https://sdtimes.com/recover-roca-vulnerability-github/

How to recover from RoCA vulnerability on GitHub

Latest News

Published: October 25th, 2017

- Ian C. Schafer

SHARE THIS

Hello akkornel,

GitHub has recently implemented new measures to identify and block insecurely generated SSH keys from being added to accounts. GitHub has also analyzed all existing keys that were added before this additional validation was in place.

As a result of these new measures the following key was identified and removed from your account:

Yubikey-based key
58:1a:a7:99:fd:14:3e:3b:f9:67:a5:ca:d4:00:cb:dd

...

After the “Revenge of Coppersmith’s Attack,” or RoCA, vulnerability left his Yubico-provided SSH key compromised, Stanford University senior infrastructure systems engineer A. Karl Kornel found that, thanks to GitHub’s necessary but inconvenient security measures, every commit he’d signed would no longer pass certification.

The Center for Research on Cryptography and security describes RoCA as a “vulnerability in generation of RSA keys used by a software library adopted in cryptographic smartcards, security tokens and other secure

hardware chips manufactured by Infineon Technologies AG allows for a practical factorization attack, in which the attacker computes the private part of an RSA key."

In a post on his personal blog, Kornel outlined his process to resign all of his commits in the least time-consuming or inconvenient way for himself and others that relied on those repositories.

"I have used my now-revoked key to sign tags and commits; in public repositories, and in Stanford-internal repositories," Kornel wrote. "One of the internal Git repositories mandates that all commits be signed; that repository validates signatures against keys kept in a separate, server-local keyring. The signed tags are easy enough to deal with: I simply re-create them, using my new key. The annoyance is that I have to go through each tag to find the ones that I have signed."

The basic process was to create a new commit signed with the old key that marks where the key will change and explains the revocation, followed by a another commit using the newly generated key that will mark prior commits with a note explaining that despite GitHub's mark for an invalid key, they can be trusted as long as the current key is valid.

Though Kornel says there are some weaknesses to this method, one of which involves someone potentially accessing his computer. The other he addresses with an out-of-band posting.

"What I've done is, in my opinion, a good human-readable solution," Kornel wrote. "I'm sure there are problems that I've missed, but I hope this provides at least some protection. Of course, this only works if a human actually reads it: programs using git verify-commit will continue to complain about commits made with my revoked sub-key."

The full write-up can be found on Kornel's blog.