Reported 2018-12-21 Accepted Important, but out of scope

bitshares
HacktheDex

Report 20181221A

Type

Private Key Leakage in 3rd Party Java Implementation

Description

It came to our (HackTheDex Team) attention that a 3rd party Java implementation of the Elliptic Curve Digital Signature Algorithm (ECDSA) has had a major flaw.

This 3rd party library implemented serialization and signing of transactions for BitShares in a non-standard (RFC6979 compliant [3]) way, resulting in the vulnerability.

Users of the BitShares reference software provided within the bitshares-ui [6] and bitshares-core [7] repositories are **not** impacted.

Root Cause
In ECDSA, it is crucial to **never re-use** a nonce `k` as it exposes the private key. From Wikipedia [1]

> Another way ECDSA signature may leak private keys is when k is generated by a faulty random number generator.

The affected Java library has a fix [0] that shows a part of the nature of the problem. In ECDSA, a cryptographically secure random number generator must be used to obtain a new `k` for ever signature.

Previous versions of the library used the private key as seed for a pseudo random number generator. Those always produce the same outcome for a given seed. As a consequence, the random number generator produced deterministic many `k` which allows to recover the private key from two signatures with similar `k`.

Reported 2018-12-21 | Accepted | Important, but out of scope

Review

OWASP Rating	
Likelihood	Low
Impact	High
Severity	High

Possible attack vector: Recovery of the Private Key

Reviewers: Fabian Schuh, Stefan Schiessl, John Jones, Peter Conrad, abit

Resolution

First and foremost: Users of the BitShares reference software provided within the bitshares-ui [6] and bitshares-core [7] repositories are **not** impacted.

The root cause [1] has since been fixed in the original repository and we expect the developers to inform their user base to make a security upgrade.

Even though the root cause of this issue has been fixed, many more developers have forked their wallet from the affected repository and have yet to fix their app and inform their user base. We urge every user of an Android App connecting to the BitShares Blockchain to contact its developers and confirm the safety of their App.

A potential fix [0] uses a secure random number to produce `k`. However, this implementation is not secure on old Android devices [2]. An alternative approach using the RFC6979 [3] has been implemented here [4].

Proactive Mitigation
A 3rd party named `random-k.com` is scanning and securing impacted accounts. At this point in time, it appears they are white-hackers and have no intention to steal funds but secure the accounts with compromised keys.

Affected accounts can be identified by the active/owner permission of the account being replaced by an authority called `random-k.com`. Their website [5] indicates how to reach out to them.

Reward Consensus

Reward	0 bitUSD

Reported 2018-12-21	Accepted	Important, but out of scope

References

- [0] https://github.com/bituniverse/bitshares_wallet/commit/5e56d52f2cd18214815a2d8b12d2a6791542773c#diff-5f62c9e0b4b2fce5ce0285e385187f1dL114
- [1] https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm#Signature_generation_algorithm
- [2] https://arstechnica.com/information-technology/2013/08/google-confirms-critical-android-crypto-flaw-used-in-5700-bitcoin-heist/
- [3] https://tools.ietf.org/html/rfc6979
- [4] https://github.com/jmjatlanta/bitshares_wallet/compare/5e56d52...5c28871
- [5] https://random-k.com
- [6] https://github.com/bitshares/bitshares-ui
- [7] https://github.com/bitshares/bitshares-core