

Bank of CSEC 380 Audit

April 24, 2019

This report was submitted to:

**Bank of CSEC**

1 Lomb Memorial Drive  
Rochester, NY 14623

For more information contact:

**Smayan Daruka**

Student

**Chaim Sanders**

Professor

# Table of Contents

1 Executive Summary 4

2 Overview 5

3 Technical Risks 8

3.1 High Priority Risks 9

3.1.1 Weak Reset Token 9

3.1.2 Unlimited Withdrawals 9

3.1.3 Unlimited Deposits 9

3.1.4 Reflected Cross Site Scripting (XSS) 9

3.1.5 Local File Inclusion 9

3.2 Medium Priority Risks 10

3.2.1 Directory Browsing or Indexing 10

3.2.2 PHP Info 10

3.2.1 X-Frame-Options header not set 10

3.3 Low Priority Risks 11

3.3.1 HTTPOnly Flag not set 10

3.3.2 X-Content-Type-Options Header is missing 11

# Document Revision History

| Version | Modification | Date | Author |
| --- | --- | --- | --- |
| 0.1 | Initial document | 4/24/19 |  |
| 0.2 | Added findings | 4/24/19 | Smayan Daruka |
|  |  |  |  |

# Executive Summary

The goal behind this penetration test was to determine vulnerabilities in the Bank of CSEC web application. This project spanned over a period of 24 hours from April 24 to April 25. After performing a careful analysis of the web application using various tools at my disposal, such as OWASP ZAP and sqlmap to name a few, I found quite a few vulnerabilities that require attention. I used a risk level matrix to accurately determine the risk of all vulnerabilities and determine which ones require immediate attention.

My final report is continued below and hope that my findings are helpful in creating a more secure environment for the bank’s customers.

# Overview

Risk Level Methodology

Once the technical risks have been identified, they are prioritized based on severity, determined by its risk and impact on business goals.

The risk value represents the danger that the technical risk poses to the Bank of CSEC infrastructure and is calculated using the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Impact** | | |
| **Low** | **Medium** | **High** |
| **Likelihood** | **High** | **Low** | **Medium** | **High** |
| **Medium** | **Low** | **Medium** | **Medium** |
| **Low** | **Low** | **Low** | **Low** |

Table 1: Risk Level Matrix

The risk rating helps drive the mitigation planning based on the following guiding principles:

* **High**: Mitigation should be scheduled as soon as possible.
* **Medium**: Bank of CSEC should develop a timeline to remediate or accept the risk on a case by case basis.
* **Low**: The Bank of CSEC team may decide to accept the risk or remediate.

# Technical Risks

Technical risks result from an actual or potential vulnerability in the . Technical risks are summarized in the following table along with an evaluation of the likelihood, impact and calculated risk value. The risk ratings in the table below are based on the National Institute of Standards and Technology Risk Management Guide (see Section 2).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Technical Risk** | **Likelihood** | **Impact** | **Risk Severity** |
| 3.1.1 | Weak Reset Token | H | H | H |
| 3.1.2 | Unlimited Withdrawals | H | H | H |
| 3.1.3 | Unlimited Deposits | H | H | H |
| 3.1.4 | Reflected Cross Site Scripting (XSS) | H | H | H |
| 3.1.5 | Local File Inclusion (LFI) | H | H | H |
| 3.2.1 | Directory Browsing or Indexing | M | H | M |
| 3.2.2 | PHP Info | M | H | M |
| 3.2.3 | X-Frame-Options header not set | M | M | M |
| 3.3.1 | HTTPOnly Flag not set | L | L | **L** |
| 3.3.2 | X-Content-Type-Options Header is missing | L | L | **L** |

Table 2: Technical Risks

## High Priority Risks

### Weak Reset Token

Weak reset tokens allow attackers to easily guess and enumerate tokens to reset passwords. The way it is setup right now, if an attacker knows someone’s username, they can easily reset their password using the app’s functionality.

Likelihood – High

It is very likely attackers will reset legitimate users’ passwords and impersonate them and steal their money.

Impact – High

This affects every single client of the bank since their password can be reset by anyone.

**Steps to Reproduce**

1. Go to the bank’s website: <http://1ab7f1513cf9716d1c224cb02ea14a8b206544.csec380practical.com/>
2. Click on sign-in at top right.
3. Click on the link to reset your password.
4. Enter the username for which you wish to reset the password. For example, enter admin.
5. Click on the link that is displayed on the page after.
6. It resets the password and displays it on the screen in plain text effectively giving you complete access of the account.

Mitigation

Ensure that the reset tokens or links are sent via email to the email address on file for the user. This will ensure that a third party cannot reset anyone’s password.

### Unlimited withdrawals

Current functionality of the app allows an individual to use more money than they currently have in their account or have access to.

