OWASP Application Security Verification

Standard 3.0

Contents

[Bank System Application 2](#_Toc485624970)

[Detailed Verification Requirements 3](#_Toc485624971)

[V1. Architecture, desing and threat modelling 4](#_Toc485624972)

[V2: AUTHENTICATION VERIFICATION REQUIREMENTS 5](#_Toc485624973)

[V3:SESSION MANAGEMENT VERIFICATION REQUREMENT 8](#_Toc485624974)

[V4: ACCESS CONTROL VERIFICATION REQUIREMENT 10](#_Toc485624975)

[V5:MALICIOUS INPUT HANDLING VERIFICATION REQUIREMENTS 10](#_Toc485624976)

[V7:CRYPTHOGRAPHY AT REST VERIFICATION REQUIREMENTS 11](#_Toc485624977)

[V8:ERROR HANDLING AND LOGGIN VERIFICATION REUREMENTS 11](#_Toc485624978)

[V9:DATA PROTECTION VERIFICATION REQUIREMENTS 12](#_Toc485624979)

[V10: COMMUNICATIONS SECURITY VERIFICATION REQUIREMENTS 14](#_Toc485624980)

[V11:HTTP SECURITY VERIFICATION REQUIREMENTS 15](#_Toc485624981)

[V13: MALICIOUS CONTROLS VERIFICATION REQUIREMENTS 15](#_Toc485624982)

[V15: BUSINESS LOGIC VERIFICATION REQUIREMENTS 15](#_Toc485624983)

[V16: FILES AND RESOURCES VERIFICATION REQUIREMENTS 16](#_Toc485624984)

[V17: MOBILE VERIFICATION REQUIREMENTS 16](#_Toc485624985)

[V18: WEB SERVICE VERIFICATION REQUIREMENTS 16](#_Toc485624986)

OWASP ASVS stands for Application Security Verification Standard, of which in this verification will be used the latest one, version 3.0. The focus is on creating a detailed analysis compared to the standard which seeks to provied normalization, as well as functional and non-function security controls required when desinging, developing and testing modern web applications.

Even if the coverage of the standard was perfect, the analysis further provided will not have been perfect, due to the risk analysis being a subjective process to some extent, which creates a challenge where it will most likely end in some parts being left behind.

ASVS has to main goals:

1. To help organizations develop and maintain secure applications
2. To allow security service, security tools vendors and consumers to align their requirements and offerings

This analysis will go through the basic steps of the ASVS pipeline to try and determine the security threats and risks:

1. Defining security requirements based on asvs requierments for a chosen level
2. Design for a particular risk level
3. Implementation
4. Performing verification where it is verified agains the selected asvs level
5. Deployment

In this analysis the choses ASVS level will be ASVS Level 2 which is for the application that contain sensitive data, which requires protection.

Each of these levels contains a list of security requirements. Each of these requirements can also be mapped to security specific features and capabilities, that must be built into software by developers.

In the next part the analysis will be presented where it will try and follow the standard with the hope of good results.

# Bank System Application

Different trheats have different motivations. Some industries have unique information and techonology assets and domain specific regulatory compliance requirements.

First it is important to identify the domain where the applicatoin is being used, and asses the securiy rist level based on the motivation for breaches that offers. Bank system an application represents a software that is designed to be used in the finanances. ASVS has provided level descriptions regarding industry of finances and insurance.

Type: Finance and Industries

With it, now the threat profile is to be acceses.

Threat Profile: Finance and Insuracne is often viewed as a high value target by motivated attackers and attacks are often financially motivated. Commonly, attackers are looking for sensitive data or account credentials that can be used to commit fraud or to benefit directly by leveraging money movement functionality built into applications. Techniques often include stolen credentials, application-level attacks, and social engineering.

After profiling, and identifying the domain, setup of the ASVS levels is possible.

1. ASVS L1 Recommendation: All netword accessible applications.
2. ASVS L2 Recommendation: Applications that contain sensitive information like credit card numbers, personal information, that can move limited amounts of money in limited ways. Examples include: money between accounts at the same institution or a slower form of money movement (e.g. ACH) with transaction limits or wire transfers with hard transfer limits within a period of time.

Bank System application fits really well into the profile as well as into the Level 1 recommendation and exampling.

