Enter the Hydra:
Toward Principled Bug Bounties and Exploit-Resistant Smart Contracts

USENIX Security '18
17 August 2018
Floyd 'Crypto' Mayweather promotes an ICO, again

Mashable
AUG 24, 2017

You can call me Floyd Crypto Mayweather from now on...Hubii.Network #ICO starts tomorrow! Smart contracts for sports?! #HubiiNetwork #CryptoMediaGroup 😎

AUGUST 23
Crypto Tokens

• Sold in Initial Coin Offerings (ICOs); **ERC20**
  • a.k.a. Token Launch, Token Generation Events (TGEs), etc.
  • Like unregulated VC
  • Token like a share (kind of…)

• Since mid-2017, ICO funding outstripping early-stage Internet VC (!)

*Exhibit 8: The pace of ICO fundraising has now surpassed Angel & Seed stage Internet VC funding globally*

Total Funds Raised by month ($, millions)

Note: ICO fundraising as of July 18th, 2017, per Coin Schedule. Angel & Seed VC funding data as of July 31st, 2017 and does not include “crowdfunding” rounds.

Source: CoinSchedule, CB Insights, Goldman Sachs Global Investment Research.
Side effects of the token mania

- Token smart contracts are compact
- Lots of money per contract
- Astonishing value per line of code
- Which makes for juicy targets...

<table>
<thead>
<tr>
<th>Token</th>
<th>Lines of Code</th>
<th>Value per line</th>
</tr>
</thead>
<tbody>
<tr>
<td>OmiseGo (OMG)</td>
<td>396</td>
<td>~$1.4M</td>
</tr>
<tr>
<td>Tether (USDT)</td>
<td>423</td>
<td>~$6.14M</td>
</tr>
<tr>
<td>EOS (EOS)</td>
<td>584</td>
<td>~$15.8M*</td>
</tr>
</tbody>
</table>

Sources: coinmarketcap.com, 17 August 2018, and published contract source code
Some (in)famous smart contracts

• The DAO (June 2016)
  • Reentrancy bug ⇒ $50+ million stolen

• Parity multisig hack (July 2017)
  • Parity 1.5 client’s multisig wallet contract
  • Problem with library contract use ⇒ $30 million stolen
    …from 3 ICO wallets (Edgeless Casino, Swarm City, and æternity)

• Parity multisig hack—Redux! (Nov. 2017)
  • Problem with library contract ⇒ >$150 million frozen
    • …much from ICO wallets (Polkadot, $98 million)
N-Version programming
(Chen & Avizienis ’78, Knight-Leveson ‘86)

Input $X$

Version 1

Version 2

Version 3

Majority Vote

Agreed output

$N$ software versions / heads
If something goes wrong…

Version 1

Version 2

Version 3

Majority Vote

Agreed output

$N$ software versions / heads
What is N-version programming doing?

A program transformation $T$ takes $N \geq 1$ programs and creates new program $f^* := T(f_1, f_2, \ldots, f_N)$. 
Some more definitions

• Let $\mathcal{I}$ be an ideal program specification
  • Conceptual! Doesn’t actually exist… (on paper or code)

• Let $f$ be an implemented program

• An exploit is an input $X$ such that $\mathcal{I}(X) \neq f(X)$

• Intuition: Any deviation from intended behavior is a potentially serious bug

• Exploit set $E(f, \mathcal{I})$: set of exploits $X$ for $f$ and $\mathcal{I}$
Mind the gap

• Let $D$ be a distribution over inputs $X$

• Definition of exploit gap:

$$
gap := \frac{\Pr_{X \in D} \left[ X \in \bigcup_{i=1}^{N} E(f_i, \mathcal{I}) \right]}{\Pr_{X \in D} \left[ X \in E(f^*, \mathcal{I}) \right]}$$

• Affirmative gap ($> 1$) means $T$ reduces exploits

• Bigger gap $\Rightarrow$ fewer relative bugs in $f^*$

• gap captures dependencies among heads
Houston... we have a gap

\[
gap := \frac{\Pr_{X \in D} [X \in \bigcup_{i=1}^{N} E(f_i, I)]}{\Pr_{X \in D} [X \in E(f^*, I)]}
\]

Input \( X \)

\( f^* \)

\( f_1 \)

\( f_2 \)

\( f_3 \)

Version 1

Version 2

Version 3

N software versions / heads

Majority Vote

Agreed output
N-version-programming criticism

• Strong gap requires independence among heads
  • Correlations hurt!
• Knight-Leveson (1986):
  • “We reject the null hypothesis of full independence at a p-level of 5%”
• Eckhardt et al. (1991):
  • “We tried it at NASA and it wasn’t cost effective”
  • Worst case: 3 versions ⇒ 4x fewer errors
But not everything is a space shuttle…

