



Lien-Wallet Penetration Test

Completed by: CertiK, LLC
August 28, 2020

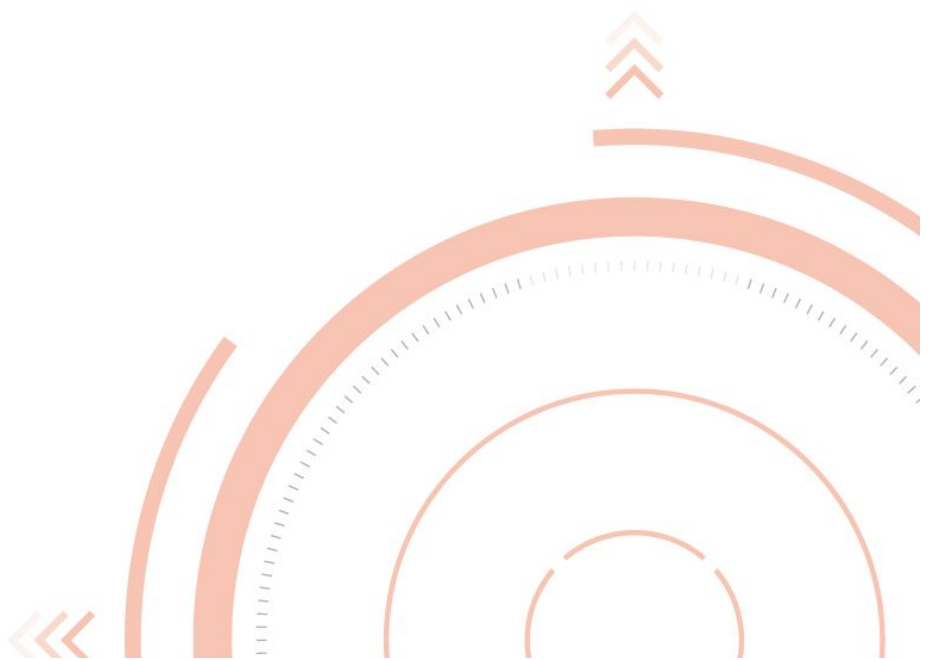


Table of Contents

1. Executive Summary	3
1.1. Scope	3
1.2. Limitations	4
1.3. Summary of Results	4
1.4. Summary of Recommendations	5
2. Web Application Penetration Test	5
2.1. Clickjacking	6
2.2. Directory Listing	9
2.3. Reconstruct source code with source map	10
2.4. Add liquidity transaction with 0 iDOL and 0 LIEN	12
3. Appendix – Methodology	13
3.1. Coverage and Prioritization	13
3.2. Reconnaissance	14
3.3. Application Mapping	14
3.4. Vulnerability Discovery	14
3.5. Vulnerability Confirmation	14
3.6. Immediate escalation of High or Critical Findings	15
3.7. Vulnerability Classes	15
3.8. Risk Assessment	16
4. Why CertiK	18

Confidentiality Statement

All information contained in this document is provided in confidence for the sole purpose of adjudication of the document and shall not be published or disclosed wholly or in part to any other party without CertiK's prior permission in writing and shall be held in safe custody. These obligations shall not apply to information that is published or becomes known legitimately from some source other than CertiK.

All transactions are subject to the appropriate CertiK Standard Terms and Conditions.

Certain information given in connection with this proposal is marked "In Commercial Confidence". That information is communicated in confidence, and disclosure of it to any person other than with CertiK's consent will be a breach of confidence actionable on the part of CertiK.

Disclaimer

This document is provided for information purposes only. CertiK accepts no responsibility for any errors or omissions that it may contain.

This document is provided without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall CertiK be liable for any claim, damages or other liability (either direct or indirect or consequential), whether in an action of contract, tort or otherwise, arising from, out of or in connection with this document or the contents thereof.

This document represents our budgetary price proposal for the solution further described in this herein and is provided for information and evaluation purposes only and is not currently a formal offer capable of acceptance.

1. Executive Summary

Lien.finance engaged CertiK to perform an application penetration test for their web application. The main objective of the engagement is to test the overall resiliency of the application to various real-world attacks against the application's controls and functions, and thereby be able to identify its weaknesses and provide recommendations to fix and improve its overall security posture.

CertiK started the test on August 24, 2020 and completed on August 28, 2020.