# Detailed Verification Requirements

There are sixteen detaild verification requiremts to be met in the ASVS 3.0 standard:

1. Arhitecture, design and threat modelling V1
2. Authentication V2
3. Session management V3
4. Access control V4
5. Malicions input handling V5
6. Crypthography at rest V7
7. Error handling and logging V8
8. Data protection V9
9. Communications V10
10. HTTP security configuration V11
11. Malicious controls V13
12. Business logic V15
13. File and resources V16
14. Mobile V17
15. Web services V18
16. Configuration V19

# V1. Architecture, desing and threat modelling

**Control objective**

Ensure that a verified application satisfies the following high level requirements:

● At level 1, components of the application are identified and have a reason for being in the app

● At level 2, the architecture has been defined and the code adheres to the architecture

● At level 3, the architecture and design is in place, in use, and effective

Note: This section has been re-introduced in version 3.0, but is essentially the same architectural controls as version 1.0 of the ASVS.

Result:

Control objective

Ensure that a verified application satisfies the following high level requirements:

● At level 1, components of the application are identified and have a reason for being in the app

● At level 2, the architecture has been defined and the code adheres to the architecture

● At level 3, the architecture and design is in place, in use, and effective

The following results follow the ASVS Level 2 Detailed Verification Requirements

Results:

**1.1 – Partially met**

The application has a lot of components that are probably unneeded essentialy, however due to the the implementation, it is unsure how the application would respond to removing these probable loose ends. However, tha applicaton was developed with model driven development which insures a good foundation.

**1.2 – Half met**

The identified libraries, modules are mostly on the front end of the application, Jquery, AngularJS, MomentJS, Angular-RoutJS and BootstrapJS. These libraries have had known security problems before.

**1.3 – Mostly met**

High-level architecture has been defined in the form of conceptual model that is a product of Model Driven Software Development. Security threat modeling has been used that used as a main guide while securing the application.

**1.4 – Not met**

The application doesn’t have defined seucity control, nor firewall rules, nor cloud based security groups. These controls are to be considered in the future when the project is in its full capacity.

**1.5 – Mostly met**

The MVC is ensured on both the front end of the application with the help of AngularJS and as well on the back end with the usage of the Spring Boot framework that is MVC based.

**1.6 –Mostly met**

The breach of critical data into the client side has not been discovered after analysis. However it should be reviewed again. User data is being secured also with the help of Spring Security API that provides reasonable insurance policy.

# 

# V2: AUTHENTICATION VERIFICATION REQUIREMENTS

**Control objective**

Authentication is the act of establishing, or confirming, something (or someone)asauthentic,thatis,thatclaims made by or about the thing are true. Ensure that a verified application satisfies the following high level requirements:

● Verifies the digital identity of the sender of a communication.

● Ensures that only those authorised are able to authenticate and credentials are transported in a secure manner.

**Results:**

**2.1 - Mostly met**

All the pages and resources are being secured via back end interceptor that interecpts the requests for pages and resources and requires authentication for the ones that are intended to be secured, those that are public are not being denied.

**2.2 - Mostly met**

The password that are being given are being processed and not brought back to the client side. No leaks have been yet found.

**2.3 – Mostly met**

Authentication is being enforced via the interceptor as stated above. It serves for both client side and server side.

**2.4 – Partially met**

Password entry field fail to meet the assesment due to their only validation being the password length. However they do not prevent the users to state a long and complex password to be entered.

**2.5 – Mostly met**

The authenticaion is being implemented via Spring Security API that is shown to be a good usage for establishing secure workframe.

**2.6 – Mostly met**

Attackers cannot log in due to solid implementation and usage of Spring Securty.

**2.8 – Not met**

The functionalities like update password is probably to have some sorts of flaws and is likely to be improved since it laks some of the front end and back end mechanisms that provide secure changes and usage.

**2.9 – Mostly met**

The password change includes requirement for the old password, as well as the confirmation of the new one.

**2.12 – Mostly met**

All activities are logged via Sprign Boot logging, as well as SLJ4F dependency that provides organizes logging.

**2.13 – Partially met**

The password are being safely stored in the databes and hashed with a well know BcryptPasswordEncoder that represents a good practice. However the application lacks mechanisms that track and stop possible brute force attacks in form of constant login attemts or password changes.

**2.16 – Not met**

The encryption for the user credential is missing during the transport from the client side to the server side.

**2.17 –Partially met**

Changing or retrieving the password doesn’t not reveal the old password, however the password is sent in clear text to the server side.

**2.18 – Mostly met**

No group or network users’ information is being retrieved via login, password reset or forgot account functionality.

**2.19 – Mostly met**

No default passwords are proposed to the users. The user is to enter his own password when changing the password.

**2.20 – Not met**

Brute force attacks are possible, and the application fails to meet this requirement.

**2.21 – Mostly met**

All the information being transported for web services is encrypted as well as user information.

**2.22 – Mostly met**

Tokens are being used with the support of Spring Security with limited availabilty.

