Florian Tramèr
Stanford

Phil Daian, **Ari Juels**
Cornell Tech, Jacobs, IC3

Lorenz Breidenbach
Cornell Tech, ETH, IC3

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

USENIX Security '18
17 August 2018


MAYWEATHER
PROMOTIONS
HUBLOT
WBC

Floyd 'Crypto' Mayweather promotes an ICO, again

Mashable AUG 24, 2017

Wait, how do you make Ethereum rain?

566.9k likes 9,425 comments

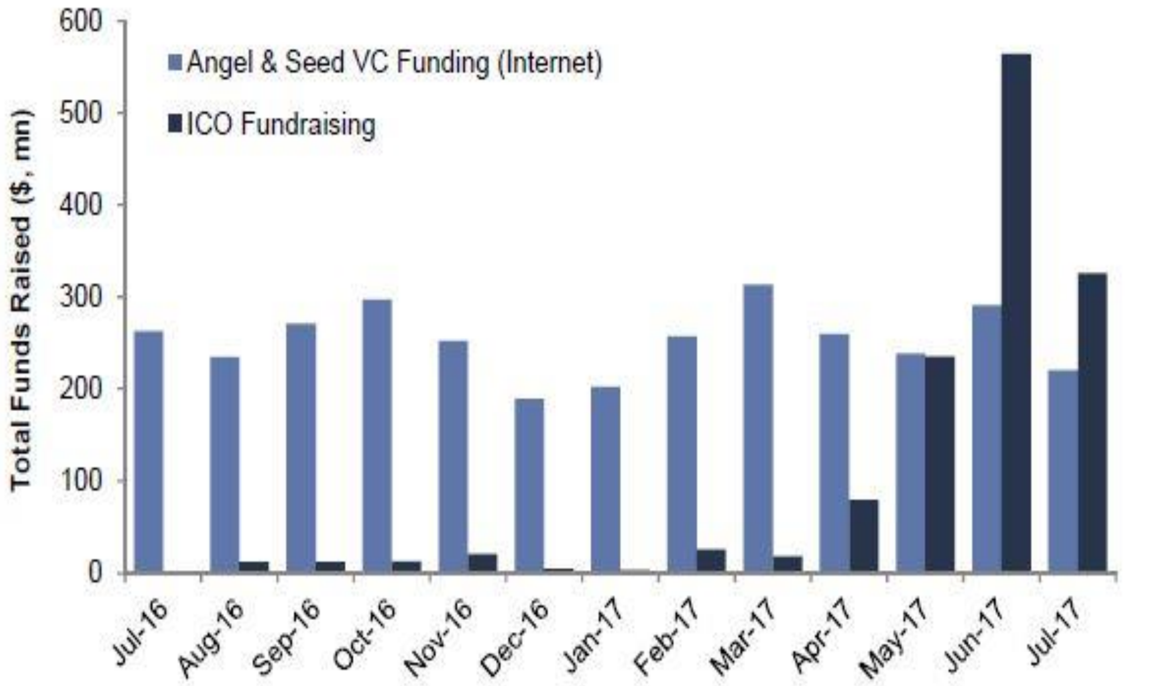
You can call me Floyd Crypto Mayweather from now on...Hubii.Network #ICO starts tomorrow! Smart contracts for sports?! #HubiiNetwork #CryptoMediaGroup 🤑🤑

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…

Token	Lines of Code	Value per line
OmiseGo (OMG)	396	~$1.4M
Tether (USDT)	423	~$6.14M
EOS (EOS)	584	~$15.8M*

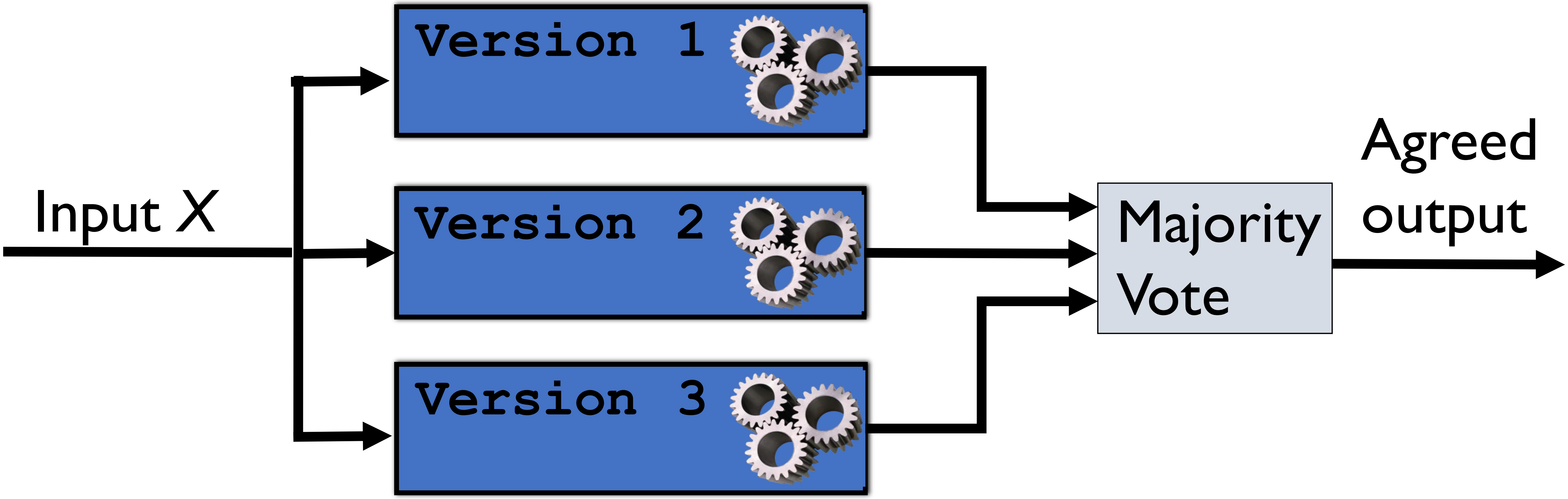
Sources: coinmarketcap.com, 17 August 2018., and published contract source code

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)

