Blockstack Desktop Wallet

Penestration Test

November 12th, 2020

For :
Mark Hendrickson @ Blockstack

By :
Peiyu Wang @ CertiK
peiyu.wang@certik.org

Minzhi He @ CertiK
minzhi.he@certik.org

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contained in this document is provided in confidence for the sole purpose of adjudication of the document and shall not be published or disclosed wholly or in part to any other party without CertiK's prior permission in writing and shall be held in safe custody.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.

Overview

Scope

At the start of the engagement, CertiK worked with Blockstack 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 public GitHub repository.

Application Name	Blockstack Desktop Wallet
Codebase	GitHub Repository
Commit hash	9d2a2835777a649a14bdcd1bc36884f1bcc128dd

Audit Summary

Delivery Date	Nov. 12, 2020
Method of Audit	Dynamic Testing, Static Code Review
Consultants Engaged	2
Timeline	Nov. 04, 2020 - Nov. 11, 2020

Vulnerability Summary

Total Issues	1
Total Low	1

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.

Executive Summary

Blockstack engaged CertiK to perform an application penetration test for their desktop wallet. The main objective of the engagement is to verify whether possible means for compromising the security of the users' wallets, identify its weaknesses and provide recommendations to fix and improve its overall security posture. By auditing sources, CertiK checked for implicit and more hidden ways for determined attackers seeking to break the BlockStack compound's integrity. A special emphasis was also placed on the integration of the Electron framework and its hardening.

After a thorough review of the application, CertiK identified one low severity security issue. We were able to crash the application by configuring the wallet to connect to a defective node we set up. Given the severity of the vulnerabilities on the application, it is unlikely that a remote attacker can directly compromise the user wallet.

Even though Electron applications tend to be prone to "Critical"-level pitfalls, the Blockstack wallet managed to avoid these and remain secure. The security posture of the project was judged as robust, and important components such as wallet secret management were implemented with security in mind.

Findings

ID	Title	Severity	Vulnerability Class	Status
BLS-01	Client-side denial of service with defective node	Low	Denial of service	Open

BLS-01: Client-side denial of service with defective node

Severity: Low

Description

The application allows the user to add a custom node in the settings interface. The node will be used by the application to communicate with the blockchain. Users are always under the threat of connecting to a malicious node with a phishing attack from the attacker. During the test, we were interested in what can a malicious node do to the wallet application. We found that a malicious can cause the application to crash.

Location

The "Settings" interface in the desktop wallet.

Impact

A malicious node can cause the application to crash, which negatively affects the user experience. Currently, the only way for the user to recover the wallet is to manually delete the config.json file under the "/Users/Username/Library/Application Support/Stacks Wallet" directly. Restart or re-install the application can't resolve the issue.

Step to Reproduce

1. Setup a defective node with the code snippet in the proof of concept section. The code contains a simple flask application.
2. Configure the wallet to connect to the defective node in the settings interface.
3. Notice the wallet crash after navigating to the send/receive interface.

Proof-of-Concept

```
from flask import Flask
import json
app = Flask(__name__)

@app.route('/<path:a>',methods=["GET","POST"])
def dos_handler(a=None,b=None):
    return json.dumps({"status":"ready"})

@app.route('/extended/v1/status',methods=["GET","POST"])
def pass_the_check():
    return json.dumps({"status":"ready"})
```

Command to run the flask app:

```
export FLASK_APP=app.py
flask run --host=0.0.0.0
```

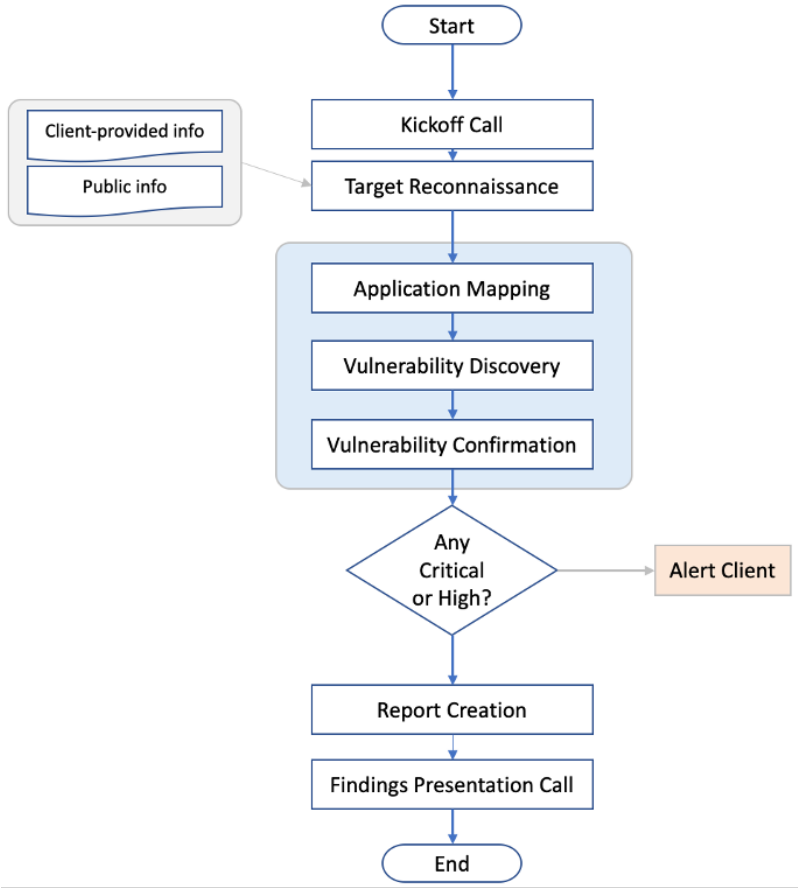
Recommendation:

It's impossible to eliminate the situation that a user connects to a malicious blockchain node if the application allows users to add custom code. The app should allow the user to change node when the node does not function correctly, instead of crash completely.

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:



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.

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.

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.

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.

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' 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.

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.

Vulnerability Classes

Security Misconfiguration	• Missing Security Headers • Debugging Enabled
Information Disclosure	• Directory Indexing • Verbose Error Messages • HTML Comments • Default Content
Account Policy	• Default / Weak Passwords • Unlimited Login Attempts • Password Reset • Insufficient Session Expiration
Session Management	• Session Identifier Prediction • Session Hijacking • Cross-Site Request Forgery • Insufficient Session Expiration
Injection	• SQL Injection • Cross-Site Scripting • LDAP Injection • HTML Injection • XML Injection • OS Command Injection
Broken Access Control	• Authentication Bypass • Authorization Bypass • Privilege Escalation
Application Resource Handling	• Path Traversal • Predictable Object Identifiers • XML External Entity Expansion • Local & Remote File Inclusion
Logic Flaws	• Abuse of Functionality • Workflow Bypass
Insufficient Cryptography	• Weak Hashing Algorithms • Weak Encryption Algorithms • Hard Coded Cryptographic Key
Denial of Service	• Server-side Denial of service • Client-side Denial of service

Risk Assessment

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

Risk Level	CVSS 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