Likelihood – High

It is very likely that users will overdraw on their accounts since there is nothing in place to stop them from doing so currently. Also, since their balance literally goes into the negative, that is a very serious security risk.

Impact – High

This affects every user or client of the bank.

**Steps to Reproduce**

1. Log into the bank’s website.
2. Click on pay bill.
3. Enter any amount bigger than the current balance for proof of concept
4. After the transaction has happened, the bank balance goes into the negative.

Mitigation

Ensure that bank balances cannot go into the negative and not allow users to perform transactions that will overdraft their account.

### Unlimited deposits

This vulnerability is very similar to the one above, but users can add an infinite amount of money to their account.

Likelihood – High

There is a high likelihood of this being exploited because who does not like free money.

Impact – High

This also affects every user of the bank.

**Steps to Reproduce**

1. Log into the bank’s website.
2. Click on pay bill.
3. Enter any amount in negative, for instance -5000.
4. After the transaction has happened, the bank balance goes up by the above amount.

Mitigation

Perform input validation to ensure that users cannot enter negative numbers in the pay bill section.

### Reflected Cross Site Scripting (XSS)

Cross Site Scripting is often used to steal session cookies that allow an attacker to impersonate the victim. Reflected XSS is a non-persistent attack where the attacker’s payload has to be a part of the request that is sent to the web server. XSS allows an attacker to inject code in an input field.

Likelihood – High

The likelihood of this attack is a high since it allows an attacker to infect a large amount of people fairly easily.

Impact – High

Attackers will be able to affect all users regardless of they are.

**Steps to Reproduce**

1. Log into the website and click on contact us at top right.
2. At the bottom of the page, click “here” to join the mailing list.
3. At this point, we enter a regular email address to see what the functionality is to add an email to the mailing list. We see below that the email is sent as a parameter to “?email=”.

A screenshot of a social media post

Description automatically generated

1. We craft the link in the screenshot above and inject script as shown in the screenshot below to exploit XSS.

A screenshot of a social media post

Description automatically generated

Mitigation

In order to mitigate this vulnerability, we can implement multiple options:

* Only allow untrusted data in certain locations
* Make sure to escape input before actually using it anywhere
* Try to avoid JavaScript input in URLs and validate to ensure that XSS cannot happen
* Lastly, set the HTTPOnly cookie flag to ensure that sessions cannot be stolen since they cannot be viewed by JavaScript.

### Local File Inclusion (LFI)

This vulnerability allows an attacker to include a file and it usually occurs when the input is not properly validated. LFI allows an attacker to read contents of files stored locally on the machine and can even lead to code execution on the web server.

Likelihood – High

There is a high likelihood that attackers will exploit this vulnerability to gain remote access to the machine or try and read confidential files off of the web server.

Impact – High

This vulnerability has a high impact since one can read highly important and confidential files and use that to further exploit the web application.

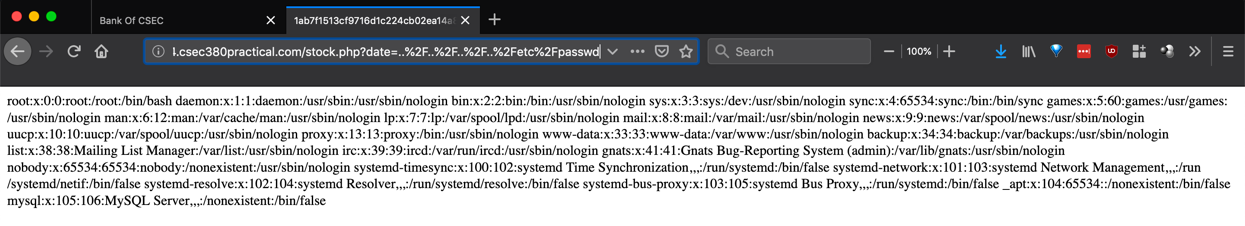
**Steps to Reproduce**

1. Log into the bank’s website as admin.
2. Click on “Modify Page”.
3. Select the “stock.php” page from the dropdown.
4. We see here that data is provided to the “date” parameter and all other requests are considered invalid.

A screenshot of a cell phone

Description automatically generated

1. Now we send a crafted link to the “stock.php” page as shown in the screenshot below. I printed the contents of the /etc/passwd file to enumerate all the usernames. One thing to keep in mind is to encode the characters as also shown below. I replaced “/” with “%2F” as is the hex representation.



Mitigation

In order to mitigate this vulnerability, we can simply disable remote file inclusion capabilities. Another method is to avoid using any input provided by the user directly for a system call or anything else of the sorts.

## Medium Priority Risks

### Directory Browsing or Indexing

It is possible to view the directory listing and find files that might include scripts or even backup sources and their files. Basically, this vulnerability also potentially gives access to sensitive information. As can be seen below, there are 4 places where directory indexing is enabled.