After a thorough review of the application, CertiK believes that the Lien.finance application is currently at a **Low** risk level. Given the severity of the vulnerabilities on the application, it is unlikely that the application will be directly compromised.

The most significant vulnerability was Clickjacking. An attacker can trick the victim into clicking on actionable content in the Lien application by loading it in a hidden iframe. Other weaknesses were also found and are detailed on the Findings section of the report.

Lien.finance has worked diligently to remediate security vulnerabilities discovered by CertiK, greatly increasing the overall security posture of their application. We suggest that Lien.finance maintain this level of security on future development and leverage our team for a follow up Penetration Test within 6 months, or immediately after and major development changes.

1.1. Scope

At the start of the engagement, CertiK worked with Lien.finance to identify the target and set the limits on the scope of the test. A White Box type of testing approach was done where CertiK performed the test with the source code available from the shared GitHub repository.

The following details the target scope of the test.

Application Name	Lien wallet web application
Target URL	https://testalphaalpha.github.io/beta/ https://beta-livid.vercel.app https://app.lien.finance/
Github Repository	https://github.com/LienProtocol/Lien-wallet https://github.com/LienProtocol/Lien-modules
Environment	Testing

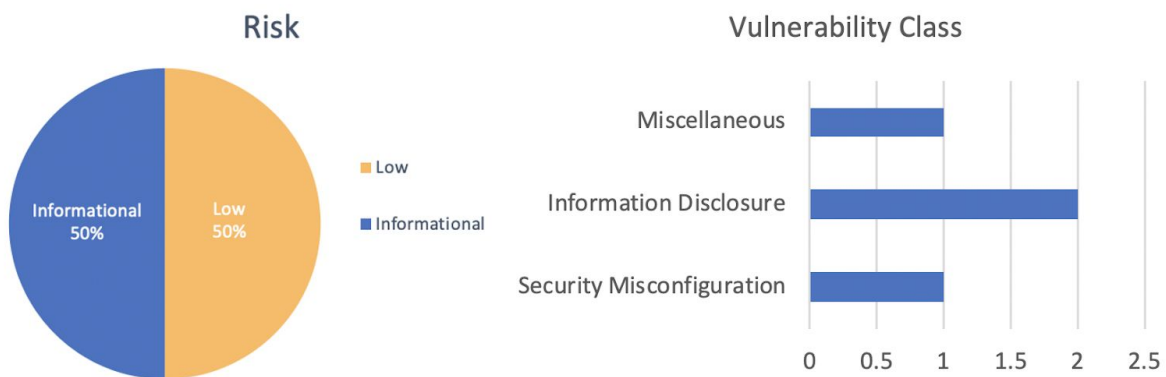
1.2. Limitations

No major limitations were identified during the test.

Testing was performed during regular hours as well as off hours throughout the course of the test.

1.3. Summary of Results

Below summarize the vulnerabilities found on the application:



ID	SEVERITY	TITLE	Vulnerability Class	Status
2.1	Low	Clickjacking	Security Misconfiguration	Fixed
2.2	Low	Directory Listing	Information Disclosure	Fixed
2.3	Informational	Reconstruct source code with source map	Information Disclosure	Fixed
2.4	Informational	Add liquidity transaction with 0 iDOL and 0 LIEN	Miscellaneous	Fixed

1.4. Summary of Recommendations

Below outline the major recommendations that CertiK suggests for Lien.finance to take in order to strategically lower the risk level of the application:

- Hold a Secure Development Training class to attack the root of security problems.
- Perform additional testing once fixes are in place to both validate vulnerabilities have been fixed and that the new functionality did not introduce additional vulnerabilities.
- Continue to perform periodic security assessments of the application source code, server security posture, and network architecture to ensure compliance with the corporate security policies and procedures.

The initiative of Lien.finance to perform these tests shows their appreciation and value for security. However, this is just the initial step in the continuous work to remediate and improve the security posture of the company's technology.

2. Web Application Penetration Test

CertiK performed a full penetration test on the web application. CertiK tested it against different web vulnerabilities including the OWASP Top Ten.

Listed below are the vulnerabilities that CertiK has found during the test. Details of the vulnerabilities are provided as well as thorough recommendations on how the vulnerabilities can be fixed.

Findings are arranged according to risk level.

2.1. Clickjacking

Severity: Low

Introduction

Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content on a decoy website. The impact of the clickjacking depends on the action that the user performed on the target application. Clickjacking can turn system features on and off, tricked users into downloading malware and steal sensitive information such as password and credit card number.

