Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report CASHLINK SC Audit 04.2019

Cure53, Dr.-Ing. M. Heiderich, Prof. N. Kobeissi, BSc. F. Fäßler

Index

Introduction

"The creation of the shares - also called token - is quite simply done directly via our platform. The token are legally based on virtual stock options, which many companies worldwide already use for employee stock options. This legal basis has the advantage for you that your investors do not become shareholders and therefore do not end up in the cap table. We provide you with all legal documents and accompany you through the entire process."

From https://cashlink.de/en/start-ups/

This report documents the findings of a security-centered audit of a Solidity smart contract created and maintained by CASHLINK. The project was carried out by Cure53 in April 2019.

To comment on the process and resources, it should be stated that Cure53 was tasked with auditing the CASHLINK smart contract with a special focus placed on its security-critical elements like signatures and upgradeability. Three members of the Cure53 were involved in completing this project and invested a total of seven days into investigating the scope. In preparation for the assessment, CASHLINK provided the smart contract with its comprehensive security documentation. The latter entailed a scope document in which both the contract and its security goals were outlined. This way, Cure53 could fully comprehend the scope and tested accordingly.

All work was performed from remote. A dedicated Slack channel was used by Cure53 for communications with the CASHLINK team. The testing team was able to use Slack and ask questions when exploring the scope further. All exchanges were fluent and, combined with the excellent preparations before this test, the available time could be spent very efficiently. Full coverage has been ultimately reached.

As for the results, Cure53 only managed to spot three general weaknesses, all with marginal, "*Informational*" severities. This means that no actual vulnerabilities were spotted by the audit team, having the contract's design and implementation appearing very strong from a security standpoint.

In the following sections, this report will first shed light on the scope and then furnishes case-by-case descriptions of the findings. Based on the results of this spring 2019 assessment, Cure53 issues a broader verdict about the privacy and security posture of the examined CASHLINK smart contract. Conclusions are supplied in the final section of this document.

Scope

- **CASHLINK Smart Contract**
  - A very detailed description of the scope was made available to Cure53.
    - https://docs.google.com/document/d/1ICpKSpcBB3nN7dpmaWA8yPQUDuCLVV9xg42prYW9bpU/edit?usp=sharing
  - Sources for all involved components were made available for Cure53.

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while vulnerability is present, an exploit might not always be possible.

CSH-01-001 SCA: Uninitialized *signatureVersion* (Info)

The *signatureVersion* in the *SignatureManager* is not initialized but is used in the *isValidVersion* function. While this has no direct security implications, clarity of the code could be improved by explicitly setting this.

Affected File:
/contracts/SignatureManager.sol

Affected Code:
```
contract SignatureManager {
    uint256 public signatureVersion;
// [...]
function isValidVersion(uint256 version) private view returns (bool) {
    return version >= signatureVersion;
}
```

Because it can be expected that the function is initialized with 0, it is not an issue. However, it is recommended to explicitly initialize the value to improve the code's clarity.

CSH-01-002 SCA: Signature *nonceType* could be misused *(Info)*

The signature implementation supports a *nonceType* 0 or 1. The function that consumes the nonce decides - based on the *nonceType* - if the nonce will be remembered or not. This means that for the *nonceType* 0 the signature can be replayed. Because this is not the intended usage and does not apply to the current usage presented in the furnished code, the issue is tracked and treated as "*Miscellaneous*" and "*Informational*".

Affected File:
/contracts/SignatureManager.sol

Affected Code:
```
function consumeNonce(uint256 nonce) private {
    require(nonceStatus[nonce] == NonceStatus.Unused, "Nonce must be unused");
    if ( nonce & 0xff == 0) {
        return;
    }
    require((nonce & 0xff) == 1);

    // TODO how expensive is yielding this event
    emit NonceUsed(nonce);
    nonceStatus[nonce] = NonceStatus.Used;
}
```

While there is no immediate risk here, it is possible that, in the future, somebody could misuse the *nonceType* for signatures, leading to a replay attack. Thus, it is recommended to either explicitly warn about this in the integration documentation, or add a new modifier that enforces *nonceType* 1 on all critical functions.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

CSH-01-003 SCA: Unnecessary & potentially dangerous *unrevocation* logic (*Info*)

The audited CASHLINK smart contract contains a so-called "*unrevocation*" logic which can be used to mark revoked nonces as "not revoked". The reason stated for the inclusion of this code is claimed as attempting "completeness" However, a "complete" implementation of a nonce revocation would be the one with nonces that cannot be "unrevoked" later.

Affected file:
contracts/SignatureManager.sol

Affected code:
```
function unrevokeNonceImpl(uint256 nonce) internal {
	require(nonceStatus[nonce] == NonceStatus.Revoked, "Nonce must be 
revoked");
	emit NonceUnrevoked(nonce);
	nonceStatus[nonce] = NonceStatus.Unused;
}
```

The code is not called from anywhere in the smart contract. However, given its lack of usage, it is still recommended to remove it. This can extend security against various low-level Solidity bugs or prevent any potential misuse in case the CASHLINK stack is maintained by an independent developer.

Conclusions

Cure53 is happy to report that the investigated CASHLINK smart contract implementation is very robust and benefits from various best practices in the security realm. After spending seven days on the scope in April 2019, three members of the Cure53 team were no able to identify any vulnerabilities affecting the CASHLINK smart contract. Only three minor issues without security impact (i.e. with *Informational* ratings) were unveiled. Despite some slightly risky design decisions taken by the CASHLINK team, the tested smart contract makes a very strong impression.

On the latter, the CASHLINK contract's possible flaws come from the decisions around the ability to upgrade contracts and the implementation of the signature. Because of its complexity, it is important to have a clear specification and code design, which are already found on CASHLINK. Upgrade mechanisms similar to the one used by CASHLINK can generally signify risks but this one was implemented carefully. Well-defined *migration* functions made it secure. Further, while there are some reentrancy anti-patterns contained in the code, they all refer to trusted contracts and make this part seemingly safe as well. Cure53 has also examined the basic access-control and found it altogether secure.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

While the upgrade logic follows industry standards, the signature logic had some unusual markings that Cure53 needed to point out. These specifically are some design elements that were implemented but apparently not used anywhere. One example concerned "*unrevocation*" for nonces as seen in CSH-01-003, while the other addresses nonce-types that could mark a nonce for being used more than once (see CSH-01-002). Cure53 strongly recommends hardening the signature implementation to remove these useless and potentially dangerous functionalities.

In essence, Cure53 attests that the provided documentation shows that CASHLINK has a deep understanding of the implementation they manage. The testing team did not manage to find any actual security issues despite a thorough code review with full coverage. The CASHLINK smart contract stack is very well-documented and the design seems to be coherent and strong. In light of this April 2019 audit's findings, Cure53 can recommend moving forward with the project into a production stage

Cure53 would like to thank Niklas Baumstark from the CASHLINK Technologies GmbH team for his excellent project coordination, support and assistance, both before and during this assignment.