A screenshot of a social media post

Description automatically generated

Likelihood – Medium

The likelihood is medium since directory browsing, albeit able to provide access to sensitive information, is not often exploited. It is also possible to not find any useful information, so it depends on the scenario.

Impact – High

It has a medium impact since any files found will not be HIGHLY confidential in nature most of the times so any information found will likely be used to further another exploit.

**Steps to Reproduce**

1. Navigate to any of the links shown in the screenshot above to see a directory listing for that particular location. The screenshot below just shows a single example.

A screenshot of a cell phone

Description automatically generated

Mitigation

One possible mitigation is to disable directory browsing entirely. If that is not possible, then ensure that the files listed in the directory are not sensitive and confidential in nature.

### PHP Info

It is possible to view the entire PHP version and configuration file for the webserver.

Likelihood – Medium

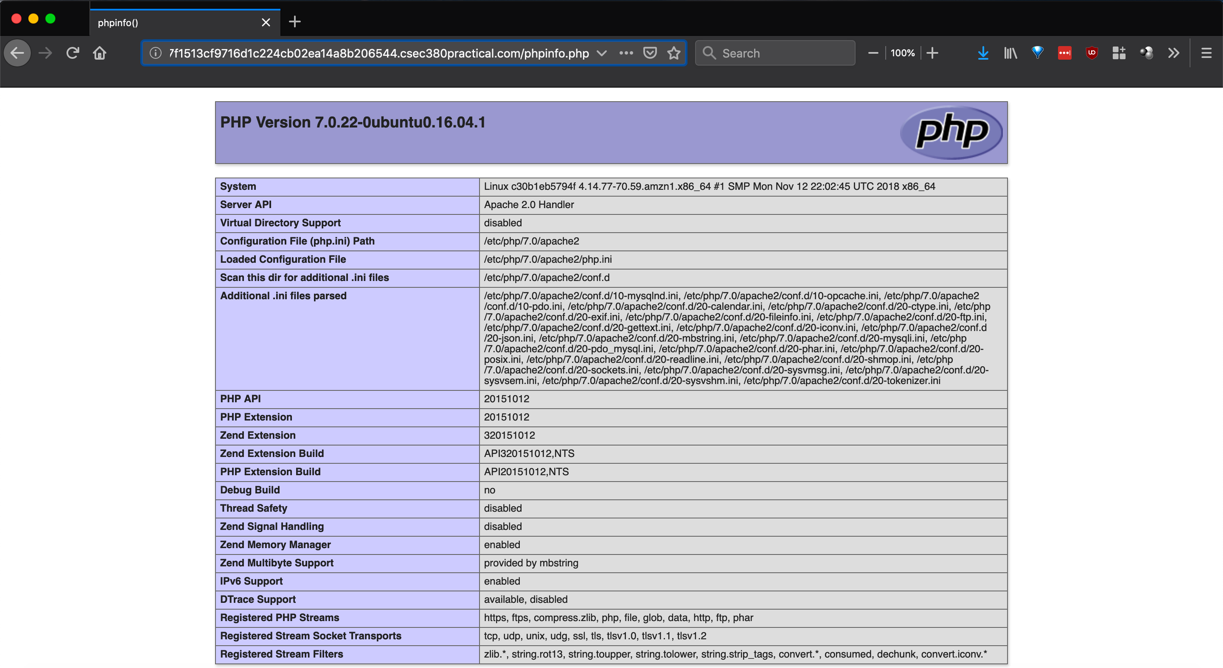
The likelihood is medium since all the information that is available in this file is very useful to an attacker in planning their exploits.

Impact – High

The reason for this having a high impact is mainly what I explained above. This file lists all the configuration for the webserver and there is a lot of useful information available here to attack and gain access to the webserver. Information found here can be used in conjunction with Metasploit to gain remote access.

**Steps to Reproduce**

1. In the screenshot below, I navigated to “/phpinfo.php” to view this page.



Mitigation

The best mitigation is to remove this from being publicly accessible. In other words, change the configuration on the webserver to hide this page from being public.

### X-Frame-Options header not set

This HTTP response header indicates whether a browser is allowed to render a page using frames or not. Currently, this header is not set which means the default value takes effect. The default is to only allow pages within frames with the same origin. This header is most often set to protect against clickjacking attacks.