Description

The following screenshot shows the HTTP response of the GET request to <https://beta-livid.vercel.app>. The response doesn't contain any security headers to prevent the website from loading in an iframe.

```
HTTP/1.1 200 OK
Date: Wed, 26 Aug 2020 20:46:13 GMT
Content-Type: text/html; charset=utf-8
Connection: close
cache-control: public, max-age=0, must-revalidate
content-disposition: inline; filename="index.html"
access-control-allow-origin: *
etag: W/"6c2b869bb5fbf6e8d1ba92ed473da7b975604dc43c9b9a8034086c5651bcce7f"
x-vercel-cache: HIT
age: 751
server: Vercel
x-vercel-id: iad1::q2lmf-1598474773106-e99c79348242
strict-transport-security: max-age=63072000; includeSubDomains; preload
Content-Length: 3063
```

Location

<https://beta-livid.vercel.app>

Impact

An attacker can trick the victim into clicking on actionable content in the Lien application by loading it in a hidden iframe. The Proof of concept HTML file demonstrates an attacker can trick users to remove liquidity from the pool.

Step to Reproduce

1. Visit the website(<https://beta-livid.vercel.app>) with a browser and connect to the MetaMask wallet.
2. Save the HTML code snippet in the Proof of Concept section into a .html file.
3. Open the saved HTML file in the same browser in step 1.
4. Notice the application loaded in the iframe.

Evidence

File | /Users/minzhihe/Desktop/poc.html

Liquid Bonds

<

Proof of Concept

```
<head>
```

```
<style>
```

```
#target_website {
  position:relative;
  width:1280px;
  height:1280px;
  z-index:2;
  opacity:0.4;
}
```

```
#decoy_website {
  position:absolute;
  width:300px;
  height:400px;
  top:1200px;
  left:1100px;
  z-index:1;
}
```

```
</style>
```

```
</head>
```



```
<body>
<br>
<h1>Clickjacking POC</h1>
<br>
  <div id="decoy_website">
    <h1>Click Me</h1>
  </div>
  <iframe id="target_website" src="https://beta-livid.vercel.app">
  </iframe>
</body>
```

Recommendation

Setting "X-Frame-Options" and "Content Security Policy" headers in the HTTP response are the most effective way to protect the application against clickjacking.

With X-Frame-Options:

Deny the website loading in any iframe with the "deny" directive:

X-Frame-Options: deny

Restricted to the same origin as the website using the sameorigin directive:

X-Frame-Options: sameorigin

Restricted to a specific website using the allow-from directive:

X-Frame-Options: allow-from https://example.com

With Content Security Policy:

Deny the website loading in any iframe with the "none" directive:

Content-Security-Policy: frame-ancestors 'none'

CSP whitelists frames to the same domain only:

Content-Security-Policy: frame-ancestors 'self'

Restricted to a specific website with:

Content-Security-Policy: frame-ancestors example.com

2.2. Directory Listing

Severity: Low

Introduction

Web servers can be configured to automatically list the contents of directories that do not have an index page present. This can aid an attacker by enabling them to quickly identify the resources at a given path and proceed directly to analyzing and attacking those resources. It particularly increases the exposure of sensitive files within the directory that are not intended to be accessible to users, such as temporary files and crash dumps.

Description

Directory listing is enabled on <https://beta-livid.vercel.app/>. It leaks a complete index of all of the resources located on the server.

Location

<https://beta-livid.vercel.app/static>

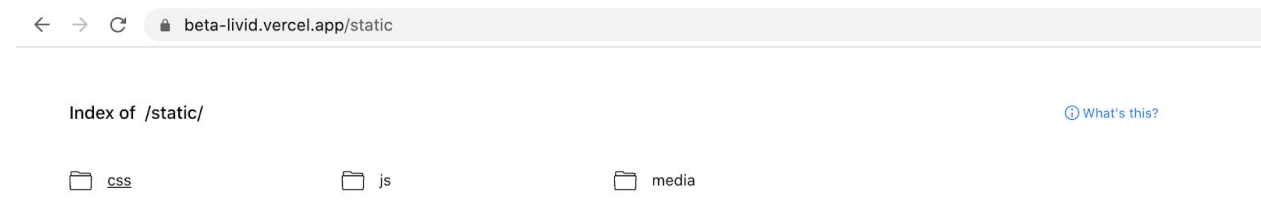
Impact

An attacker can gain important detail on folder structure and sensitive files such as data files and source code on the server. The risk of directory listing depends upon what type of directory is discovered and what types of files are contained within it.