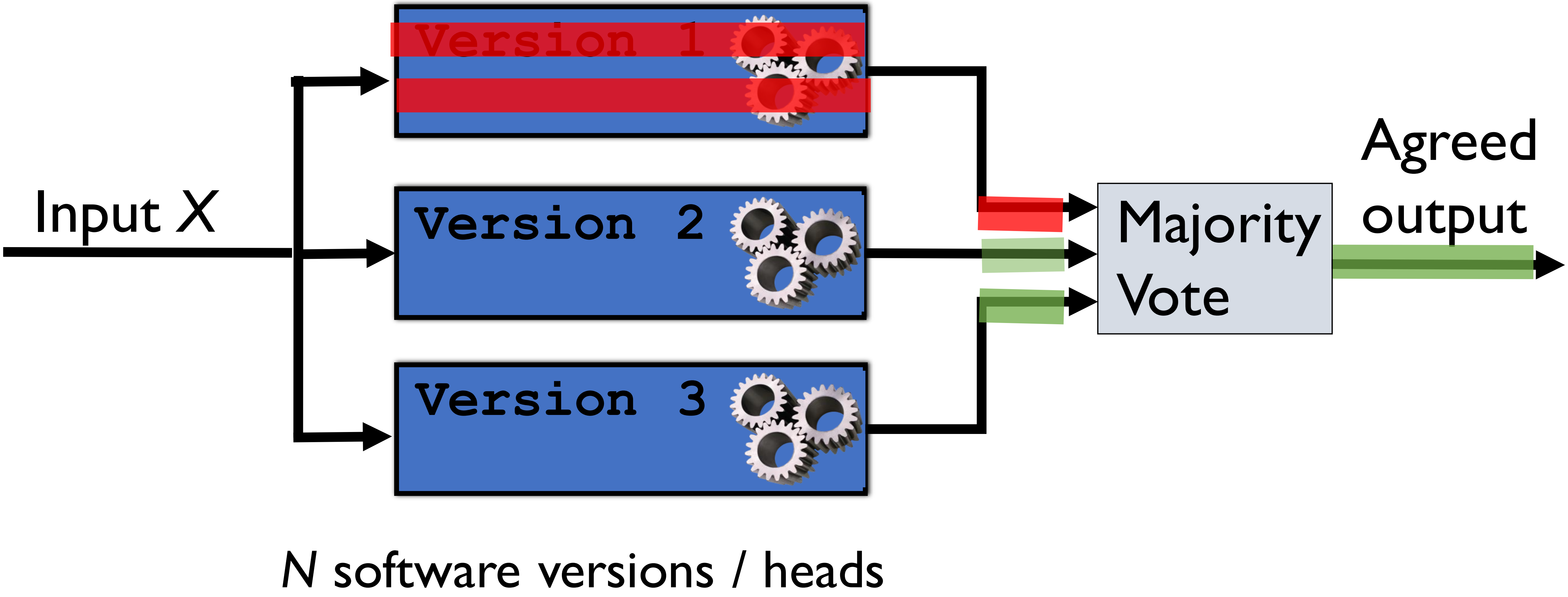

BACK TO THE FUTURE™
A ROBERT ZEMECKIS FILM
A ROBERT ZEMECKIS FILM
OUTATIME

N-Version programming
(Chen & Avizienis '78, Knight-Leveson '86)



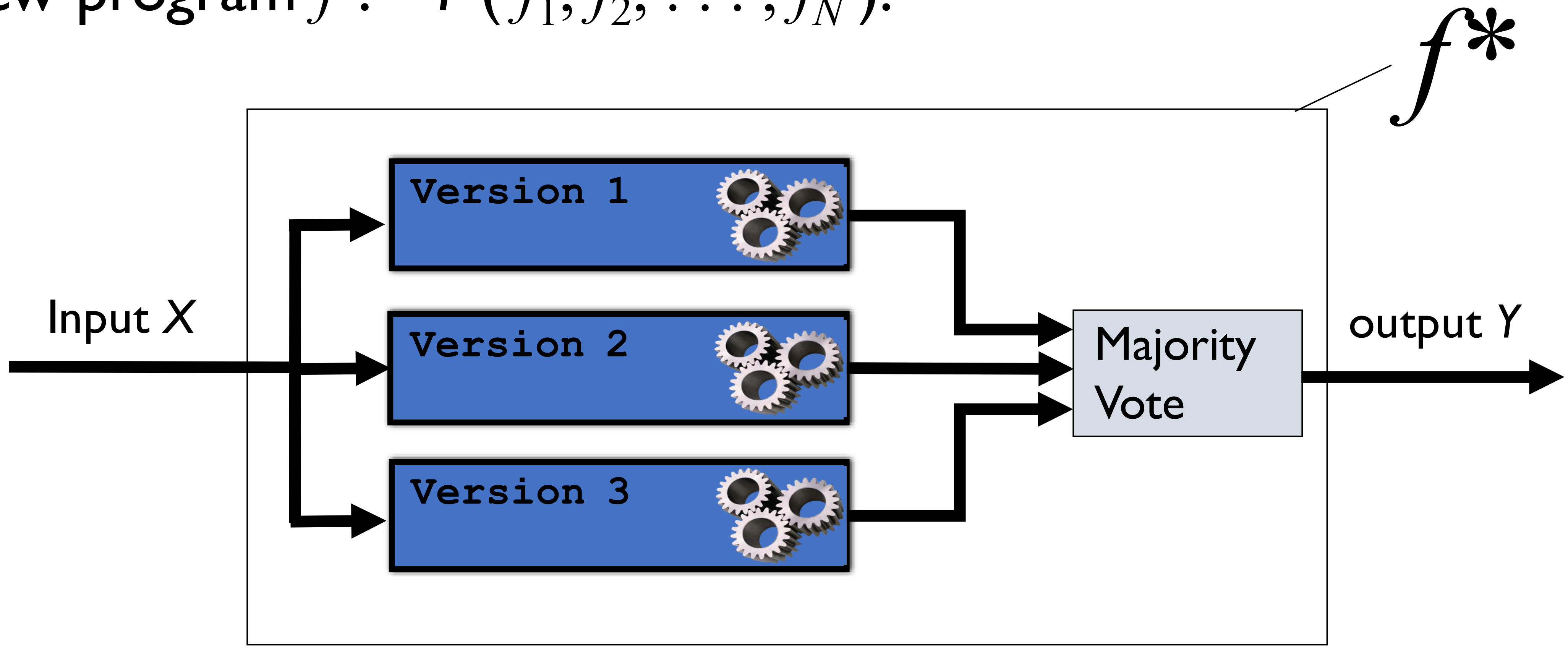
N software versions / heads

If something goes wrong…



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***:

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

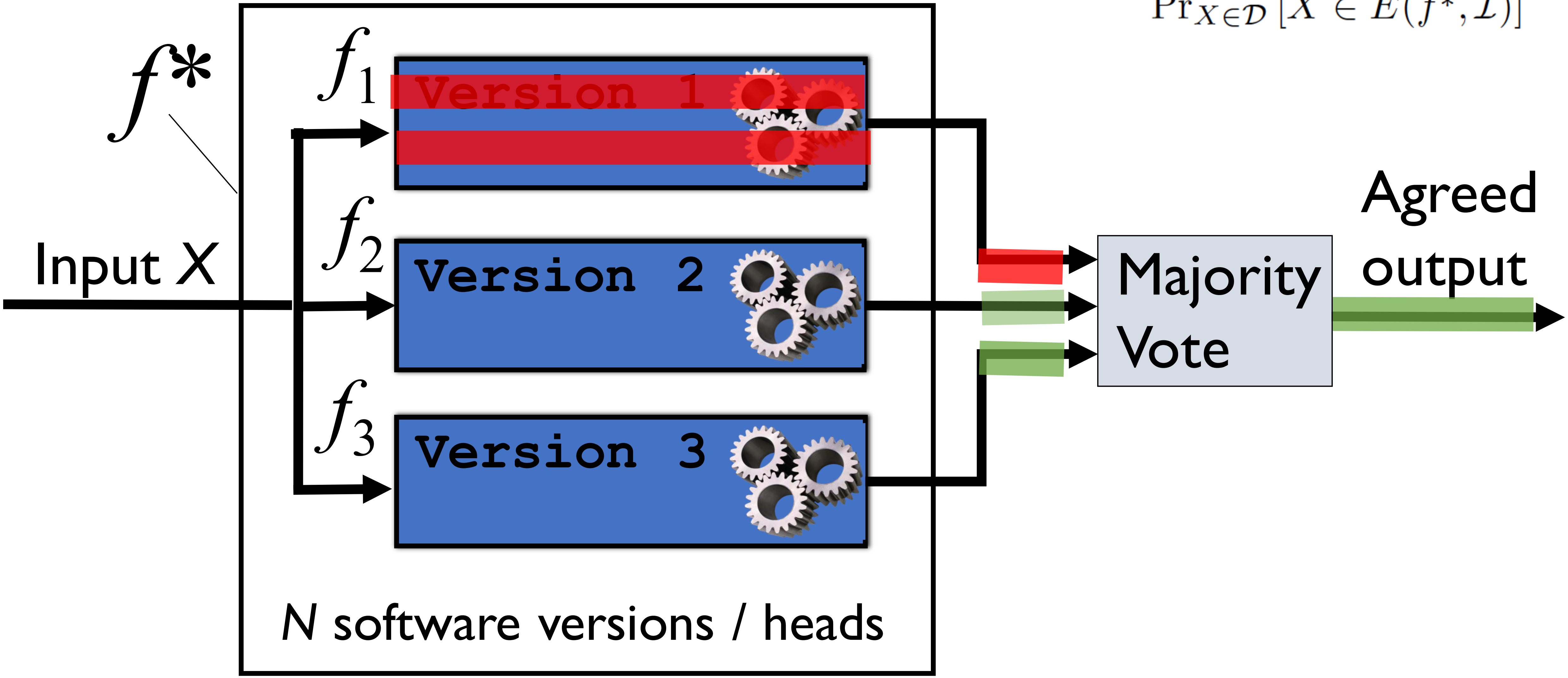
Exploits against $f_1, f_2, f_3 \ldots$

Exploits against f^*

- *Affirmative* `gap` (> 1) means *T* reduces exploits
- Bigger `gap` ⇒ fewer relative bugs in f^*
- `gap` captures dependencies among heads

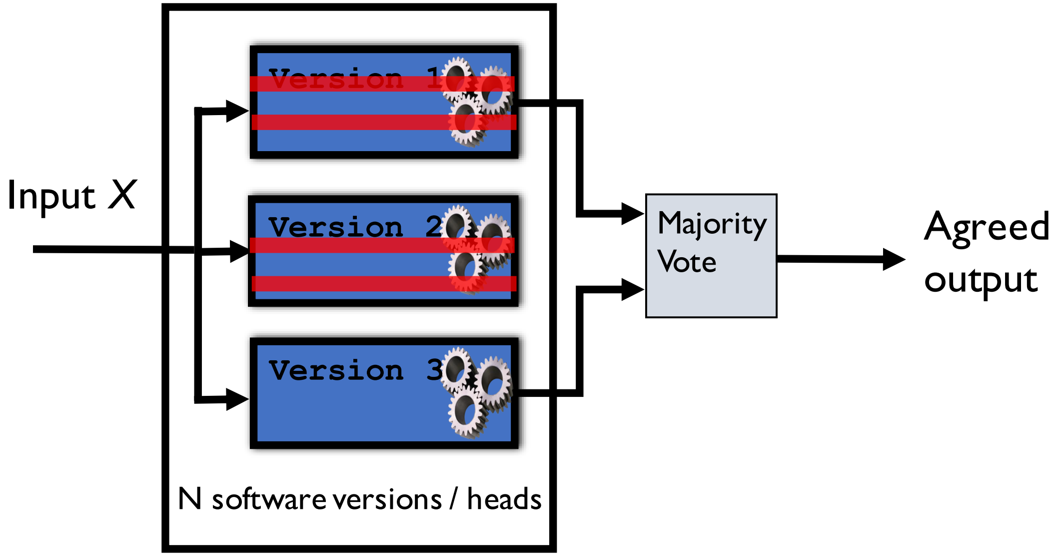
Houston… we have a gap

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



N-version-programming criticism

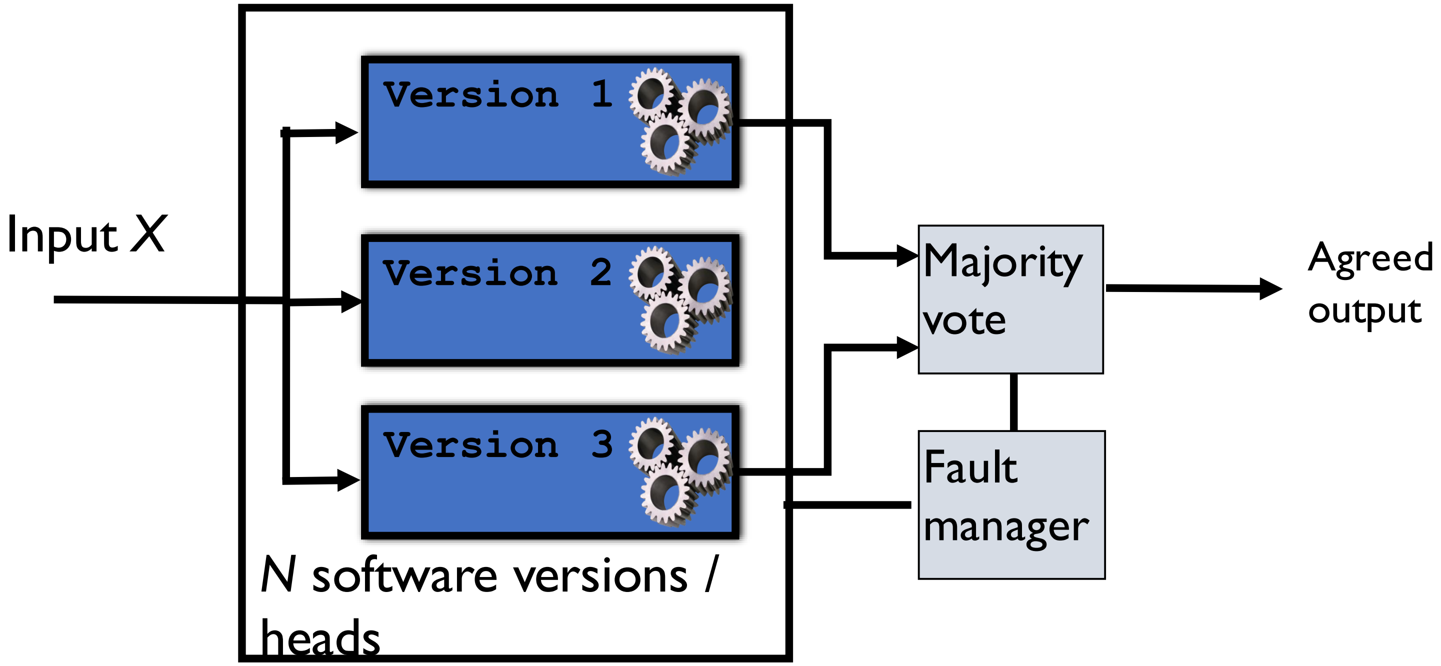
- Strong gap requires independence among heads
  - Correlations hurt!
- Knight-Leveson (1986):
  - "We reject the null hypothesis of full independence at a p-level of 5%"
- Eckhardt et al. (1991):
  - "We tried it at NASA and it wasn't cost effective"
  - Worst case: 3 versions ⇒ 4x fewer errors



