Building Secure Decentralized Applications the DECENT Way

Haofan Zheng
Xiaowei Chu
Owen Arden

University of California, Santa Cruz
Remote Attestation

- A process for the enclave to gain the trust of a remote service, so that the remote service will confidently reveal the secret to the requesting enclave
- In general, the remote service needs to ensure:
  - The enclave is running on a legitimate platform
  - The enclave is the expected one (by comparing the hash of the enclave)
Remote attestation (RA) enables developers to deploy trusted code to enclaves on untrusted hosts and authenticate them remotely.
Problem Statement

- Remote attestation (RA) enables developers to deploy trusted code to enclaves on untrusted hosts and authenticate them remotely.
- However
  - The RA protocol is complex
Problem Statement

- Remote attestation (RA) enables developers to deploy trusted code to enclaves on untrusted hosts and authenticate them remotely.
- However
  - The RA protocol is complex
  - Mutual authentication is non-trivial
  - Updating components is challenging
White List and Agreement

- For mutual authentication, a load-time white list is required

Hash a: Enclave A
Hash b: Enclave B
Hash c: Enclave B
Hash d: Enclave C

...
White List and Agreement

- But white list mismatch could lead to insecure flows
- Therefore, enclaves should trust other enclaves with the same white list
Self Attestation

- DECENT enclaves authenticate using certificates that bind a unique key pair to an enclave instance using a RA.
- Remote attestation is done once at load-time, and periodically refreshed.
- Use the unique private key to sign the certificate (which includes the white list).
How to Update Trusted Code/Components?

- Since white list are defined at load-time, updating would require restarting all components
- **Verifiers** are distinguished enclaves (defined by the white list) that may authenticate new enclave by signing their certificates
How to Update Trusted Code/Components?

- Example verifier:
How to revoke compromised keys or vulnerable code?

- If a TEE platform is compromised or an enclave is found to be vulnerable, the platform or the enclave must be revoked
  - Revoked platforms will fail to produce fresh RAs
  - Revoking vulnerable enclaves requires mechanism similar to verifiers
How to revoke compromised keys or vulnerable code?

- Example revoker:
DECENT Handshake

- All the procedures are done during the TLS handshake.
Case Study

- DECENT DHT
- DECENT Ridesharing
DECENT DHT

- DHT data stored is based on Chord, but it is an encrypted data store, where only the authorized application can access the data
- Advantages
  - Data is encrypted by enclave's seal key, thus, no centralized proxy or separate key management mechanism is needed
  - Even if one node is compromised, the rest of data remains secure
  - Protects the integrity of the fingertable metadata
DECENT Ridesharing

- Inspired by the microservice architecture of Uber
- It evaluates that DECENT framework supports complex decentralized applications with multiple components
- Advantages
  - Enclaves provide integrity of workflow and billings
  - Driver's or passenger's information only revealed when matched
  - Location and routes are private to drivers and passengers

*Uber microservice architecture: https://dzone.com/articles/microservice-architecture-learn-build-and-deploy-a
Conclusion

- **DECENT framework supports** building decentralized applications with enclaves
  - Enclave authentication uses certificate backed by RA
  - Load-time white list ensures that only authorized components can join the system
  - Verifiers and revokers provide run-time modification to the set of authorized components
- **Implemented** DECENT framework with Intel SGX
  - Built DECENT DHT and DECENT Ridesharing to evaluate the DECENT framework
  - We are still working on experiment
  - Early result from simple experiment shows the overhead is lower than native SGX RA protocol
Thank You!
& Questions?