Step to Reproduce

1. Visit <https://beta-livid.vercel.app/static> in the browser

Evidence



Recommendation

Disable directory listing on the server. For Vercel, directory listing can be disabled by adding an index file or a custom 404 page. For more information on disable directory listing on Vercel, please visit <https://vercel.com/docs/directory-listing>.

2.3. Reconstruct source code with source map

Severity: Informational

Introduction

JavaScript running in a page is often machine-generated, as when compiled from a language like CoffeeScript or TypeScript. The JavaScript sources executed by the browser are often transformed in some way from the original sources created by a developer. Sources are often combined and minified to make delivering them from the server more efficient. In these situations, it's much easier to debug the original source, rather than the source in the transformed state that the browser has downloaded. A "source map" is a file that maps from the transformed source to the original source, enabling the browser to reconstruct the original source and present the reconstructed original in the debugger.

Description

The web application front-end is written in React. In the compile-time, react converts source files into a minified JavaScript file. We were able to recover the original React source code in Chrome and Firefox with the JavaScript source and the source map file.

Location

<https://testalphaalpha.github.io/beta/static/js/2.032f8afe.chunk.js.map>

<https://testalphaalpha.github.io/beta/static/js/main.ff65d254.chunk.js.map>

<https://testalphaalpha.github.io/beta/static/css/main.311bec4d.chunk.css.map>

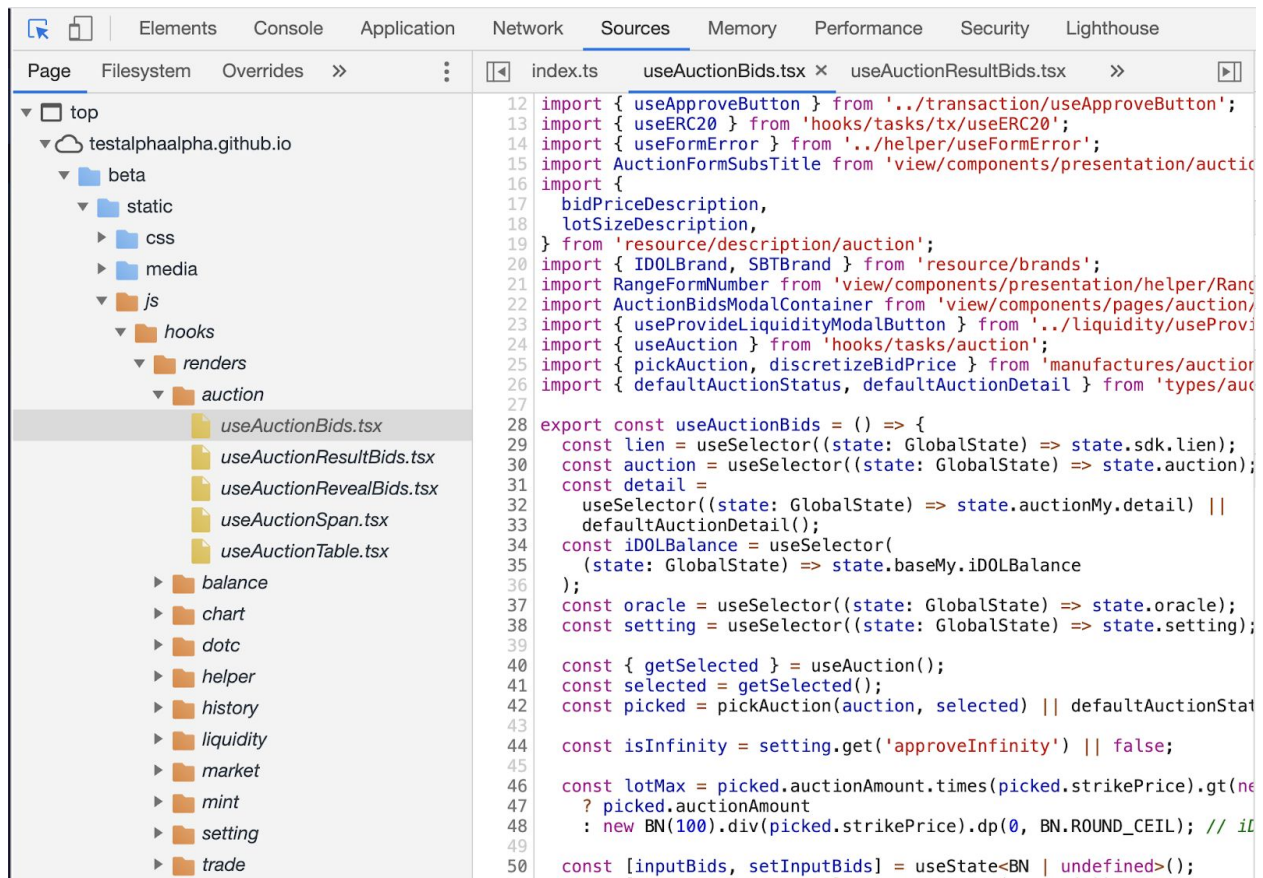
Impact

An attacker can gain important detail on code structure or potential flaws in the application by analyzing the front-end source code. This information can potentially be used by an attacker to perform more sophisticated attacks. Although the attacker can still obtain the JavaScript source without the source map, it will be much harder for them to analyze the minified version of the JavaScript code.

Step to Reproduce

1. Visit the application <https://testalphaalpha.github.io/beta> with Chrome or Firefox.
2. In Chrome, right-click the web page, select "Inspect" and navigate to the "sources" tab. In Firefox, right-click the web page, select "Inspect Element" and navigate to the "Debugger" tab.
3. The Front-end source should present under the "webpack" section.

Evidence



```

12 import { useApproveButton } from '../transaction/useApproveButton';
13 import { useERC20 } from 'hooks/tasks/tx/useERC20';
14 import { useFormError } from '../helper/useFormError';
