Penetration Testing Report for Demystify -

Metamask Snap Application

Testers:

1. Or Duan
2. Avigdor Sason Cohen
3. Ido Shdaimah

Table of Contents

Management Summary

Demystify contacted Sayfer Security in order to perform penetration testing on their MetaMask Snap in July 2023. It involves a process of probing and exploiting security vulnerabilities in the Snap to assess its resilience against malicious activities. The test aims to identify weaknesses in the Snap's code, configuration, or integrations, ensuring the security of user assets and data.

Before assessing the above services, we held a kickoff meeting with the Demystify technical team and received an overview of the system and the goals for this research.

During the audit we discovered 3 potential vulnerabilities which were addressed by the Demystify team.

Demystify Metamask snap has passed this security audit and we can attest that the security system is competent.

During the audit, we evaluated the security of Demystify Snap using the OWASP WSTG standard. In total, we ran 97 test scenarios in 12 different domains. Demystify Snap scored 97 out of 97.

Risk Methodology

At Sayfer, we are committed to delivering the highest quality penetration testing to our clients. That's why we have implemented a comprehensive risk assessment model to evaluate the severity of our findings and provide our clients with the best possible recommendations for mitigation.

Our risk assessment model is based on two key factors: IMPACT and LIKELIHOOD. Impact refers to the potential harm that could result from an issue, such as financial loss, reputational damage, or a non-operational system. Likelihood refers to the probability that an issue will occur, taking into account factors such as the complexity of the attack and the number of potential attackers.

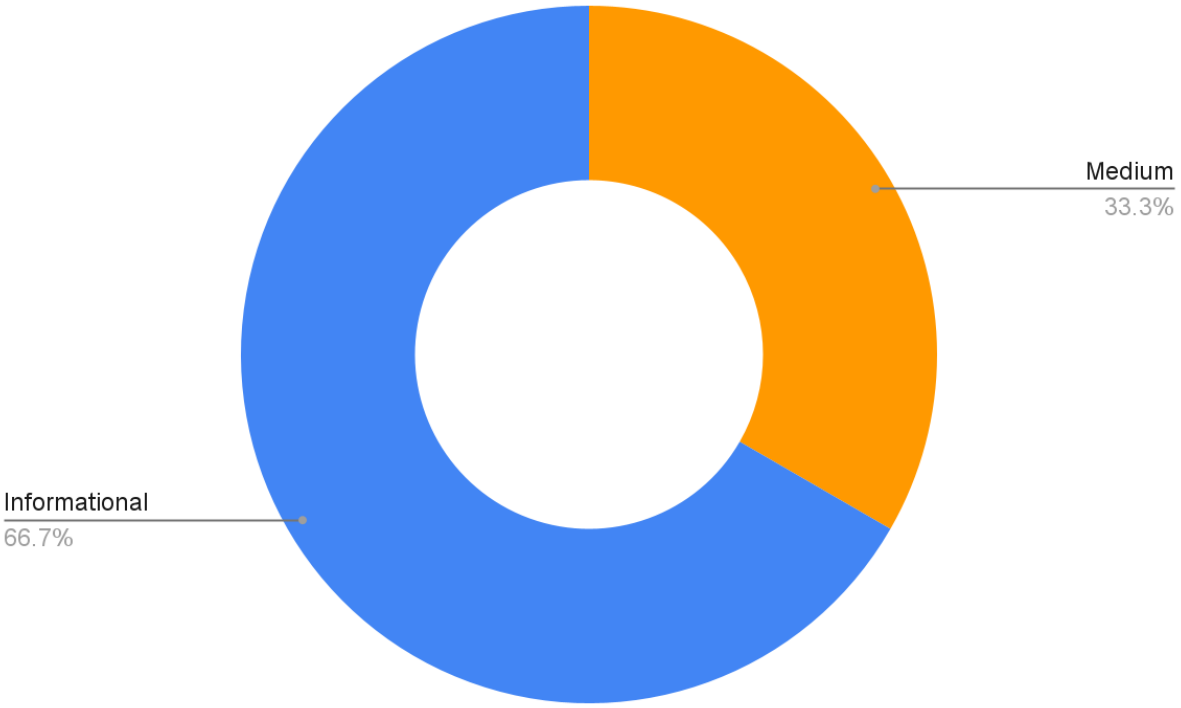
By combining these two factors, we can create a comprehensive understanding of the risk posed by a particular issue and provide our clients with a clear and actionable assessment of the severity of the issue. This approach allows us to prioritize our recommendations and ensure that our clients receive the best possible advice on how to protect their business.

