Open to a Fault: On the Passive Compromise of TLS Keys via Transient Errors

Presenter: Patrick Marschoun

March 4, 2025

Background

- TLS connections are ubiquitous on the internet
- Faults during the TLS handshake could lead to the private key being leaked
- Faults can be triggered by active scans [1]
- Exploits code optimizations using Chinese Remainder Theorem (CRT)

Contribution

- Studies passively collected datasets [2]
- Focuses on hardware faults instead of adversarial scans
- Examines historical TLS scan data
- Discusses defenses to discovered vulnerabilities

Network Tap

- Collected traffic from two campuses
- Used IDS scripts to filter only TLS traffic
- Collected both TLS 1.2 and 1.3 traffic

Passive Analysis

- TLS handshake could fail during key computation, transmission, or collection
- Most faults occurred from the hash function, not the RSA calculation
- For 200 PKCS#1v1.5 signatures with faulty padding, 11 leaked the private key

Passive Analysis

- RSA-PSS is fault tolerant if salt input for padding is randomized and unpredictable
- It was found that practical exploitation of transient faults is unlikely

Passive Analysis

- ECDSA is vulnerable if a correct and faulty signature are sent which use the same message hash and signature nonce
- The client and server randoms are supposed to be generated for each handshake
- The paper found a non-negligible number of connections which repeated some or all of these randoms
- A faulty RNG was also found to have caused some leaked private keys

Defenses

- The fault which leaked private keys was an error in the signature generation
- Protection gained by validating signatures before transmissions
- Most open-source cryptographic libraries patched this is 2015

Defenses

- Randomization was also found to provide protection for faulty signatures
- Must be balanced with risk of improper RNG implementation
- RSA fault attack through the CRT optimization

Discussion

Highlights

- Practical and undetectable attack
- Root cause analysis of TLS handshake failures
- Large dataset analyzed
- Responsible reporting and patching
- Detailed background material on relevant cryptographic primitives

Improvement and Next Steps

- Investigate if certain hardware and/or software configurations cause more faults
- Many faults had no identifiable cause
- More investigation into faults tolerance of TLS 1.3
- Empirical analysis of proposed defenses
- Investigating RSA and ECDSA outside of TLS

Discussion Questions

- What are the trade-offs between using randomization to increase fault tolerance at the expense of increasing implementation vulnerabilities?
- How severe of an attack has been exposed here? Do you agree that this attack models nation state adversaries?
- Given that signature validation has already been implemented in most of the open-source cryptographic libraries, how much impact has this paper had on the security community?
- Could similar attacks be performed against other key generation protocols?

Discussion Questions

- Since TLS 1.3 has countermeasures which help make it more fault tolerant, what could be done to increase adoption of modern protocols?
- At what point should security concerns take precedence over backwards compatibility concerns?
- Should regulations exist to ensure that vulnerabilities in closed-sourced cryptographic implementations get patched in a timely manner?

References

F. Weimer, "Factoring rsa keys with tls perfect forward secrecy," 2015.

G. A. Sullivan, J. Sippe, N. Heninger, and E. Wustrow, "Open to a fault: On the passive compromise of TLS keys via transient errors," in *31st USENIX Security Symposium (USENIX Security 22)*, (Boston, MA), pp. 233–250, USENIX Association, Aug. 2022.

Questions?