification\_Standard\_

4.0-en.pdf

“SDLC (Software Development Life Cycle)”. (2019, January 25). *Learntek.*

Learntek.org/blog/sdlc-phases