Likelihood – Medium

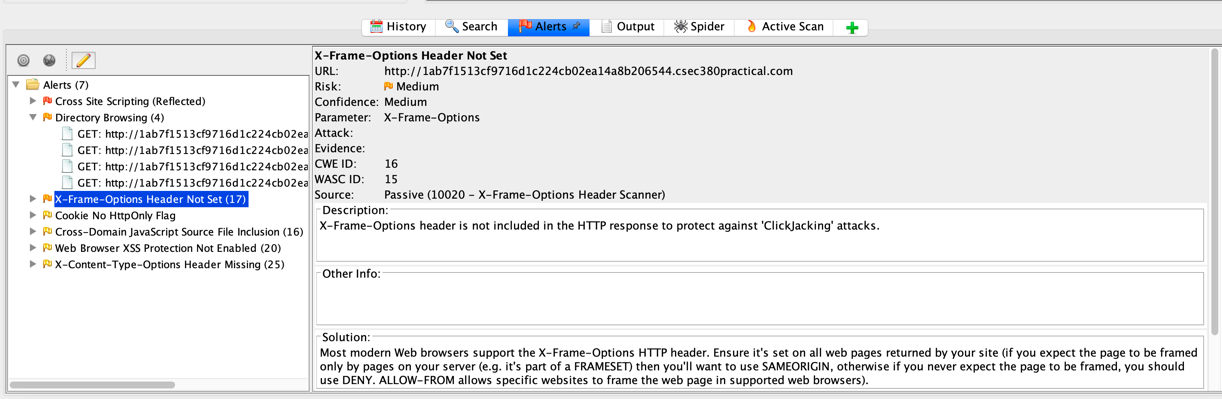
The likelihood is medium since due to the default value, an attacker cannot load a malicious webpage in a frame but rather only exploit what is natively available.

Impact – Medium

This is of medium impact due to the reasons I mentioned above.

**Steps to Reproduce**

1. In order to reproduce this issue, simply navigate to any page and craft the URL to include an iframe.



Mitigation

Set the X-Frame-Options header to “DENY” making sure that the website does not load any frames whatsoever. Another method is to set it to “SAMEORIGIN” so only pages within the domain can load frames.

## Low Priority Risks

### HTTPOnly Flag not set

This flag ensures that cookies cannot be accessed via JavaScript. It essentially protects against a client-side script from accessing the cookie.

Likelihood – Low

The likelihood is low since attackers are much more likely to explore other avenues of attack.

Impact – Low

This has a low impact since it also has a low likelihood of occurring.

**Steps to Reproduce**

1. There is not a real step to reproduce here, but the vulnerability is that a client-side script can view cookies as shown in the screenshot below.

A screenshot of a social media post

Description automatically generated

Mitigation

Set the HTTPOnly flag in the header to “TRUE” to ensure that no client-side scripts can access the cookies. One important thing to note is that this functionality needs to be supported by the browser since an incompatible browser will just ignore this flag. Most modern browsers have support for this functionality so it should be fairly simple to mitigate this vulnerability.

### X-Content-Type-Options Header is missing

This header ensures that the advertised content types should not be changed and must be followed. It ensures that if a style content type is requested, only CSS files can be served by the webserver.

Likelihood – Low

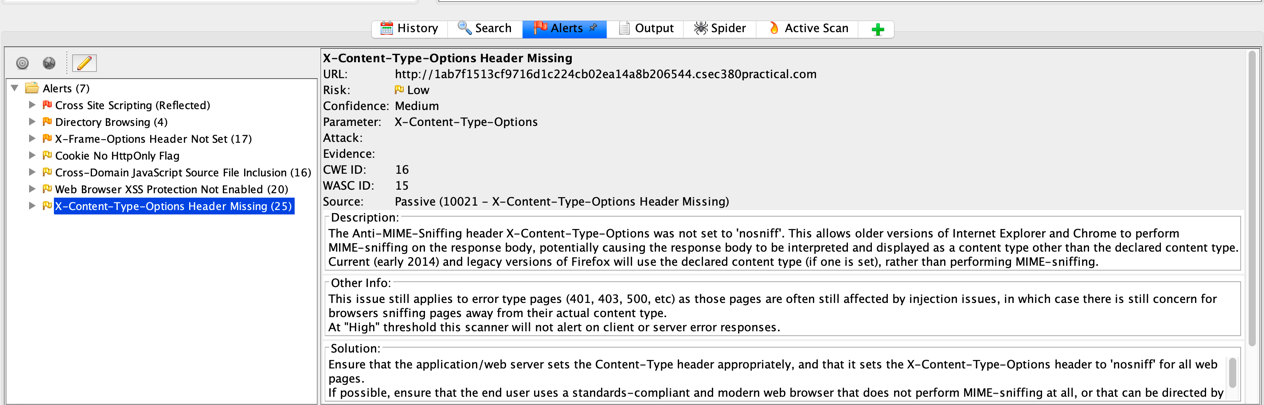
The likelihood is low since attackers are much more likely to explore other avenues of attack.

Impact – Low

This has a low impact since it also has a low likelihood of occurring.

**Steps to Reproduce**

1. Once again, there are no real steps to reproduce this vulnerability but here is a screenshot showing the vulnerability scan from OWASP ZAP.



Mitigation

Set the header to “nosniff” so as to protect against displaying a content type other than the declared content type.