Risk is defined as follows:

Overall Risk Security

IMPACT ↓ / LIKELIHOOD →	LOW	MEDIUM	HIGH
HIGH	Medium	High	High
MEDIUM	Low	Medium	High
LOW	Informational	Low	Medium

Vulnerabilities by Risk



Risk	Low	Medium	High	Informational
# of issues	0	1	0	2

- **Low** – No direct threat exists. The vulnerability may be exploited using other vulnerabilities.
- **Medium** – Indirect threat to key business processes or partial threat to business processes.
- **High** – Direct threat to key business processes.
- **Informational** – This finding does not indicate vulnerability, but states a comment that notifies about design flaws and improper implementation that might cause a problem in the long run.

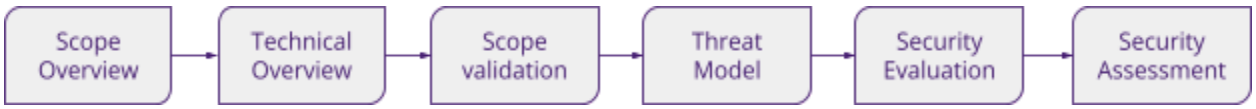
Approach

Introduction

Demystify contacted Sayfer to perform penetration testing on their MetaMask Snap application.

This report documents the research carried out by Sayfer targeting the selected resources defined under the research scope. Particularly, this report displays the security posture review for the Demystify MetaMask Snap application and its surrounding infrastructure and process implementations.

Our penetration testing project life cycle:



Scope Overview

During our first meeting and after understanding the company's needs, we defined the application's scope that resides at the following URLs as the scope of the project:

- Demystify's MetaMask Snap
  - **Audit commit:** eb5fd94e5135fe2ddd74d0141795ebdbfd506e27
  - **Fixes commit:** d029acbe2c657ae4c9c2318ee765096107de82c6

Our tests were performed from July to August 2023

Scope Validation

We began by ensuring that the scope defined to us by the client was technically logical.
Deciding what scope is right for a given system is part of the initial discussion. Getting the scope right is key to deriving maximum business value from the research.

Threat Model

During our kickoff meetings with the client we defined the most important assets the application possesses.
We defined that the largest current threat to the system is sensitive user information.

Security Evaluation Methodology

Sayfer uses OWASP WSTG as our technical standard when reviewing web applications. After gaining a thorough understanding of the system we decided which OWASP tests are required to evaluate the system.

Security Assessment

After understanding and defining the scope, performing threat modeling, and evaluating the correct tests required in order to fully check the application for security flaws, we performed our security assessment.

Issue Table Description

Issue title

ID	The OWASP ID of the issue. Additional tests that we conduct and are not included in the OWASP table will have Sayfer ID. Example ID: **WSTG-INFO-002** **WSTG** - Web Security Test Guide. **INFO** - A shorthand for the topic to which the issue belongs. **002** - Issue number.
Risk	Represents the risk factor of the issue. For further description refer to the Vulnerabilities by Risk section.
Required Skill	Describes the skill level required to conduct successful exploitation. The lower the skill level the easier the exploitation process.
OWASP Reference	A link to the relevant OWASP page for further knowledge.
Location	The URL in which this issue was detected. Issues with no location have no particular location and refer to the product as a whole.
Tools	The tools used to detect the issue.
Description	Here we provide a brief description of the issue and how it formed, the steps we made to find or exploit it, along with proof of concept (if present), and how this issue can affect the product or its users.
Mitigation	Suggested resolving options for this issue and links to advised sites for further remediation.