**2.23 – Not met**

No locking of the service is present when it comes to brute force attacks which in turns mean that this requirement cannot be met.

**2.24 – Cannot asses**

The application provides no secret question since it is a finances domain app that is highly prone to security risks.

**2.25 – Not met**

The application is not able to defent itself from this type of the attack. No defense mechanism is implemented.

**2.26 – Partially met**

Some of the functionalities require re-authentication however this is not fully implemented.

**2.27 – Not met**

Commonly used password can be used, as well as those connected to the username.

**2.30 – Partially met**

Application allows users to authenticate without the possibility of forgery via some methods, however as stated the data is not fully secured during the transport.

**2.31 – Not met**

Application does nto offer two-factor authentication or other strong authentication, or any similar scheme tat provides protection against username + password disclosure.

**2.32 – Mostly met**

Administrative interfaces are not accessible tountrusted parties . Constantinterception of the request for their usage is being done.

# V3:SESSION MANAGEMENT VERIFICATION REQUREMENT

**Control objective**

One of the core components of any web-based application is the mechanism by which it controls andmaintains the state for a user interacting with it. This is referred to this as Session Management and is defined as the set of all controls governing state-full interaction between a user and the web-based application.

Ensure that a verified application satisfies the following high level session management requirements:

● Sessions are unique to each individual and cannot be guessed or shared.

● Sessions are invalidated when no longer required and timed out during periods of inactivity.

**Results:**

**3.1- Mostly met**

The custom token/session manager is resistant agains all commom session managemt attacks, since it’s based on Spring Security API.

**3.2 - Mostly met**

Session/Token is being invalidated via Spring Security proven methods

**3.3 - Mostly met**

Spring Security tokens are being given are not infinite and have a expiration time in case of inactivity.

**3.5 – Mostly met**

All pages that require authentication have an easy and visible access to logout functionality.

**3.6 Mostly met – Cannot asses**

The seesion id is never available in URL’s, however it is not being used.

**3.7 – Mostly met**

Every successful authentication and re-authentication generates a new Spring Security token, with limited time in case of inactivity.

**3.10 – Mostly met**

This is being being met via Spring Security reliable mechanism.

**3.11 – Mostly met**

Session is not being used as a authentication and authorization directly.

**3.12 – Unknown**

**3.16 – Not met**

With the usage of Spring Security the active concurrent sessions can be managed.

**3.17 – Mostly met**

The user is able to terminte any active session at any given time.

**3.18 – Not met**

The user is not being monitored and prompted, or reminded of the necessity to deactivate the session afthe a password change.

# 

# V4: ACCESS CONTROL VERIFICATION REQUIREMENT

**Control objective**

Authorization is the concept of allowing access to resources only to those permitted to use them. Ensure that a verified application satisfies the following high level requirements:

● Persons accessing resources holds valid credentials to do so.

● Users are associated with a well-defined set of roles and privileges.

● Role and permission metadata is protected from replay or tampering.

**Results:**

**4.1 – Mostly met**

The principle of the least privilege has been followed. The users can only access functions, data files, URLs and proceses for which they possess specific authorization. (spoofing and elevation of privilege is unavailable)

Admin -> Bank adming -> Bank staff -> Clients

**4.4 – Mostly met**

Sensitive data is protected via RBAC and with the usage of request interceptor that pre-handles everything.

**4.5 – Mostly met**

Directory browsing is not available because of Spring Boot usage

**4.8 - Mostly met**

The request interceptor deals with fail attempts in a proper way not giving the request a chance to go further.

**4.9 – Mostly met**

All of the roles present in the presentation layer exist on the server side.

**4.10, 4.11, 4.12 – Mostly met**

Via interceptor and proper logging mechanisms and trustes API’s.

**4.13 – Mostly met**

The CRSF tokens are being denied with the help of Spring Security.

**4.14, 4.15, 4.16 – Partially met**

The system lacks in these requirements and has a lot of security risks, that are yet to be implemented, so they won’t be covered here.

# V5:MALICIOUS INPUT HANDLING VERIFICATION REQUIREMENTS

**Control objective**

The most common web application security weakness is the failure to properly validate input coming from the client or from the environment before using it. This weakness leads to almost all of the major vulnerabilities in web applications, such as cross site scripting, SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows. Ensure that a verified application satisfies the following high level requirements:

● All input is validated to be correct and fit for the intended purpose

● Data from an external entity or client should never be trusted and should be handled accordingly.

**Results:**

**5.1 – Mostly met**

Application is written in Java language, so environment is not suspectible to buffer overflows.