15 import AuctionFormSubTitle from 'view/components/presentation/auction';
16 import {
17   bidPriceDescription,
18   lotSizeDescription,
19 } from 'resource/description/auction';
20 import { IDOLBrand, SBTBrand } from 'resource/brands';
21 import RangeFormNumber from 'view/components/presentation/helper/RangeFormNumber';
22 import AuctionBidsModalContainer from 'view/components/pages/auction/auctionBidsModalContainer';
23 import { useProvideLiquidityModalButton } from '../liquidity/useProvideLiquidityModalButton';
24 import { useAuction } from 'hooks/tasks/auction';
25 import { pickAuction, discretizeBidPrice } from 'manufactures/auction';
26 import { defaultAuctionStatus, defaultAuctionDetail } from 'types/auction';
27
28 export const useAuctionBids = () => {
29   const lien = useSelector((state: GlobalState) => state.sdk.lien);
30   const auction = useSelector((state: GlobalState) => state.auction);
31   const detail =
32     useSelector((state: GlobalState) => state.auctionMy.detail) ||
33     defaultAuctionDetail();
34   const iDOLBalance = useSelector(
35     (state: GlobalState) => state.baseMy.iDOLBalance
36   );
37   const oracle = useSelector((state: GlobalState) => state.oracle);
38   const setting = useSelector((state: GlobalState) => state.setting);
39
40   const { getSelected } = useAuction();
41   const selected = getSelected();
42   const picked = pickAuction(auction, selected) || defaultAuctionStatus;
43
44   const isInfinity = setting.get('approveInfinity') || false;
45
46   const lotMax = picked.auctionAmount.times(picked.strikePrice).gt(
47     ? picked.auctionAmount
48     : new BN(100).div(picked.strikePrice).dp(0, BN.ROUND_CEIL); // if
49
50   const [inputBids, setInputBids] = useState<BN | undefined>();

```

Recommendation

Reconstruct program code with the source map file makes the debugging process much easier. However, the source map file should be removed in the production environment.

Two recommend ways to remove source map are:

1. Add "GENERATE_SOURCEMAP=false" in the build script in the "package.json" file. Specifically, change line 55 in package.json to the following:
"build": "yarn run build:css &&GENERATE_SOURCEMAP=false react-scripts build"
2. Remove source map files after building the application.

Reference:

- <https://stackoverflow.com/questions/51984146/how-to-disable-source-maps-for-react-js-application>
- <https://github.com/facebook/create-react-app/issues/1341>

2.4. Add liquidity transaction with 0 iDOL and 0 LIEN

Severity: Informational

Description

The "Add Liquidity" UI in the liquidity page allows the user to submit the "Add Liquidity" transaction with 0 iDOL and 0 LIEN. The transaction will fail, but gas is consumed in the transaction.