Security Evaluation

The following tests were conducted while auditing the system

Information Gathering	Test Name	Status
WSTG-INFO-01	Conduct Search Engine Discovery Reconnaissance for Information Leakage	Pass
WSTG-INFO-02	Fingerprint Web Server	Pass
WSTG-INFO-03	Review Webserver Metafiles for Information Leakage	Pass
WSTG-INFO-04	Enumerate Applications on Webserver	Pass
WSTG-INFO-05	Review Webpage Content for Information Leakage	Pass
WSTG-INFO-06	Identify application entry points	Pass
WSTG-INFO-07	Map execution paths through application	Pass
WSTG-INFO-08	Fingerprint Web Application Framework	Pass
WSTG-INFO-09	Fingerprint Web Application	Pass
WSTG-INFO-10	Map Application Architecture	Pass

Configuration and Deploy Management Testing	Test Name	Status
WSTG-CONF-01	Test Network Infrastructure Configuration	Pass
WSTG-CONF-02	Test Application Platform Configuration	Pass
WSTG-CONF-03	Test File Extensions Handling for Sensitive Information	Pass
WSTG-CONF-04	Review Old Backup and Unreferenced Files for Sensitive Information	Pass
WSTG-CONF-05	Enumerate Infrastructure and Application Admin Interfaces	Pass
WSTG-CONF-06	Test HTTP Methods	Pass
WSTG-CONF-07	Test HTTP Strict Transport Security	Pass
WSTG-CONF-08	Test RIA cross domain policy	Pass
WSTG-CONF-09	Test File Permission	Pass
WSTG-CONF-10	Test for Subdomain Takeover	Pass
WSTG-CONF-11	Test Cloud Storage	Pass

Identity Management Testing	Test Name	Status

WSTG-IDNT-01	Test Role Definitions	Pass
WSTG-IDNT-02	Test User Registration Process	Pass
WSTG-IDNT-03	Test Account Provisioning Process	Pass
WSTG-IDNT-04	Testing for Account Enumeration and Guessable User Account	Pass
WSTG-IDNT-05	Testing for Weak or unenforced username policy	Pass

Authentication Testing	Test Name	Status
WSTG-ATHN-01	Testing for Credentials Transported over an Encrypted Channel	Pass
WSTG-ATHN-02	Testing for Default Credentials	Pass
WSTG-ATHN-03	Testing for Weak Lock Out Mechanism	Pass
WSTG-ATHN-04	Testing for Bypassing Authentication Schema	Pass
WSTG-ATHN-05	Testing for Vulnerable Remember Password	Pass
WSTG-ATHN-06	Testing for Browser Cache Weaknesses	Pass
WSTG-ATHN-07	Testing for Weak Password Policy	Pass
WSTG-ATHN-08	Testing for Weak Security Question Answer	Pass
WSTG-ATHN-09	Testing for Weak Password Change or Reset Functionalities	Pass
WSTG-ATHN-10	Testing for Weaker Authentication in Alternative Channel	Pass

Authorization Testing	Test Name	Status
WSTG-ATHZ-01	Testing Directory Traversal File Include	Pass
WSTG-ATHZ-02	Testing for Bypassing Authorization Schema	Pass
WSTG-ATHZ-03	Testing for Privilege Escalation	Pass
WSTG-ATHZ-04	Testing for Insecure Direct Object References	Pass

Session Management Testing	Test Name	Status
WSTG-SESS-01	Testing for Session Management Schema	Pass
WSTG-SESS-02	Testing for Cookies Attributes	Pass
WSTG-SESS-03	Testing for Session Fixation	Pass
WSTG-SESS-04	Testing for Exposed Session Variables	Pass
WSTG-SESS-05	Testing for Cross Site Request Forgery	Pass
WSTG-SESS-06	Testing for Logout Functionality	Pass
WSTG-SESS-07	Testing Session Timeout	Pass
WSTG-SESS-08	Testing for Session Puzzling	Pass