**5.3 – Mostly met**

If input is bad server side will reject it and log is shown.

**5.5 – Mostly met**

Every input is validated on back end via constraints javax.validation package.

**5.10 – Mostly met**

All SQL queries are resistant to sql injection. JPA uses parametrized queries .

**5.11 – Mostly met**

All SQL queries are resistant to LDAP injection. JPA uses parametrized queries .

**5.12 – Mostly met**

Application do not use shell using Runtime.exe, instead application is using Java API.

**5.13 – Mostly met**

Application does not provide input type file, so application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI).

**5.15 – Mostly met**

All string placed in input fields, are automatically encoded in utf-8 format.

**5.18 Not met**

There is no client side validation, only server side.

**5.19 Not met**

There is no whitelisting, nor greylisting nor blacklisting.

**5.20 Mostly met**

Structured data is strongly validated on server side with allowed characters, length and pattern.

**5.21 Partially met**

Unstructured data is strongly validated on server side with allowed type and length.

**5.22 Not met**

Untrusted HTML from WYSIWYG editors are not properly sanitized.

**5.26 Mostly met**

Authenticated data is cleared from client storage, after session is terminated.

# V7:CRYPTHOGRAPHY AT REST VERIFICATION REQUIREMENTS

# V8:ERROR HANDLING AND LOGGIN VERIFICATION REUREMENTS

**Control objective**

The primary objective of error handling and logging is to provide a useful reaction by the user, administarotrs and incident response teams. The objective is not to create massive amounts of logs, but high quality logs, with more signal than discarded noise.

High quality logs will often contain sensitive data, and must be protected as per local data privacy laws or directives. This should include:

● Not collecting or logging sensitive information if not specifically required.

● Ensuring all logged information is handled securely and protected as per its data classification.

● Ensuring that logs are not forever, but have an absolute lifetime that is as short as possible.

**8.1 –Mostly met**

No stack traces or output messages provide sensitive or compromising information. Managed by Spring Boot and JPA mechanisms.

**8.2 – Mostly met**

The error handling logic instanty denies access of the given request with the help of GlobalExceptionHandler and as well as JPA and Spring Boot mechanisms.

**8.3 – Partially met**

Loggin is present however it doesn’t not provide monitoring and loggin of activites such as constant failure events.

**8.4 – Mostly met**

Logs contain information about the owner of the request as well as the activiy requested and the response given, wheter it was error or success.

**8.5 – Mostly met**

Logging is managed by a reliable tool that proviedes defensive mechanisms against tampering.

**8.6 – Mostly met**

API being used has protective me chanisms.

**8.7 – Mostly met**

No sensitive data such as passwords are being logged.

**8.8, 8.9, 810, 11, 8.12 – Mostly met**

All of these requirements are being handled by the used library that logs the events that occur. Usage of Spring Boot loggin as well as SL4JF annotations.

# V9:DATA PROTECTION VERIFICATION REQUIREMENTS

**Control objective :**

There are three key elements to sound data protection: Confidentiality, Integrity and Availability (CIA). This standard assumes that data protection is enforced on a trusted system, such as a server, which has been hardened and has sufficient protections. The application has to assume that alluserdevicesarecompromisedin some way. Where an application transmits or stores sensitive information on insecure devices, such as shared computers, phones and tablets, the application is responsible for ensuring data stored on these devices is encrypted and cannot be easily illicitly obtained, altered or disclosed.

Ensure that a verified application satisfies the following high level data protection requirements:

● Confidentiality: Data should be protected from unauthorisedobservationordisclosurebothintransitand when stored.

● Integrity: Data should be protected being maliciously created, altered or deleted by unauthorized attackers.

● Availability: Data should be available to authorized users as required.

**Results:**

**9.1 – Mostly met**

No sensitive information is being caches and the cache is turned off via Spring Security.

**9.3 – Mostly met**

Almost no data is being send via URL’s, and especially sensitive data that might compromise the confidentiality.

**9.4- Not met**

Anti-caching hasn’t been set every thime, which puts users at risk when it comes to confidentiality of their sensitive data.

**9.5 – Mostly met**

Sensitive data on the server side is always being protected from unauthorized access.

**9.7 – Mostly met**

Cookies, path variables, header variables have been used minimally.

**9.9 – Not met**

**9.10 – Mostly met**

Every request activiy is being logged via interceptor, so is the case then with the sensitive data.

**9.11 – Partially met**

Data sometimes stays too long in the memory as well as in proccesses, needs to be improved.

# V10: COMMUNICATIONS SECURITY VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application satisfies the following high level requirements:

● That TLS is used where sensitive data is transmitted.