But not everything is a space shuttle…

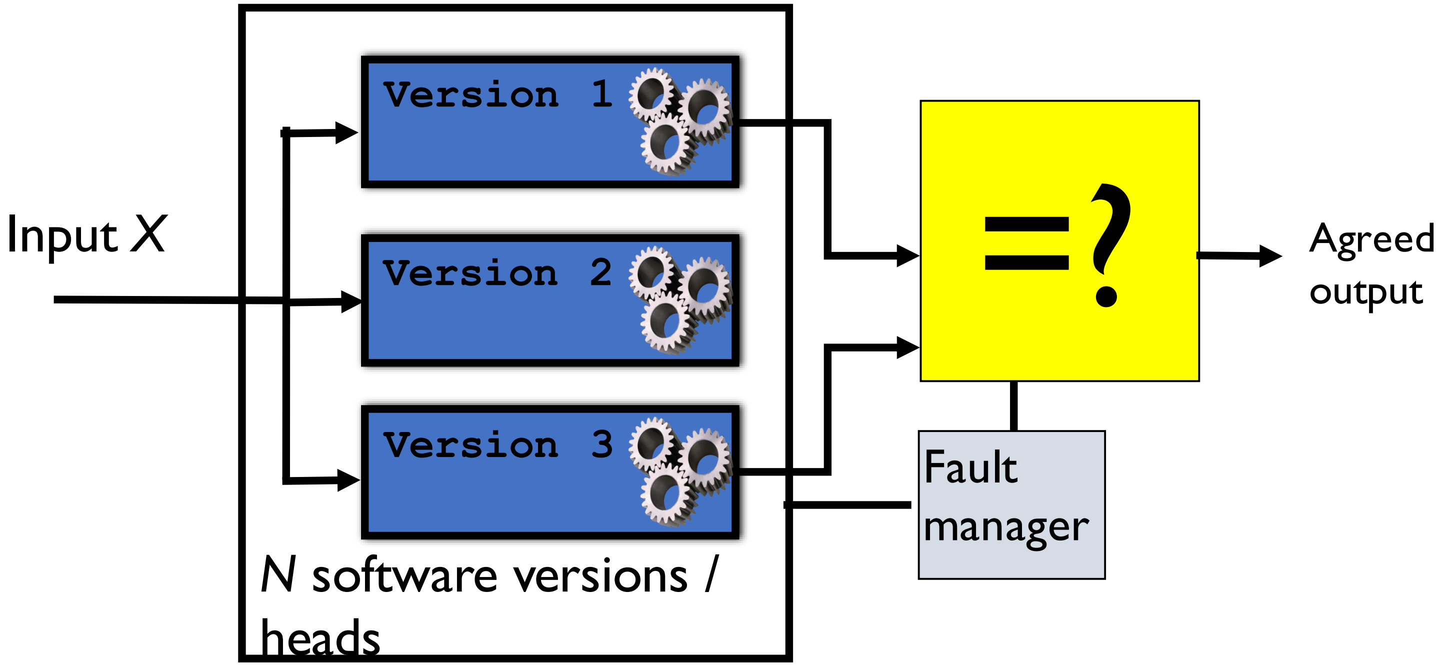
- Not all software needs to be available at all times!
  - E.g., Smart contracts: How bad if it's down for a while?
- In fact, often ***better no answer than the wrong one***
  - Bugs are *often harmful*
- ***N-of-N-Version Programming (NNVP)***

