# Baxter Credit Union

## Web Application Penetration Test

### Lumin API Testing

July 06, 2022

The following files are part of the entire report:
**WAPT-BCU-2022-07-06.pdf (This Report)**
**DetailedFindings-BCU-WAPT-2022-07-06.xlsx**

## Objective

The objective of the Web Application Penetration Test was to provide valuable insight regarding the ability of Baxter Credit Union's web applications to resist attacks over the Internet and locally from unauthorized and valid users.

## Scope

The scope of the Web Application Penetration Test focused on the security posture of the application(s) listed below. The testing was performed between May 09, 2022 and May 20, 2022.

Table 1: Scope

| Asset | Description |
|---|---|
| https://dev-digitalbankingcashadvanceapi.asedev.bcu.org/ | DigitalBanking Cash Advance API |
| https://dev-digitalbankingcdrenewalapi.asedev.bcu.org/ | DigitalBanking CD Renewal API |
| https://dev-digitalbankingcheckimageapi.asedev.bcu.org/ | DigitalBanking Check Image API |
| https://dev-digitalbankingedocumentapi.asedev.bcu.org/ | DigitalBanking E-Dcoument API |
| https://dev-digitalbankingmortgageapi.asedev.bcu.org/ | DigitalBanking Mortgage API |
| https://dev-digitalbankingpowerpluscheckingapi.asedev.bcu.org/ | DigitalBanking PowerPlus Checking API |
| https://dev-digitalbankingshareapi.asedev.bcu.org/ | DigitalBanking Share API |
| https://dev-digitalbankingskipapayapi.asedev.bcu.org/ | DigitalBanking SkipAPay API |
| https://dev-digitalbankingtravelnoteapi.asedev.bcu.org/ | DigitalBanking Travel Notification API |
| https://dev-digitalbankingvcaapi.asedev.bcu.org/ | DigitalBanking Vca API |

## Executive Summary

Intended Audience: Management, Board, Audit and Supervisory Committees

Baxter Credit Union (BCU) contracted CliftonLarsonAllen (CLA) to perform an Web Application Penetration Test in May 2022.

**Best Practices Observed**
During the WAPT, CLA identified best practices implemented in BCU's environment. These included:

- API endpoints that handle user data required an authentication token, disallowing unauthorized access.
- BCU was able to quickly remediate critical and medium-risk vulnerabilities while CLA was still actively testing.

**Prioritized Vulnerabilities**
From CLA's perspective, the following vulnerabilities have the largest impact to the organization:

- One API endpoint was vulnerable to SQL injection which allowed CLA to extract the content of the database such as credit card numbers.
- One API endpoint was vulnerable to XML injection which could give an attacker access to sensitive information. CLA was not able to exploit vulnerability within the testing timeframe.

**Prioritized Recommendations**
From CLA's perspective, the following remediation activities should be prioritized to address the key observations:

- CLA validated the SQL injection vulnerability was remediated in production.
- CLA validated the XML injection vulnerability was remediated in production.

## Approach

This web application penetration test procedure follows the OWASP Testing Guide v4. The engagement was conducted utilizing a test account for the purposes of logging into the application to evaluate the security of the application and threat to user data. It is designed to take a holistic approach in analyzing the dynamic processes of the web application.

**Information Gathering**
The first phase in security assessment is focused on collecting as much information as possible about a target application. Information Gathering is the most critical step of an application security test. The security test should endeavor to test as much of the code base as possible. Thus mapping all possible paths through the code to facilitate thorough testing is paramount.

This task can be carried out in many different ways.

By using public tools (search engines), scanners, sending simple HTTP requests, or specially crafted requests, it is possible to force the application to leak information, e.g., disclosing error messages or revealing the versions and technologies used.

**Configuration and Deployment Management**
Understanding the deployed configuration of the server hosting the web application is almost as important as the application security testing itself. After all, an application chain is only as strong as its weakest link. Application platforms are wide and varied, but some key platform configuration errors can compromise the application in the same way an unsecured application can compromise the server.

**Identity Management Testing**
This section deals with account privileges, and access. The tester spends most of their time during this phase on the login page working to understand how the application allows users to sign up and whether this system can be exploited if you know part of the login information (like the username).