● That strong algorithms and ciphers are used at all times.

**Results:**

**10.1 –Mostly met**

Certificates are being generated locally and are transported in web services, that are secured, and certificate validatoin is done via OCSP.

**10.3 – Partially met**

TLS is not used for all connections.

**10.14 – Mostly met**

Proper certification revocation is done via implemented web services that are connected with the certificate tool that provides the OCSP protocol.

**10.15 – Mostly met**

Strong algorithms with sizing (1024, 2048) are being used when encrypting.

**10.16 – Partially met**

TLS lacks implementation.

# V11:HTTP SECURITY VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application satisfies the following high level requirements:

● The application server is suitable hardened from a default configuration. ● HTTP responses contain a safe character set in the content type header

One of the core components of any web-based application is the mechanism by which it controls andmaintains the state for a user interacting with it. This is referred to this as SessionManagement and is defined as the set of all controls governing state-full interaction between a user and the web-based application.

Ensure that a verified application satisfies the following high level session management requirements:

● Sessions are unique to each individual and cannot be guessed or shared.

● Sessions are invalidated when no longer required and timed out during periods of inactivity.

**11.1 – Mostly met**

The request are being handled via Spring Boot therefore the only requests being processed are the ones that were stated in the controllers. There are not TRACE-PUT-DELETE methods.

**11.2 – Mostly met**

Every HTTP response contained a content type with a sace character set.

**11.3, 11.4, 11.5, 11.6, 11.7 -> Requirements exceed knoweledge**

# V13: MALICIOUS CONTROLS VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application satisfies the following high level requirement:

● Detected malicious activity is handled securely and properly as to not affect the rest of the application

**These verifications were not handler due to the lack of knoweledge.**

# V15: BUSINESS LOGIC VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application satisfies the following high level requirements:

● The business logic flow is sequential and in order

**Result:**

**15.1 – Mostly met**

All the operations are in specific order following strict procedure, and are being handled through transactions, and JPA’s mechanisms like versioning in order to prevent any type of persistancy failiure.

**15.2 – Not met**

Application doesn’t have a monitoring system that tracks automated or unusual attacks.

# V16: FILES AND RESOURCES VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application satisfies the following high level requirements:

● Untrusted file data should be handled accordingly and in a secure manner. ● obtained from untrusted sources are stored outside the webroot and limited permissions.

**Result:**

No data or files are being retrieved from the client side.

However there are some leaks like no defense agains accessing from remote systems or resources.

# V17: MOBILE VERIFICATION REQUIREMENTS

The analyzed application doesn’t have any type of mobile support or verification, so in this case it will be excluded from the analysis.

# V18: WEB SERVICE VERIFICATION REQUIREMENTS

**Control objective**

Ensure that a verified application that uses RESTful or SOAP based web services has:

● adequate authentication, session management and authorization of all web services

● Input validation of all parameters that transit from a lower to higher trust level

● Basic interoperability of SOAP web services layer to promote API use

**Results:**

**18.1 – Mostly met**

Same encoding is being used between the applications that are communication via web services.

**18.3 – Partially met**

There are proper xsd schemas that are providing validation for xml documents, json validation is being done on rest services through JPA validators.

**18.4 – Mostly met**

Input data is being validated with JPA validators on given communication entities.

**18.5 – Mostly met**

WSDL is used to enable the description of services as sets of endpoints operating on messages.

**18.6 – Mostly met**

Session is based on Spring Security API.

**18.7 – Not met**

Spring csrf protection not implemented, csrf token not included.

**18.8 – Mostly met**

On rest controller, content-type is set to be expected one.

**18.9 – Not met**

Message payload between client and server is not signed.

**18.10 – Mostly met**

Alternatives and less secured paths do not exist.

# V19: Configuration

**Control objective**

Ensure that a verified application that uses RESTful or SOAP based web services has:

● Up to date libraries and platform(s)

● A secure by default configuration

● Sufficient hardening that user initiated changes to default configuration do not unnecessarily expose or create security weaknesses or flaws to underlying systems

**Results:**

**19.1 – Mostly met**

All components are up to date, back end uses Spring Boot API, with Spring Security, while front end uses AngularJS.

**19.2 – Partially met**

Communicaton between back end and database server is encrypted for only sensitive data that is store. Back end hashes sensitive data with BcryptPasswordEncoder and then send it to database server. Other data is not encrypted.

**19.3 – Mostly met**

Communicaton between back end and database server authenticated using account with least necessary privileges.

**19.4 – Not met**

Application deployments are not adequaly sandboxed, nor containerized nor isolated.

.