• Not all software needs to be available at all times!
  • E.g., Smart contracts: How bad if it’s down for a while?
• In fact, often better no answer than the wrong one
  • Bugs are often harmful
• N-of-N-Version Programming (NNVP)
NNVP a.k.a. **Hydra Framework**

Idea: Strengthen majority vote of N-Version Programming
NNVP a.k.a. **Hydra Framework**

Unless *all versions agree, abort!*
NNVP a.k.a. Hydra

• Aborting in NNVP:
  Correctness ← Availability

• NASA numbers much better for NNVP
  • Some availability loss, but…
  • $\text{gap} = 4,409$ for $N = 3$ heads
  • $\text{gap} = 34,546$ for $N = 4$ heads
  • Probably even better!
Hydra creates a (strong) gap...

Serious bug in one head now rarely fatal...
Smart contracts are Hydra-friendly!

<table>
<thead>
<tr>
<th>Contract name</th>
<th>Exploit value (USD)</th>
<th>Root cause</th>
<th>Independence source</th>
<th>Exploit gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>The DAO* [19]</td>
<td>150M</td>
<td>Re-entrancy</td>
<td>language</td>
<td>✓</td>
</tr>
<tr>
<td>SmartBillions [20]</td>
<td>500K</td>
<td>Bug in caching mechanism</td>
<td>programmer</td>
<td>✓</td>
</tr>
<tr>
<td>HackerGold (HKG)* [21]</td>
<td>400K</td>
<td>Typo in code</td>
<td>programmer+language</td>
<td>✓</td>
</tr>
<tr>
<td>MakerDAO* [22]</td>
<td>85K</td>
<td>Re-entrancy</td>
<td>language</td>
<td>✓</td>
</tr>
<tr>
<td>Rubixi [23]</td>
<td>&lt;20K</td>
<td>Wrong constructor name</td>
<td>programmer+language</td>
<td>✓</td>
</tr>
<tr>
<td>Governmental [23]</td>
<td>10K</td>
<td>Exceeds gas limit</td>
<td>None?</td>
<td>✗</td>
</tr>
</tbody>
</table>

Hydra could probably have addressed cases in green and yellow vulnerabilities
Application: Bug Bounties
Some problems with bug bounties:

1. Bounties often fail to incentivize disclosure
   • Apple: \( \leq \$200k \) bounty
   • Zerodium: \$1.5 million for certain iPhone jailbreaks

2. Time lag between reporting and action
   • Weaponization can happen after disclosure

3. Bounty administrator doesn’t always pay!
Some problems with bug bounties:

1. Bounties often fail to incentivize disclosure
   - Apple: ≤ $200k bounty
   - Zerodium: $1.5 million for certain iPhone jailbreaks

2. Time lag between reporting and action
   - Weaponization can happen after disclosure

3. Bounty administrator doesn’t always pay!
The perfect bug bounty

1. **High leverage**: Small bounty incentivizes disclosure for valuable program

2. **Automatic payout**: Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable

3. **Automatic remediation**: Immediate intervention in affected software
Bug bounties: The Rational Attacker’s Game

Program
Value: $A
Bug bounties: The Rational Attacker’s Game

Find
Exploit

Attack

Disclose

$A

$B

Classic bounty: $B
Bug bounties: The Rational Attacker’s Game

Find 
Exploit 
Disclose if 
$B > A$

$A$

$B$

Classic bounty: $B$
The Hydra Framework for Bug Bounties

Input $X$

Fault manager

Abort

Pay $bounty$

$bounty$
The Hydra Hacker’s Dilemma

Claim bounty ($B$) now?  Try to break all heads ($A$)?
The Hydra Hacker’s Dilemma

Claim bounty ($B) now?  Try to break all heads ($A)?
Our goal: High leverage

Find
Exploit

Attack
$A/gap

Disclose
$B
Our goal: High leverage

For $\text{gap} \gg 1$
Our goal: High leverage

Find

Exploit

Disclose if $B > \frac{A}{\text{gap}}$

Attack

$A/\text{gap}^*$

Disclose

$B$

Exploit gap
Our goal: High leverage

Find

Exploit

Disclose

Attack

$B > \frac{A}{(\text{gap}+1)}$

Exploit gap
Wait a minute…

Program

Value: $A$

Disclose, i.e., don’t attack even though $B < A$?!
Example

• Recall: NASA experiments imply:
  • $\text{gap} = 4,409$ for $N = 3$ heads
  • $\text{gap} = 34,546$ for $N = 4$ heads