Location

<https://beta-livid.vercel.app/liquidity>

Impact

Users might accidentally submit a "Add Liquidity" with 0 iDOL and 0 LIEN and lose the money in terms of gas fee for the transaction.

Step to Reproduce

1. Navigate to the liquidity page and connect to the MetaMask wallet.
2. Enter 0 in the "Add liquidity" interface and click the "Add Liquidity" button to initiate the transaction and confirm the transaction in MetaMask.
3. Noticed the transaction would fail and the gas fee was gone.

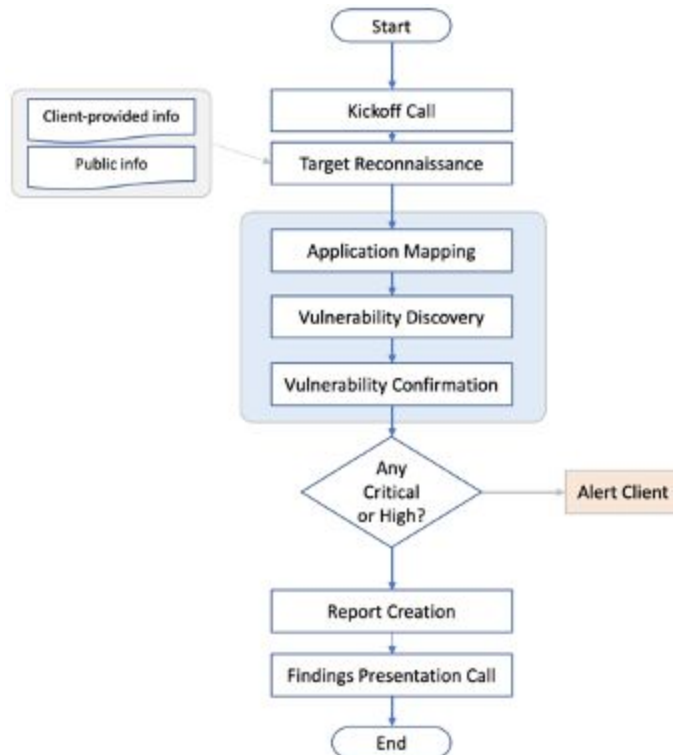
Recommendation

Disable the "Add Liquidity" button if the input is 0, similar to the UI in the "trade" and "Mint" page.

3. Appendix – Methodology

CertiK uses a comprehensive penetration testing methodology which adheres to industry best practices and standards in security assessments including from **OWASP** (Open Web Application Security Project), **NIST**, **PTES** (Penetration Testing Execution Standard).

Below is a flowchart of our assessment process:



3.1. Coverage and Prioritization

As many components as possible will be tested manually. Priority is generally based on three factors: critical security controls, sensitive data, and the likelihood of vulnerability.

Critical security controls will always receive the top priority in the test. If a vulnerability is discovered in the critical security control, the entire application is likely to be compromised, resulting in a critical-risk to the business. For most applications, critical controls will include the login page, but it could also include major workflows such as the checkout function in an online store.

The Second priority is given to application components that handle sensitive data. This is dependent on business priorities, but common examples include payment card data, financial data, or authentication credentials.

Final priority includes areas of the application that are most likely to be vulnerable. This is based on CertiK' experience with similar applications developed using the same technology or with other applications that fit the same business role. For example, large applications will often have older sections that are less likely to utilize modern security techniques.

3.2. Reconnaissance

CertiK gathers information about the target application from various sources depending on the type of test being performed. CertiK obtains whatever information that is possible and appropriate from the client during scoping and supplements it with relevant information that can be gathered from public sources. This helps provide a better overall picture and understanding of the target.

3.3. Application Mapping

CertiK examines the application, reviewing its contents, and mapping out all its functionalities and components. CertiK makes use of different tools and techniques to traverse the entire application and document all input areas and processes. Automated tools are used to scan the application and it is then manually examined for all its parameters and functionalities. With this, CertiK creates and widens the overall attack surface of the target application.