WSTG-SESS-09	Testing for Session Hijacking	Pass

Data Validation Testing	Test Name	Status
WSTG-INPV-01	Testing for Reflected Cross Site Scripting	Pass
WSTG-INPV-02	Testing for Stored Cross Site Scripting	Pass
WSTG-INPV-03	Testing for HTTP Verb Tampering	Pass
WSTG-INPV-04	Testing for HTTP Parameter Pollution	Pass
WSTG-INPV-05	Testing for SQL Injection	Pass
WSTG-INPV-06	Testing for LDAP Injection	Pass
WSTG-INPV-07	Testing for XML Injection	Pass
WSTG-INPV-08	Testing for SSI Injection	Pass
WSTG-INPV-09	Testing for XPath Injection	Pass
WSTG-INPV-10	Testing for IMAP SMTP Injection	Pass
WSTG-INPV-11	Testing for Code Injection	Pass
WSTG-INPV-12	Testing for Command Injection	Pass
WSTG-INPV-13	Testing for Format String Injection	Pass
WSTG-INPV-14	Testing for Incubated Vulnerability	Pass
WSTG-INPV-15	Testing for HTTP Splitting Smuggling	Pass
WSTG-INPV-16	Testing for HTTP Incoming Requests	Pass
WSTG-INPV-17	Testing for Host Header Injection	Pass
WSTG-INPV-18	Testing for Server-side Template Injection	Pass
WSTG-INPV-19	Testing for Server-Side Request Forgery	Pass

Error Handling	Test Name	Status
WSTG-ERRH-01	Testing for Improper Error Handling	Pass
WSTG-ERRH-02	Testing for Stack Traces	Pass

Cryptography	Test Name	Status
WSTG-CRYP-01	Testing for Weak Transport Layer Security	Pass
WSTG-CRYP-02	Testing for Padding Oracle	Pass
WSTG-CRYP-03	Testing for Sensitive Information Sent via Unencrypted Channels	Pass
WSTG-CRYP-04	Testing for Weak Encryption	Pass

Business logic Testing	Test Name	Status
WSTG-BUSL-01	Test Business Logic Data Validation	Pass

WSTG-BUSL-02	Test Ability to Forge Requests	Pass
WSTG-BUSL-03	Test Integrity Checks	Pass
WSTG-BUSL-04	Test for Process Timing	Pass
WSTG-BUSL-05	Test Number of Times a Function Can be Used Limits	Pass
WSTG-BUSL-06	Testing for the Circumvention of Work Flows	Pass
WSTG-BUSL-07	Test Defenses Against Application Mis-use	Pass
WSTG-BUSL-08	Test Upload of Unexpected File Types	Pass
WSTG-BUSL-09	Test Upload of Malicious Files	Pass

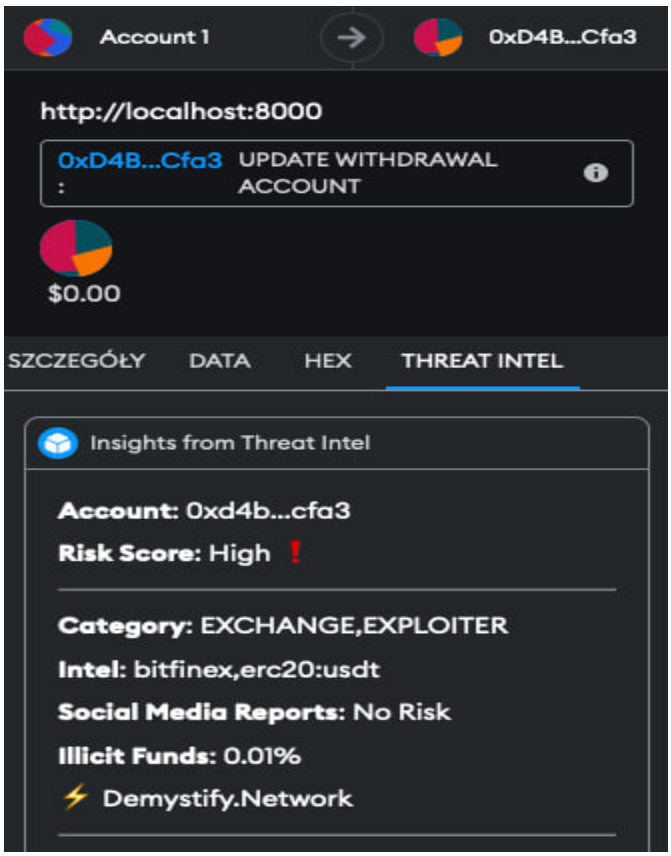
Client Side Testing	Test Name	Status
WSTG-CLNT-01	Testing for DOM-Based Cross Site Scripting	Pass
WSTG-CLNT-02	Testing for JavaScript Execution	Pass
WSTG-CLNT-03	Testing for HTML Injection	Pass
WSTG-CLNT-04	Testing for Client Side URL Redirect	Pass
WSTG-CLNT-05	Testing for CSS Injection	Pass
WSTG-CLNT-06	Testing for Client Side Resource Manipulation	Pass
WSTG-CLNT-07	Test Cross Origin Resource Sharing	Pass
WSTG-CLNT-08	Testing for Cross Site Flashing	Pass
WSTG-CLNT-09	Testing for Clickjacking	Pass
WSTG-CLNT-10	Testing WebSockets	Pass
WSTG-CLNT-11	Test Web Messaging	Pass
WSTG-CLNT-12	Testing Browser Storage	Pass
WSTG-CLNT-13	Testing for Cross Site Script Inclusion	Pass

