DASP Top 10 (Smart Contract Vulns)

Portland State
Computer Science

Recall

- Bitcoin
  - Scripting language with limited functionality to validate conditions for transactions to occur
  - Intended for handling transfers of digital stores of value (i.e. cash)
  - Reasonable security and verifiability of correctness due to language being non-Turing complete
- Ethereum
  - Full turing-complete language to implement arbitrary distributed application (DApp)
  - High-level language such as Solidity compiled down to Ethereum Virtual Machine bytecode (EVM)

But...

"Imagine trying to hack Bank of America—except you can read all of their code in advance, all of their transactions are public, and if you steal the money it's irreversible. Sounds like a paranoid worst-case scenario? Well, this is exactly the setup Ethereum smart contract developers have to deal with every day."
-- Ivan Bogatyy

Problems

- Improperly programmed contracts have led to an estimated $400 million in losses
- More than 30k contracts have known vulnerabilities in them
- Contracts immutable
  - Once deployed, code and any of its bugs remain forever
- Requires a deep understanding of security issues and secure programming to get right

Not easy

"[The DAO] contract, even if coded using best practices and following language documentation exactly, would have remained vulnerable to attack. […] the EVM was operating as intended, but Solidity was introducing security flaws into contracts that were not only missed by the community, but missed by the designers of the language themselves."

-P. Daian on DAO re-entrancy vulnerability

Some issues

- Semantic mismatches of Solidity language
  - Looks like Javascript, but doesn't act like it sometimes
  - Numbers are all `int` (no floating point supported in Solidity)
- Lack of code audits to catch errors
  - Improper visibility modifiers
  - Lack of input validation
  - Lack of error checking on calls
- Common language features misunderstood
  - Data types misused
  - Fallback functions
  - Low-level calls

- Languages obfuscate underlying mechanics of the blockchain paper
  - Conceal low-level operations for ease of programming at the cost of security
  - Examples
    - Making a transaction as simple as a `send()` command
      - Programmer unaware of complications due to failed transactions
    - Making time-relative computations as simple as (`now > a+1 day`)
      - `now` looks like variable, but is actually a function that changes on every use
      - Programmer not aware how timestamp can be manipulated
      - Underlying security, payment, and execution issues hidden in an API call wrapper

DASP Top 10

- Decentralized Application Security Project
- Similar to OWASP Top 10, but for DApps built on Ethereum VM
  - https://dasp.co/

1. Reentrancy
2. Access Control
3. Arithmetic
4. Unchecked Low Level Calls
5. Denial of Service
6. Bad Randomness
7. Front Running
8. Time Manipulation
9. Short Addresses
10. Unknown Unknowns

- Sobering statistic: Automatic analysis of 19,366 contracts worth $62M found 44% vulnerable paper

Security Innovations CTF

Portland State
Computer Science

SI CTF Lab 3.1 (D0_Donation)