NNVP a.k.a. **Hydra Framework**



Idea: Strengthen majority vote of N-Version Programming

NNVP a.k.a. **Hydra Framework**



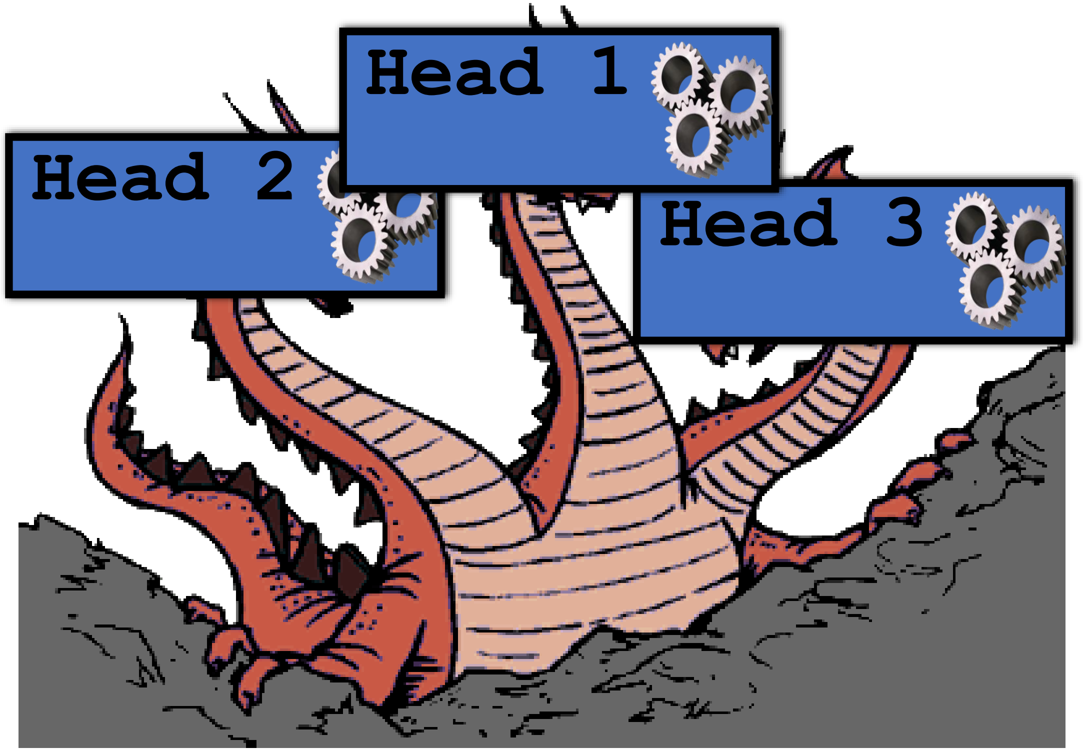
Unless *all versions agree, abort!*

NNVP a.k.a. **Hydra**

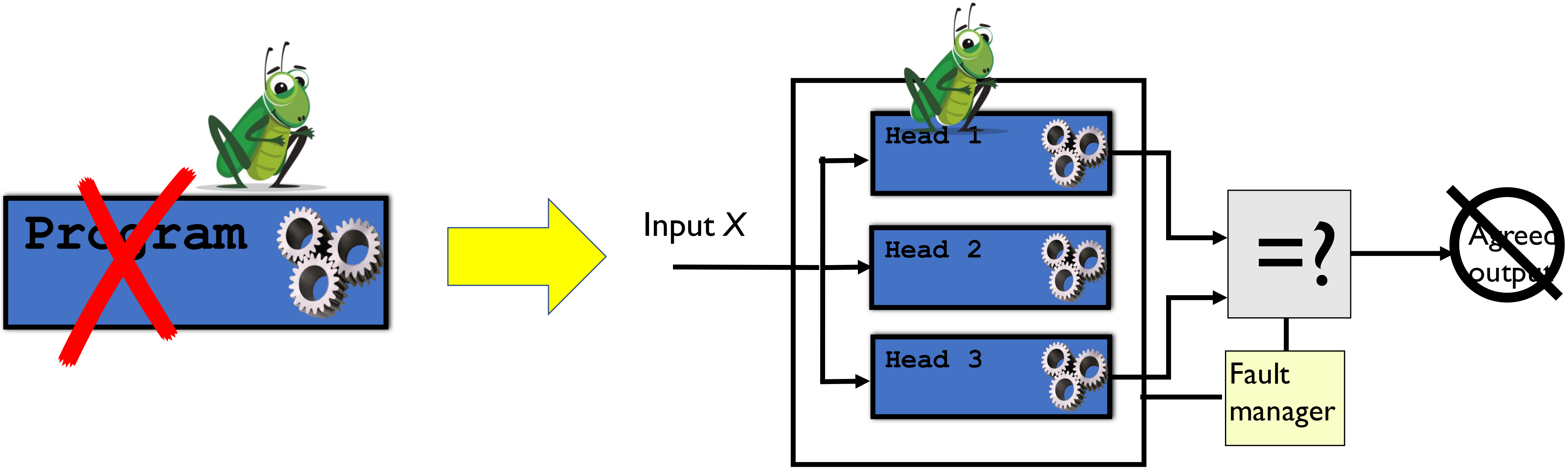
- Aborting in NNVP:

  Correctness ← Availability

- NASA numbers much better for NNVP
  - Some availability loss, but…
  - `gap` = 4,409 for N = 3 heads
  - `gap` = 34,546 for N = 4 heads
  - Probably even better!



Hydra creates a (strong) `gap`...

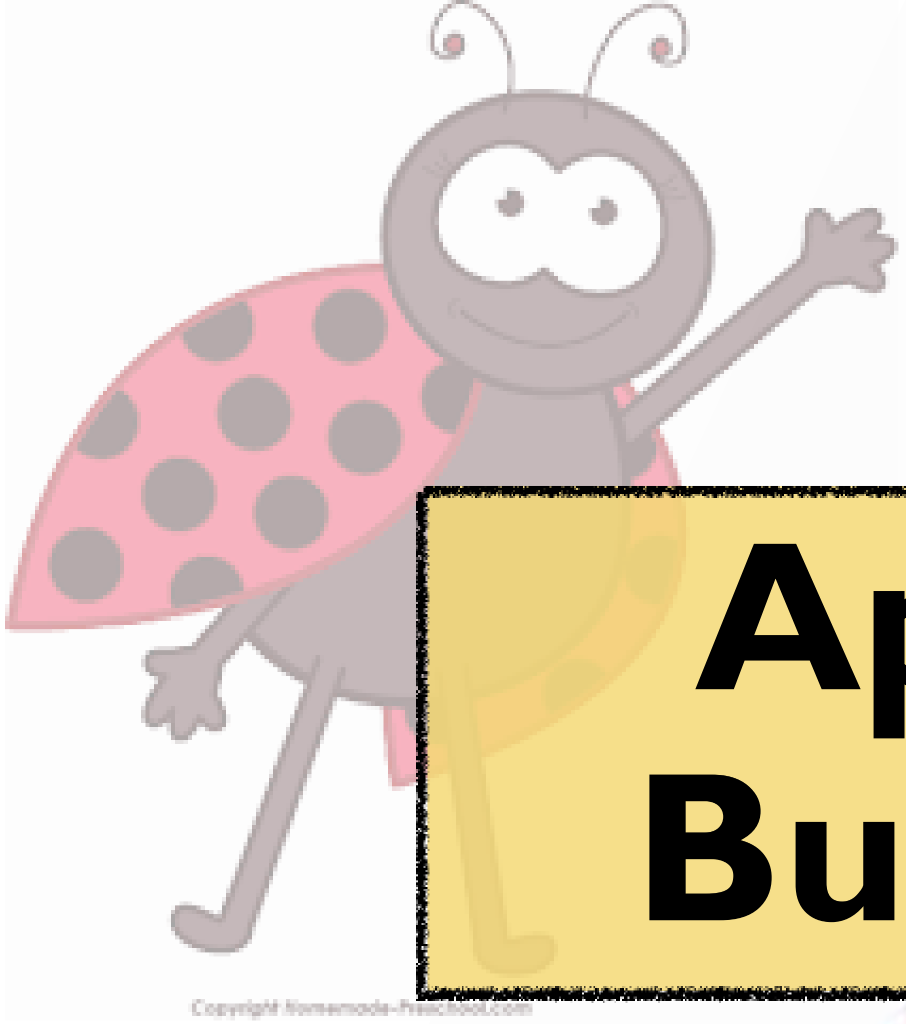


Serious bug in one head now rarely fatal...