3.4. Vulnerability Discovery

Using the information that is gathered, CertiK comes up with various attack vectors to test against the application. CertiK uses a combination of automated tools and manual techniques to identify vulnerabilities and weaknesses. Industry-recognized testing tools will be used, including Burp Suite, Nikto, Metasploit, and Kali. Furthermore, any controls in place that would inhibit the successful exploitation of a particular system will be noted.

3.5. Vulnerability Confirmation

After discovering vulnerabilities in the application, CertiK validates the vulnerabilities and assesses its overall impact. To validate, CertiK performs a Proof-of-Concept of an attack on the vulnerability, simulating real world scenarios to prove the risk and overall impact of the vulnerability.

Through CertiK's knowledge and experience on attacks and exploitation techniques, CertiK is able to process all weaknesses and examine how they can be combined to compromise the application. CertiK may use different attack chains, leveraging different weaknesses to escalate and gain a more significant compromise.

To minimize any potential negative impact, vulnerability exploitation was only attempted when it would not adversely affect production applications and systems, and then only to confirm the presence of a specific vulnerability. Any attack with the potential to cause system downtime or seriously impact business continuity was not performed. Vulnerabilities were never exploited to delete or modify data; only read-level access was attempted. If it appeared possible to modify data, this was noted in the list of vulnerabilities below.

3.6. Immediate escalation of High or Critical Findings

If critical or high findings are found whereby application elements are compromised, client's key security contacts will be notified immediately.

3.7. Vulnerability Classes

Below is the list of common vulnerability classes to be tested.

Security Misconfiguration	<ul style="list-style-type: none">• Missing Security Headers• Debugging Enabled
Information Disclosure	<ul style="list-style-type: none">• Directory Indexing• Verbose Error Messages• HTML Comments• Default Content
Account Policy	<ul style="list-style-type: none">• Default / Weak Passwords• Unlimited Login Attempts• Password Reset• Insufficient Session Expiration
Session Management	<ul style="list-style-type: none">• Session Identifier Prediction• Session Hijacking• Session Replay• Session Fixation/Trapping• Cross-Site Request Forgery
Injection	<ul style="list-style-type: none">• SQL Injection• Cross-Site Scripting• LDAP Injection• HTML Injection• XML Injection• OS Command Injection
Broken Access Control	<ul style="list-style-type: none">• Authentication Bypass• Authorization Bypass• Privilege Escalation
Application Resource Handling	<ul style="list-style-type: none">• Path Traversal• Predictable Object Identifiers• XML External Entity Expansion• Local & Remote File Inclusion
Logic Flaws	<ul style="list-style-type: none">• Abuse of Functionality• Workflow Bypass

Insufficient Cryptography	<ul style="list-style-type: none">• Weak Hashing Algorithms• Weak Encryption Algorithms• Hard Coded Cryptographic Key
Denial of Service	<ul style="list-style-type: none">• Server-side Denial of service• Client-side Denial of service

3.8. Risk Assessment

The following risk levels categorize the risk level of issues presented in the report:

Risk Level	CVS Score	Impact	Exploitability
Critical	9.0-10.0	Root-level or full-system compromise, large-scale data breach	Trivial and straightforward
High	7.0-8.9	Elevated privilege access, significant data loss or downtime	Easy, vulnerability details or exploit code are publicly available, but may need additional attack vectors (e.g., social engineering)
Medium	4.0-6.9	Limited access but can still cause loss of tangible assets, which may violate, harm, or impede the org's mission, reputation, or interests.	Difficult, requires a skilled attacker, needs additional attack vectors, attacker must reside on the same network, requires user privileges
Low	0.1-3.9	Very little impact on an org's business	Extremely difficult, requires local or physical system access
Informational	0.0	Discloses information that may be of interest to an attacker.	Not exploitable but rather is a weakness that may be useful to an attacker should a higher risk issue be found that allows for a system exploit

4. Why CertiK

CertiK is a blockchain cybersecurity company with a global headquarter in New York City and presence in Beijing, Seattle, Seoul and Tokyo, pioneering the use of cutting-edge technologies, including static/dynamic analysis, Formal Verification, and Penetration Testing.

CertiK has received grants from IBM, the Qtum Foundation, and the Ethereum Foundation to support its research of improving security across the blockchain industry. CertiK also contributes to the technical communities and ecosystems by providing guidance, research, and advisory about blockchain and smart contract best practices.

Our Penetration Testing service envisions to empower individuals and businesses to thrive in the new digital security age, especially in the blockchain space.