• So…
  • Approx $1 \text{ billion}$ contract (e.g., OmiseGo)
  • $N = 4$
  • $30k$ $bounty$ incentivizes adversary to disclose!
The perfect bug bounty

1. “Strong exploit gap”: Small bounty incentivizes disclosure for valuable program

2. Automatic payout: Bounty hunter need not trust bounty administrator to pay
   • Censorship-resistant, verifiable

3. Automatic remediation: Immediate intervention in affected software
Implementation

• ERC20
  • Standard token-management contract
  • $N = 3$
  • $\texttt{bounty} = 3\text{ETH} \sim \text{\$1k}$
  • Deployed @ 0xf4ee935a3879ff07362514da69c64df80fa28622

• Generalized Monty-Hall game
  • Extension of Monty Hall game to $K$ out of $M$ doors
  • In progress
Metacontract: EVM/Solidity governor, fault manager

Automatic Deployment Scripts
Test Scripts
Community contributions – Canonical Vyper ERC20, First 100% coverage ERC20 test suite

Instrumenter: EVM -> EVM transpiler

```solidity
function f(int x) payable {
    // reimburse sender and call g(x)
    (msg.sender).g.value(msg.value)(x);
}
```

```solidity
function f(int x, addr sender, uint val) {
    // send all call args to meta-contract
    MC.call(bytes4(sha3("g(int256)")),x,sender,val);
}
```

```assembly
MSTORE(M, 0x7877b803) #store sig of g in memory
MSTORE(M+4, CALLDATALOAD(4)) #store x
PUSH32(0) #output size and memory location
PUSH32(0) #output memory
PUSH32(36) #input size
PUSH32(M) #input memory
CALLVALUE #use msg.value as the call value
CALLER #use msg.sender as the dest address
GAS
CALL #this opcode will be instrumented

MSTORE(M', 0x7877b803) #store sig of g in memory
MSTORE(M'+4, CALLDATALOAD(4)) #store x
MSTORE(M'+36, CALLDATALOAD(36)) #store sender
MSTORE(M'+68, CALLDATALOAD(68)) #store value
PUSH32(0) #output size
PUSH32(0) #output memory
PUSH32(100) #input size
PUSH32(M') #input memory
PUSH32(0) #send 0 ether
PUSH32(MCaddress) #destination address of the call
GAS
CALL #after call returns, cleanup stack
```
## Is it practical?

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Contracts</th>
<th>Transactions</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODECOPY</td>
<td>50,147</td>
<td>5,646,607</td>
<td>medium</td>
</tr>
<tr>
<td>CALLCODE</td>
<td>30,109</td>
<td>1,213,064</td>
<td>hard</td>
</tr>
<tr>
<td>SELFDESTRUCT</td>
<td>24,707</td>
<td>739,249</td>
<td>easy</td>
</tr>
<tr>
<td>DELEGATECALL</td>
<td>19,749</td>
<td>2,695,326</td>
<td>hard</td>
</tr>
<tr>
<td>CREATE</td>
<td>11,559</td>
<td>1,143,961</td>
<td>easy</td>
</tr>
<tr>
<td>Other</td>
<td>6681</td>
<td>195,569</td>
<td>-</td>
</tr>
<tr>
<td>None</td>
<td>268,652</td>
<td>12,780,929</td>
<td>supported</td>
</tr>
</tbody>
</table>

Blocks 4690101 to 5049100  
(Dec-07-2017 -- Feb-07-2018)
Does it scale?

- Hydra + tail call opti.
- Hydra
- Linear scaling

Graph showing the gas cost of ERC20 operations (approval, deposit, transfer, transferFrom, withdraw) and Monty Hall as a function of the number of heads (1, 3, 5).
Is it fair? Submarine Commitments

- Prevent **frontrunning**
- Adversary sits on exploit
- Reveals when it detects pre-emption

Security analysis involved:
- New, strong adversarial model introduced for blockchains, see paper

---

\[ \mathcal{F}_{\text{withhold}} \text{ with } \mathcal{P} = \{P_0, P_1, \ldots, P_m\}, (\delta, \rho)-\text{adversary } A, \text{ blocksize } s, \text{ target height } n \]

**Initialisation**
- \( B \leftarrow \emptyset, B.\text{Height} \leftarrow 0, \text{MaxHeight} \leftarrow 0, \text{Mempool} \leftarrow \emptyset \)