Smart contracts are Hydra-friendly!

Contract name	Exploit value (USD)	Root cause	Independence source	Exploit gap
Parity Multisig [3]	180M	Delegate call+unspecified modifier	programmer/language?	✔/✗
The DAO* [19]	150M	Re-entrancy	language	✔
SmartBillions [20]	500K	Bug in caching mechanism	programmer	✔
HackerGold (HKG)* [21]	400K	Typo in code	programmer+language	✔
MakerDAO* [22]	85K	Re-entrancy	language	✔
Rubixi [23]	<20K	Wrong constructor name	programmer+language	✔
Governmental [23]	10K	Exceeds gas limit	None?	✗

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: ≤ $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!

Information Security News, IT Security News & Expert Insights: SecurityWeek.Com

SECURITY WEEK

INTERNET AND ENTERPRISE SECURITY NEWS, INSIGHTS & ANALYSIS

Subscribe (Free) | CISO

Malware & Threats | Cybercrime | Mobile & Wireless | Risk & Compliance | Security Architecture

Cyberwarfare | Fraud & Identity Theft | Phishing | Malware | Tracking & Law Enforcement

Home › Vulnerabilities

Researchers Claim Wickr Patched Flaws but Didn't Pay Rewards

By Ionut Arghire on October 31, 2016

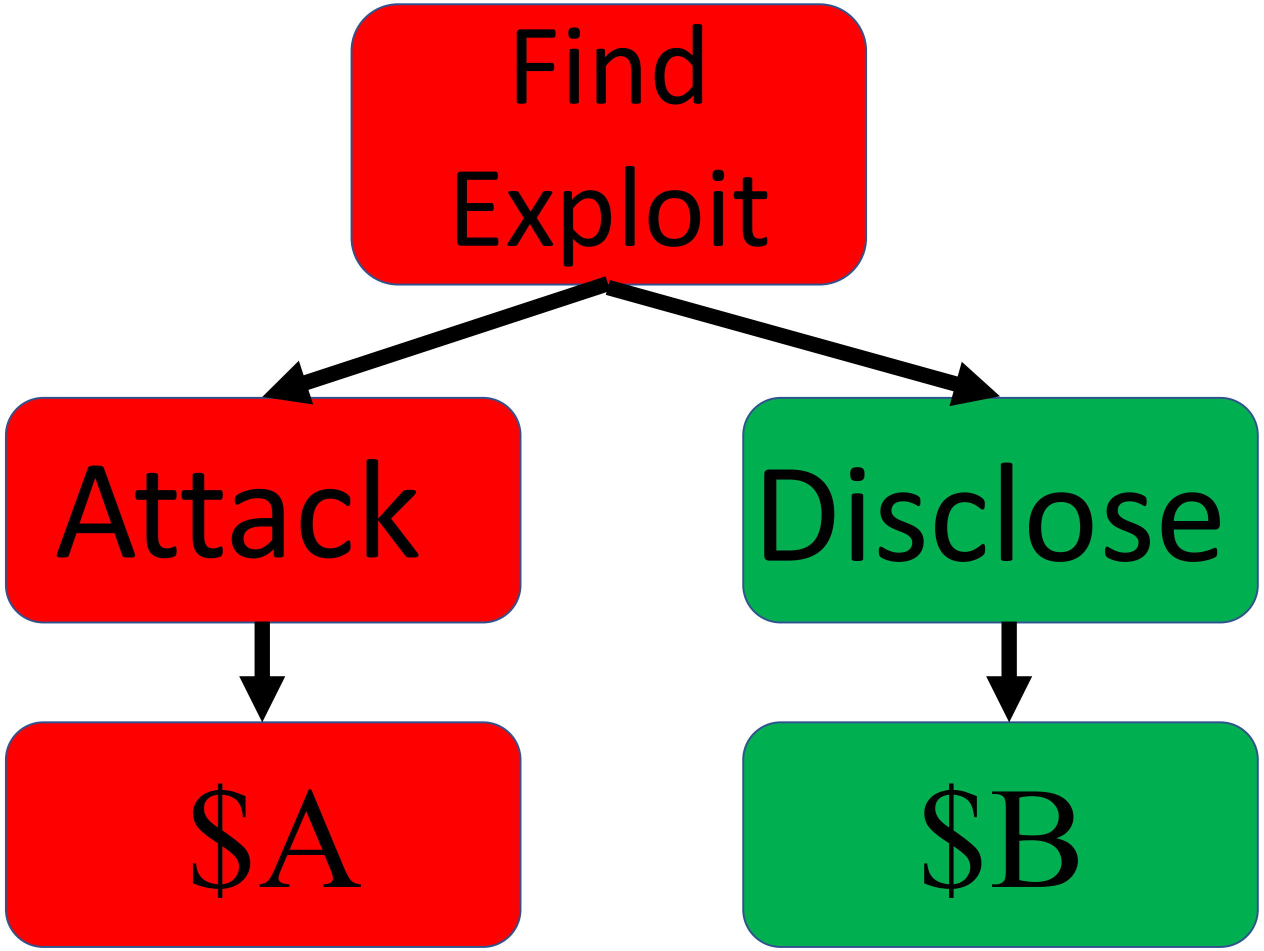
3. Bounty administrator doesn’t doesn’t always pay!