During the configuration and deployment management testing, the tester looked for administrator interfaces. During Identity management testing, all possible application roles (user, administrator, author, etc.) are to understand what access or privileges come with different roles.

Finally, the tester digs into the system to prepare for future tests by checking whether error messages give clues about existing usernames and trying to find username patterns to help them find those existing usernames and accounts.

**Authentication Testing**
Authentication is the act of establishing or confirming something (or someone) as authentic, that is, that claims made by or about the thing are true. Authenticating an object may mean confirming its provenance, whereas authenticating a person often consists of verifying her identity. Authentication depends upon one or more authentication factors.

In computer security, authentication is the process of attempting to verify the digital identity of the sender of a communication. A common example of such a process is the log on process. Testing the authentication schema means understanding how the authentication process works and using that information to circumvent the authentication mechanism.

One of the core components of any web-based application is the mechanism by which it controls and maintains 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. This broadly covers anything from how user authentication is performed, to what happens upon them logging out.

**Authorization Testing**
Authorization is the concept of allowing access to resources only to those permitted to use them. Testing for Authorization means understanding how the authorization process works, and using that information to circumvent the authorization mechanism.

Authorization is a process that comes after a successful authentication, so the tester will verify this point after he holds valid credentials, associated with a well-defined set of roles and privileges. During this kind of assessment, it should be verified if it is possible to bypass the authorization schema, find a path traversal vulnerability, or find ways to escalate the privileges assigned to the tester.

**Session Management Testing**
HTTP is a stateless protocol, meaning that web servers respond to client requests without linking them to each other. Even simple application logic requires a user's multiple requests to be associated with each other across a "session". This necessitates third party solutions – through either Off-The-Shelf (OTS) middleware and web server solutions, or bespoke developer implementations. Most popular web application environments, such as ASP and PHP, provide developers with built-in session handling routines. Some kind of identification token will typically be issued, which will be referred to as a "Session ID" or Cookie.

There are a number of ways in which a web application may interact with a user. Each is dependent upon the nature of the site, the security, and availability requirements of the application. Whilst there are accepted best practices for application development, such as those outlined in the OWASP Guide to Building Secure Web Applications, it is important that application security is considered within the context of the provider's requirements and expectations.

**Input Validation Testing**
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 the major vulnerabilities in web applications, such as cross site scripting, SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows.

Data from an external entity or client should never be trusted, since it can be arbitrarily tampered with by an attacker. "All Input is Evil", says Michael Howard in his famous book "Writing Secure Code". That is rule number one. Unfortunately, complex applications often have numerous entry points, which makes it difficult for a developer to enforce this rule. This chapter describes Data Validation testing. This is the task of testing all the possible forms of input to understand if the application sufficiently validates input data before using it.

**Testing for Error Handling**
An important aspect of secure application development is to prevent information leakage. Error messages give an attacker great insight into the inner workings of an application.
The purpose of reviewing the Error Handling code is to assure the application fails safely under all possible error conditions, expected and unexpected. No sensitive information is presented to the user when an error occurs.

**Cryptography**
The assessor checks whether and how sensitive data is being protected during transmission and whether it is possible for an attacker to decrypt the encrypted data.

**Business Logic Testing**
Testing for business logic flaws in a multi-functional dynamic web application requires thinking in unconventional methods. If an application's authentication mechanism is developed with the intention of performing steps 1, 2, 3 in that specific order to authenticate a user. What happens if the user goes from step 1

straight to step 3? In this simplistic example, does the application provide access by failing open; deny access, or just error out with a 500 message?

There are many examples that can be made, but the one constant lesson is "think outside of conventional wisdom". This type of vulnerability cannot be detected by a vulnerability scanner and relies upon the skills and creativity of the penetration tester. In addition, this type of vulnerability is usually one of the hardest to detect, and usually application specific but, at the same time, usually one of the most detrimental to the application, if exploited.

The classification of business logic flaws has been under-studied; although exploitation of business flaws frequently happens in real-world systems, and many applied vulnerability researchers investigate them. The greatest focus is in web applications. There is debate within the community about whether these problems represent particularly new concepts, or if they are variations of well-known principles.

