CYREX

Penetration Test Report

Prepared for: **Roby Weir, COO, JigStack**
Prepared by: **Tim De Wachter, CTO**, *Cyrex Ltd*

14/06/2021

Table of Contents

Executive summary

Cyrex was contracted by JigStack to conduct a penetration test to determine its exposure to a targeted attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attack against the scope with the goals of:

- Identifying if a remote attacker could penetrate the scope its defences.
- Determining the impact and possibility of a security breach.

Efforts were placed on the identification and exploitation of security weaknesses that could allow a remote attacker to gain unauthorized access to organizational data. The attacks were conducted with all levels of access that a general internet user would have.

As JigStack provided Cyrex with source code concerning the scope, we can label this kind of test as a white box penetration test. Cyrex was granted access to the application with minimal user privileges.

What follows is a detailed explanation of all the different vulnerabilities, advised patches and an accompanying risk analysis. We are confident that the penetration test and this report helps the customer to raise its security regarding the client application to a higher level.

Conclusion

The overall security maturity of the tested scope seems sufficient at this stage. No vulnerabilities were discovered during the audit of the contracts and we can clearly notice that they have been designed with security in mind and only contain the basic and necessary functionality for them to operate in a correct manner.

Security best practices have been implemented in different parts of the contract. This both on a code level (changing state before transferring ETH, usage of SafeMath library, applying the necessary ReentrancyGuards, etc) as on the design level.

The implemented modifiers (`onlyWhitelisted`, `onlyNonBlacklisted`) were not noticed to have unwanted side-effects. Only functions that need to be public are not onlyOwner. Such functions operate without modifying the internal state of the contract. The gradual release formula appears to be sound and safe enough in terms of math.

Defensive checks have been written throughout the contract. First and foremost, all inputs are validated using the necessary require-statements. Next to this the result of an ETH transfer-operation is always checked and corresponding exceptions are raised in case of failure.

We are confident that we tested the whole scope and all functionalities that were available to us as in-depth as possible.

We want to thank JigStack for putting trust in our knowhow and expertise concerning ethical hacking.