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

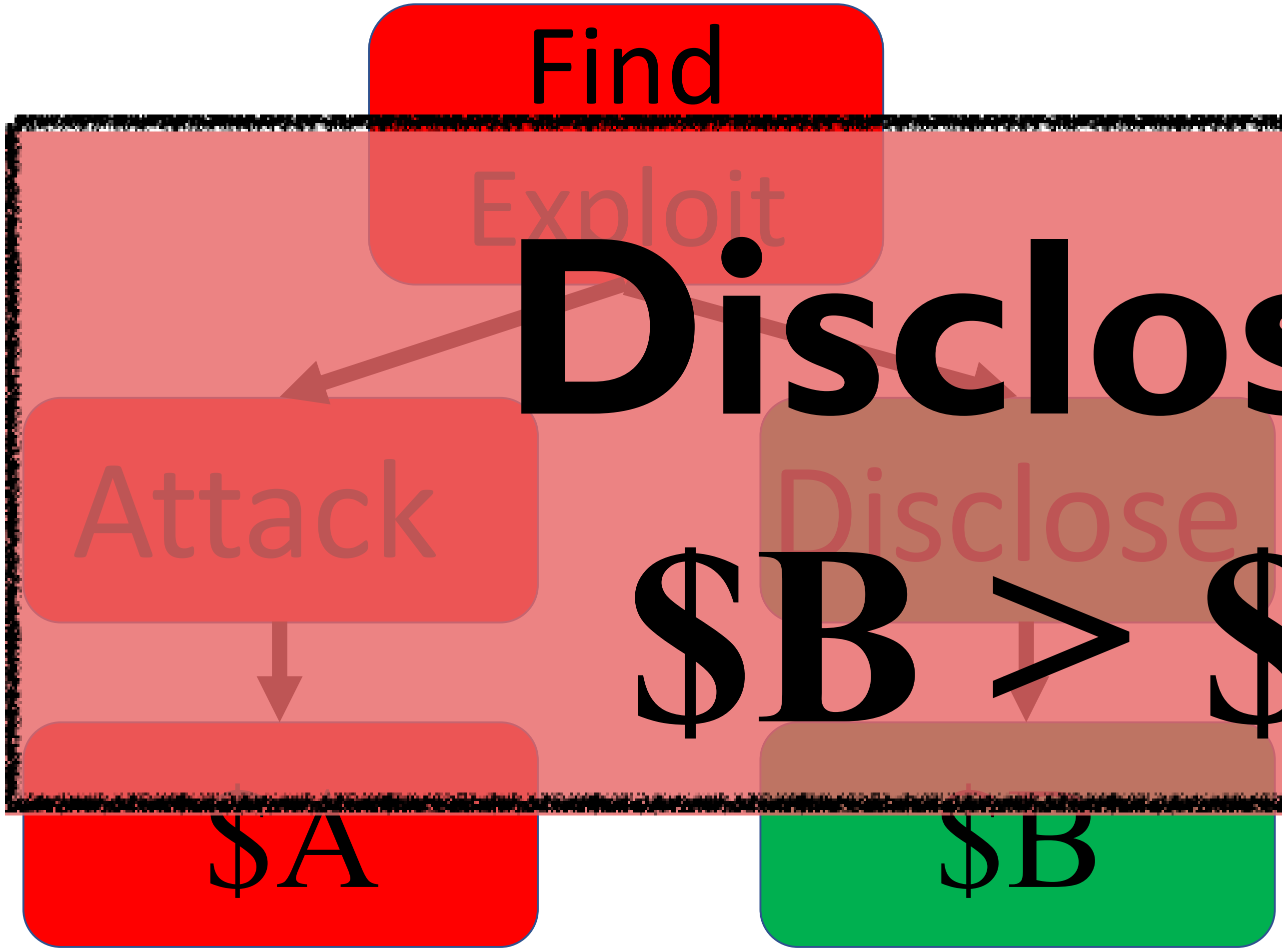
Bug bounties: The Rational Attacker's Game