Testing of business logic flaws is similar to the test types used by functional testers that focus on logical or finite state testing. These types of tests require that security professionals think a bit differently, develop abused and misuse cases and use many of the testing techniques embraced by functional testers. Automation of business logic abuse cases is not possible and remains a manual art relying on the skills of the tester and their knowledge of the complete business process and its rules.

**Client Side Testing**
Client-Side testing is concerned with the execution of code on the client, typically natively within a web browser or browser plugin. The execution of code on the client-side is distinct from executing on the server and returning the subsequent content.

## Tools

The Web Application Penetration Test was conducted using the following tools:

- Burp Suite: An integrated platform for performing security testing of web-enabled application, as well as an intercepting proxy used to man-in-the-middle requests from client to server and analyze responses being returned from the service
- Dradis Framework: Collaboration and reporting tool for information security teams.
- Fierce: Fierce is a reconnaissance tool designed to scan domains to locate likely targets both inside and outside a corporate network.
- GXFR: GXFR replicates domain transfers by enumerating subdomains with public facing web servers using advanced search engine queries.
- The Harvester: The Harvester is an open source intelligence tool (OSINT) for obtaining email addresses and user names from public sources such as Google or LinkedIn.
- Medusa: Medusa is remote login brute-force that supports numerous services.
- Nikto: A web server scanner that tests web servers for dangerous files/CGIs, outdated server software and other problems.
- Nmap: A utility for network discovery and security auditing.
- Nessus: A vulnerability scanning tool.
- sqlmap: sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.

## Severity Definitions

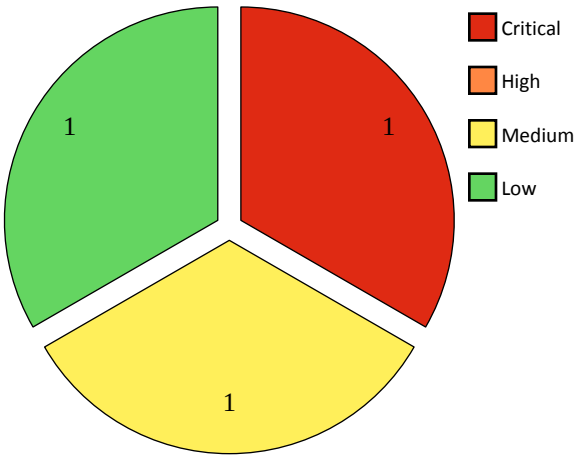
Findings are assigned a severity rating. Severity was determined based on the Common Vulnerability Scoring System (CVSS) score, where applicable, along with our expertise with consideration of the impact to the organization. CLA also has a recommended "time to fix" for each severity to help management develop timeframes for remediation. The CVSS provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Two common uses of CVSS are prioritization of vulnerability remediation activities and in calculating the severity of vulnerabilities discovered on one's systems. The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities.

