CS 6431: Security and Privacy Technologies

Smart Contracts

Instructor: Ari Juels
2 December 2016
Five Big Research Challenges

• Challenge 1: Scaling and performance
• Challenge 2: Correctness
• Challenge 3: Confidentiality
• Challenge 4: Strongly authenticated data
• Challenge 5: Safety: resilience to error and mitigation of crime
Correctness
The DAO

• Poster child for smart contract coding errors
  • http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/
• Crowd-funded investment fund
• Launched 30 April 2016
• Idea:
  • Buy “DAO tokens”
    • Like shares of equity
  • Token-holders vote on investment proposals
  • Community profits from returns
• Complicated (!) investment vehicle
  • How does it work? Read the (1400 lines of) code…
Splitting

- Of course, you might want out
  - You don’t like investment choices
  - You want to withdraw funds
- “Splitting” (splitDAO function) creates child DAO
  - Independent tokens + voters
- Unfortunately, had reentrancy bug…
Example

```
contract SendBalance {
  mapping (address => uint) userBalances;
  bool withdrawn = false;

  function getBalance(address u) constant returns (uint){
      return userBalances[u];
  }

  function addToBalance () {
      userBalances[msg.sender] += msg.value;
  }

  function withdrawBalance () {
      if (! (msg.sender.call.value(0) )) { throw; }
      userBalances[msg.sender] = 0;
  }
}
```

Figure 7: An example of the reentrancy bug. The contract implements a simple bank account.

Sends money to caller contract, invokes “fallback” function ...before changing user balance

“fallback” function recursively calls

From Luu et al. 2016
The DAO attack

• The DAO attacker exploited reentrancy bug in SplitDAO starting 17 June 2016
  • (Actually, rather more complicated)

• Eventually siphoned $60 million (3.6 million Ether) in tokens into child DAO it controlled

• But…

• The DAO enforced 27-day waiting period for child DAO creation
Solution?

• “Soft fork” proposed, but buggy

• “Hard fork” on 20 July
Market impact

DAO attack

Hard fork

+ creation of ETC (about 10% of miners)
Long-term solutions?

・Luu et al. (2016), static analysis via Oyente tool
・Better programming languages
・EVM modification
Strongly authenticated data
Example application: Self-enforcing insurance policy

Flight Insurance

Gimme a $100 policy
(Flight #1215, 17 May, Policy price: $1)

$100
Smart contracts are data-hungry, but...

- Blockchains are self-contained (can't browse internet)
- Data must be injected from user account!
- Proposed data-delivery approaches:
  - Prediction markets (e.g., Gnosis)
  - Oracles, e.g., Oraclize.it
  - Big data brokers serve data
Town Crier (TC): Basic Idea

[Zhang et al. 2016]

Blockchain

Flight Insurance

Is DL 2777 delayed?

Trusted Website XYZ.com

Authenticity property: Data delivered by TC is exactly as served on source site XYZ.com
How to ensure authenticity?

Blockchain
Flight Insurance

Trusted Website XYZ.com
Intel SGX: Isolation

**Integrity**

Other processes and even OS cannot tamper with control flow of X

**Confidentiality**

Other processes and even OS can learn nothing* about the state of X

* Side-channels like page faults excepted
Intel SGX: Remote attestation

 attestation \( \text{att} = \Sigma_{\text{intel}}[\text{Build}(X) \ || \ \text{User data}] \)

*Signature \( \Sigma \) (EPID) can be anonymous (group) or pseudonymous

Remote entity
TC goal / adversarial model

- Relying contract sends query $Q = (XYZ, \text{params, } T)$ to TC
- Goal: TC returns correct answer $A$ to query $Q$
- Assumption: TC code trustworthy (publicly verified)
- Adversary controls TC node OS and the network
  - Simpler view: adversary controls network outside enclave
TC goal / adversarial model

Adversary controls network outside enclave
- TC code is published
- TC instance created with public key $PK_{TC}$ to sign data
• Creator of Flight Insurance checks $\texttt{att} = \Sigma_{\text{intel}}[\text{Build(TC)} \parallel PK_{\text{TC}}]$

• Contract Flight insurance:
  • Hardwired with $PK_{\text{TC}}$
  • On receiving flight data, checks signature $\Sigma_{SK_{\text{TC}}}[\text{flight data}]$
Actually, a little more complicated…
Tripartite trust model

- **Smart contract**
  - Trustworthy execution
  - No confidentiality
  - Expensive computation

- **TC enclave**
  - Trustworthy execution
  - Cheap computation
  - No network stack

- **Trusted website**
  - Source authentication
  - No digital signatures on content

Blockchain

- Flight Insurance
- TC Contract
- Town Crier

https://www
Blockchain

Flight Insurance

Town Crier

Trusted Website
XYZ.com
- Problem: Q' may be corrupted version of Q—or altogether fake!
- Could run blockchain client (Ethereum) in enclave
  - Could verify Q from blockchain
  - Would bloat TC code (TCB)!
• Our approach: Leverage hybrid trust model
• Potentially *corrupted* query $Q'$ processed by TC
• TC digitally signs $(A', Q')$ using $\text{SK}_{TC}$
• TC contract verifies $Q = Q'$
- Problem: HTTPS doesn't sign data
  - How can TC forward authenticated data?
- Solution: Free!
  - TLS connection *terminates in enclave*
  - We just need to trust (attested) TC code to fetch correct data
Smart Contract Challenge #2: Confidentiality

How to provide confidentiality for data processed by a smart contract with no confidential state?
Running example: Self-enforcing insurance policy
Idea: Leverage enclave confidentiality

- Private datagram!
- Additional steps needed
  - E.g., delay response / payment to noise timing side-channel
TC for sale of online goods

Steam Community Marketplace → Ether

Online game license
Smart contract for fair exchange

Blockchain

Steam Trader
Problem

- For fair exchange SteamTrader needs to verify delivery
- This requires Alice's (or Bob's) Steam marketplace credentials
Again, we can leverage enclave confidentiality!

✓ A game has been delivered to Bob.

Blockchain
Steam Trader

Alice’s Account
✓ A game has been delivered to Bob.

Town Crier
Again, we can leverage enclave confidentiality

**TC** enables confidential smart contracts for nearly any digitally represented asset ($, cryptocurrency, online accounts, etc.)
Private financial instruments with automated execution

Blockchain

Financial Instrument

+ Confidential settlement

Town Crier
And some unsavory uses...
E.g., password theft...

 ✓ Money delivered to Cruella.

 Town Crier

 ✓ Stolen password for Alice's account works!

 Alice's Account

 Alice's password

 Blockchain

 Password Theft