API Testing	Test Name	Status
WSTG-APIT-01	Testing GraphQL	Pass

Security Assessment Findings

Incorrect Risk Score Calculation

ID	SAY-01
Status	Fixed
Risk	Medium
Business Impact	The impact depends on the end user's behavior.
Location	—
Description	The demystify.network API returns a riskScore based on an internal, implicit classification. While this works fine for known malicious contracts, it doesn't necessarily work for known but secure contracts. Probably due to their communication with the malicious ones (which is logical since the example concerns USDT), the snap returns the highest possible Risk Score in relation to the Tether contract address, i.e. USDT. • We sent a request with the USDT contract:  • This leads to the following warning in metamask.



Mitigation	The risk calculation method for such contracts should be improved over time when the algorithm gets better.

Usage of Encoded Hardcoded Values Without Description

ID	SAY-02
Status	Acknowledged. Demystify: This is by design. Since it's a public repository and bots can copy a URL and launch attacks, it's less obvious to encrypt the URL.
Risk	Informational
Business Impact	The need to decode the content of the TARGET variable when reading the code may cause a reduction in readability. There is no direct security impact so we rated this finding as informational.
Location	packages/snap/src/insights.ts:20-21
Description	The Snap uses a base64-encoded endpoint for communication without any comment about what's inside. • The relevant constant: `const TARGET = 'aHR0cHM6Ly9hcGkuZGVteXN0aWZ5Lm5ldHdvcmsvYWRkcmVzcy90aHJlYXRJbnRlbA==';`
Mitigation	Add a comment detailing the contents of the encoded variable.

Usage of Confusing “Magic Numbers”

ID	SAY-03
Status	Fixed
Risk	Informational
Business Impact	This is a simple readability issue, hence the informational risk rating.
Location	packages/snap/src/index.ts:42-43

Description	Snap is communicating with demystify.network API sending information regarding address, with which client’s account is trying to interact. Then, the API responds with data regarding the potential risks associated with that address, and if it is somehow malicious, *percentTransactionByRisk*, a table with three entries, is returned. However, only the third ([2]) element is used in the code. It may be difficult for the reader to deduct why the code is using only that part of the response, leaving the first two untouched.
	
	• As you can see, only the second element is used:
	`const { percentTransactionByRisk } = insights;`
	`let highRisk = 'n/a';`
	
	`if (percentTransactionByRisk !== undefined) {`
	`highRisk = percentTransactionByRisk[2];`
	`}`

Mitigation	Leave a short comment explaining this oddity.

Appendix A: Security Evaluation Fixes

After a review by the Sayfer team, we certify that all the above-mentioned security issues have been addressed by the Demystify team.
Fixes commit: d029acbe2c657ae4c9c2318ee765096107de82c6