Table 2: Severity Definition Table

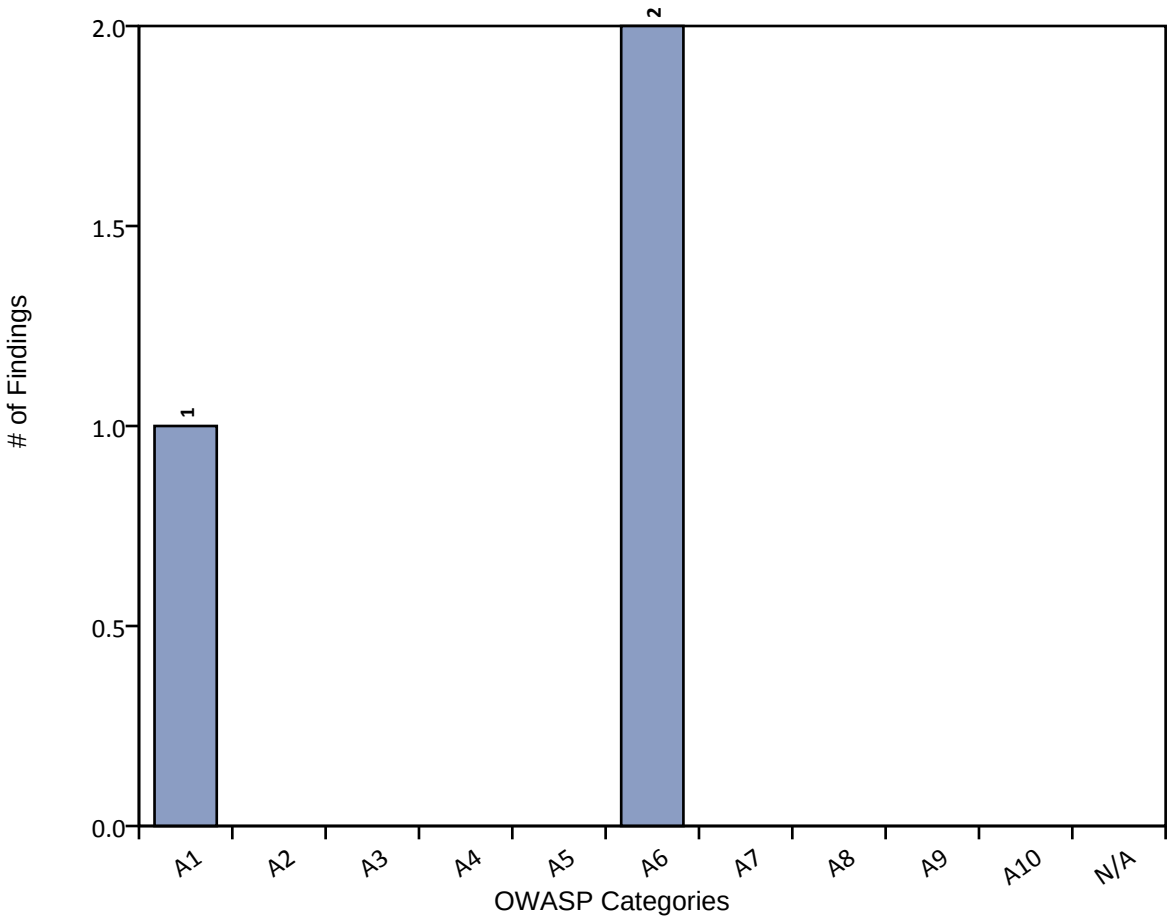
| Severity | CVSS | Time to Fix | Definition |
|---|---|---|---|
| Critical | 9.0 – 10.0 | < 30 days | This vulnerability is rated Critical by the vendor, or based on the CVSS score, or this vulnerability by itself directly impacts confidentiality, integrity and availability of systems and/or sensitive information. The vulnerability can be exploited remotely across the network. |
| High | 7.0 – 8.9 | 30 – 60 days | This vulnerability is rated High by the vendor, or based on the CVSS score, or this vulnerability by itself potentially impacts confidentiality, integrity or availability of systems and/or sensitive information. |
| Medium | 4.0 – 6.9 | 60 – 120 days | This vulnerability is rated Medium by the vendor, or based on the CVSS score, or in combination with other vulnerabilities has the potential to impact confidentiality, integrity or availability of systems and/or sensitive information. |
| Low | 0.1 – 3.9 | 120+ days | This vulnerability is rated Low by the vendor, or based on the CVSS score, or this vulnerability does not meet "best practice," but does not directly impact confidentiality, integrity or availability of systems and/or sensitive information. |

## Finding Severity Chart (Unique Findings)

The pie chart below represents the number of unique findings for each severity level.



## Summary of Vulnerabilities OWASP 2017 Top 10 Categories




A1: Injection
A2: Broken Authentication
A3: Sensitive Data Exposure
A4: XML External Entities (XXE)
A5: Broken Access Control
A6: Security Misconfiguration
A7: Cross - Site Scripting (XSS)
A8: Insecure Deserialization
A9: Using Components with Known Vulnerabilities
A10: Insufficient Logging & Monitoring

## Summary of Findings

The table below summarizes the observations from the engagement. For technical details and remediation strategies for each of these findings, please see the accompanying spreadsheet.

Table 3: Summary of Findings

| Finding | Description | Severity |
|---|---|---|
| 1 | SQL Injection | Critical |
| 2 | XML injection | Medium |
| 3 | Strict transport security not enforced | Low |