**On receive** ("post", \( r \)) from \( P_i \):
- \( B \) submits tx
  - \( \text{assert ValidTx}(\tau; B, \text{Mempool}) \)
  - \( \text{tag}(r) \leftarrow (B.\text{Height}, P_i) \)  // Label tx with current chain height and sender
  - \( \text{Mempool} \leftarrow \text{Mempool} \cup r \)
  - \( \text{send Mempool to } A \)

**On receive** ("add block", \( B \)) from \( A \):
- \( A \) extends blockchain
  - \( \text{if } B.\text{Height} = n \) then
    - \( \text{output } B; \text{halt} \)  // To complete chain, \( A \) adds arbitrary \( n + 1 \)th block
  - \( \text{assert } (|B| = s) \land (B \subseteq \text{Mempool}) \)
  - \( \text{assert } \exists r \in \text{Mempool} \setminus B \text{ s.t. } (\text{tag}(r) = (h, P_0)) \land (h \leq B.\text{Height} - \delta) \)
  - \( \text{// Ensure delay at most } \delta \text{ for } P_0 \text{'s transactions} \)
  - \( B.\text{Height} \leftarrow B.\text{Height} + 1 \)
  - \( B.\text{Height} \leftarrow B \)  // Add new block to chain
  - \( \text{Mempool} \leftarrow \text{Mempool} \setminus B \)  // Remove processed txs from Mempool
  - \( \text{MaxHeight} \leftarrow \max(B.\text{Height}, \text{MaxHeight}) \)
  - \( \text{send } B \text{ to } P_0 \)

**On receive** ("rewind", \( r \)) from \( A \):
- \( A \) rewinds by \( r \) blocks
  - \( \text{assert MaxHeight} - (B.\text{Height} - r) \leq \rho \)
  - \( \text{// Ensure that } A \text{ rewinds by no more than } \rho \)
  - \( \text{Mempool} \leftarrow \text{Mempool} \cup \{B_i\}_{i \in [\text{MaxHeight} - r, B.\text{Height}]} \)
  - \( \text{// Return rewound transactions to Mempool} \)
  - \( B.\text{Height} \leftarrow B.\text{Height} - r \)

**Figure 2:** Ideal functionality \( \mathcal{F}_{\text{withhold}} \) for \((\delta, \rho)-\text{adversary } A\)
Smart Contracts - Innovate, Don't Apply

- Rich, new adversarial setting for security

- **Novel properties over classical system**
  - Known program value - dynamic bounties
  - Rigorous/programmatic/"Cartesian" security
  - Can derive known, precise economic security level

- New challenges in underlying environment/modeling
  - (find me offline! :))
The Hydra Project [alpha]

Hydra is a cutting-edge Ethereum contract development framework for:

decentralized security and bug bounties
rigorous cryptoeconomic security guarantees
mitigating programmer and compiler error

READ THE PAPER  TRY THE ALPHA  CHAT ON RIOT

www.thehydra.io
IC3: ADVANCING THE SCIENCE AND APPLICATIONS OF BLOCKCHAINS

Latest on Blog

Paralysis Proofs: How to Prevent Your Bitcoin From Vanishing
by Fan Zhang, Phil Daian, Iddo Bentov, and Ari Juels on Thursday January 18, 2018 at 09:30 AM

Suppose that N players share cryptocurrency using an M-of-N multisig scheme. If N-M+1 players disappear, the remaining ones have a problem: They've permanently lost their funds. In this blog, we propose a solution to this critical problem using the power of the trusted hardware.

The Social Workings of Contract
by Karen Levy on Wednesday January 17, 2018 at 01:00 PM

Guest blogger Prof. Karen Levy describes how contracts often include terms that are unenforceable, purposefully vague, or never meant to be enforced, how this helps set expectations, and what this means for smart contracts.

News & Events

May 10-11, 2018
IC3 Spring Retreat in NYC
IC3 faculty, students and industry members gather twice per year to discuss the major technical challenges and innovative solutions to widespread blockchain adoption.

February 26, 2018 - March 2, 2018
Financial Cryptography and Data Security 2018 and the 5th Workshop on Bitcoin and Blockchain Research.
Prof. Sarah Meiklejohn is co-Program Chair for FC18 and Prof. Ittay Eyal is co-Program Chair for the 5th Workshop on Bitcoin and Blockchain Research.
Thanks!

We thank Paul Grubbs and Rahul Chatterjee for comments and feedback. This research was supported by NSF CNS-1330599, CNS-1514163, CNS-1564102, and CNS-1704615, ARL W911NF-16-1-0145, and IC3 Industry Partners. Philip Daian is supported by the National Science Foundation Graduate Research Fellowship DGE-1650441. Lorenz Breidenbach was supported by the ETH Studio New York scholarship.

IC3 Industry Partners -