Bug bounties: The Rational Attacker's Game



Classic bounty: $B

Bug bounties: The Rational Attacker's Game

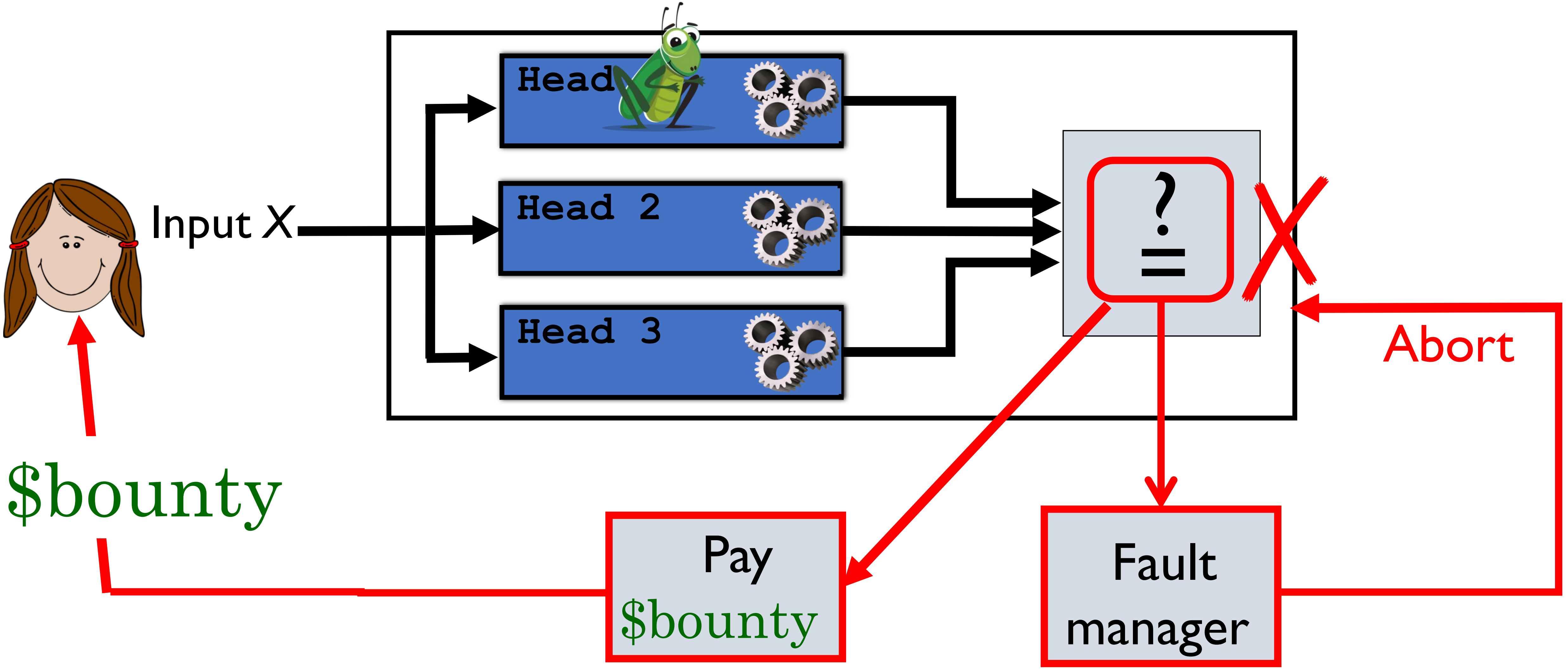


Disclose if

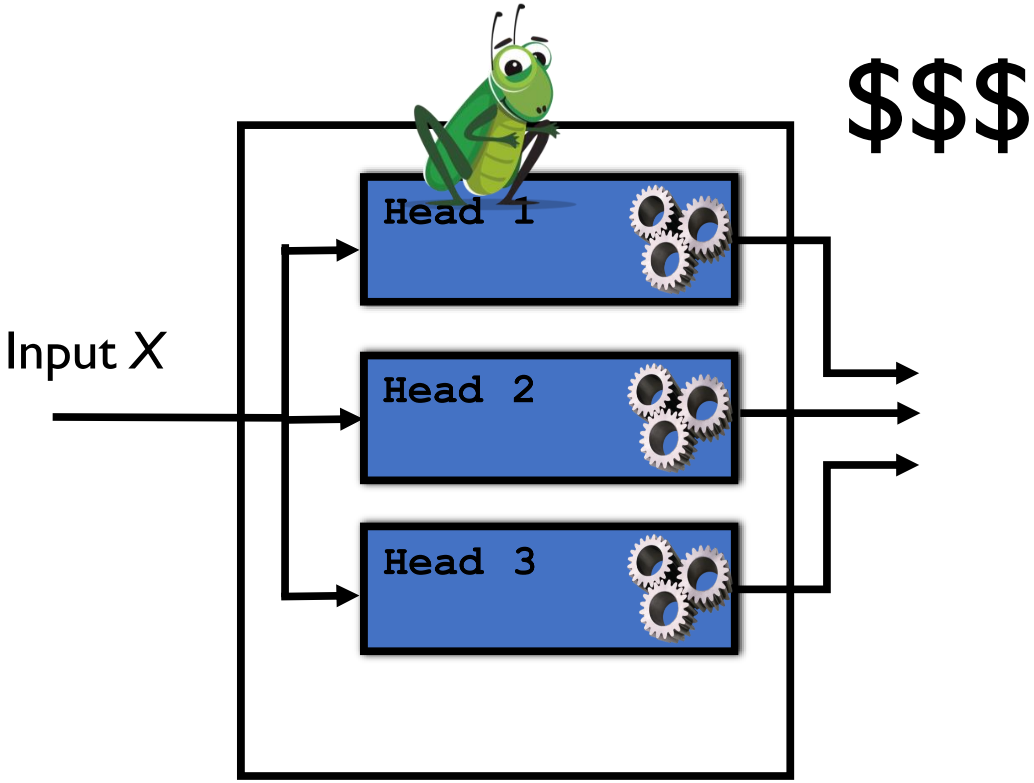
$B > $A!

Classic bounty: $B

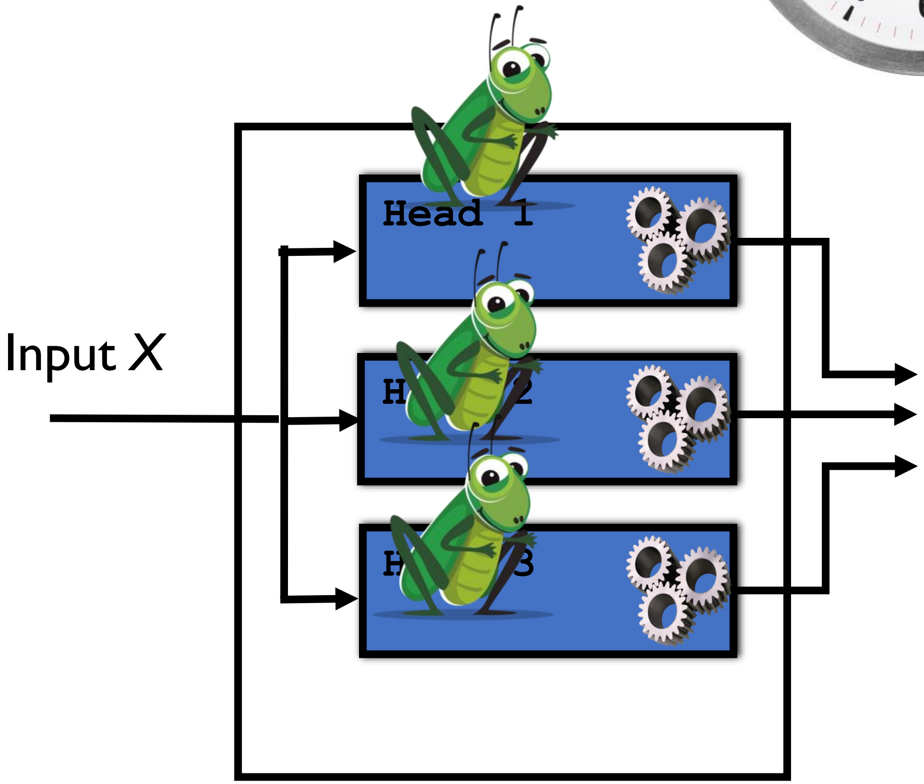
The Hydra Framework for Bug Bounties



The Hydra Hacker’s Dilemma

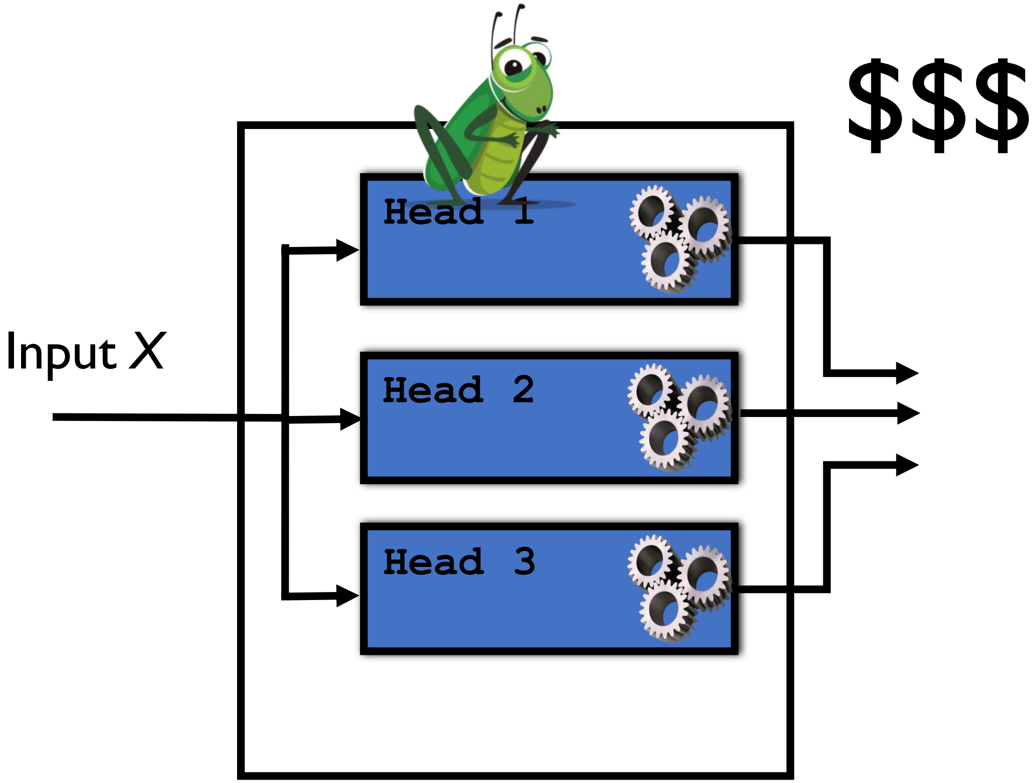


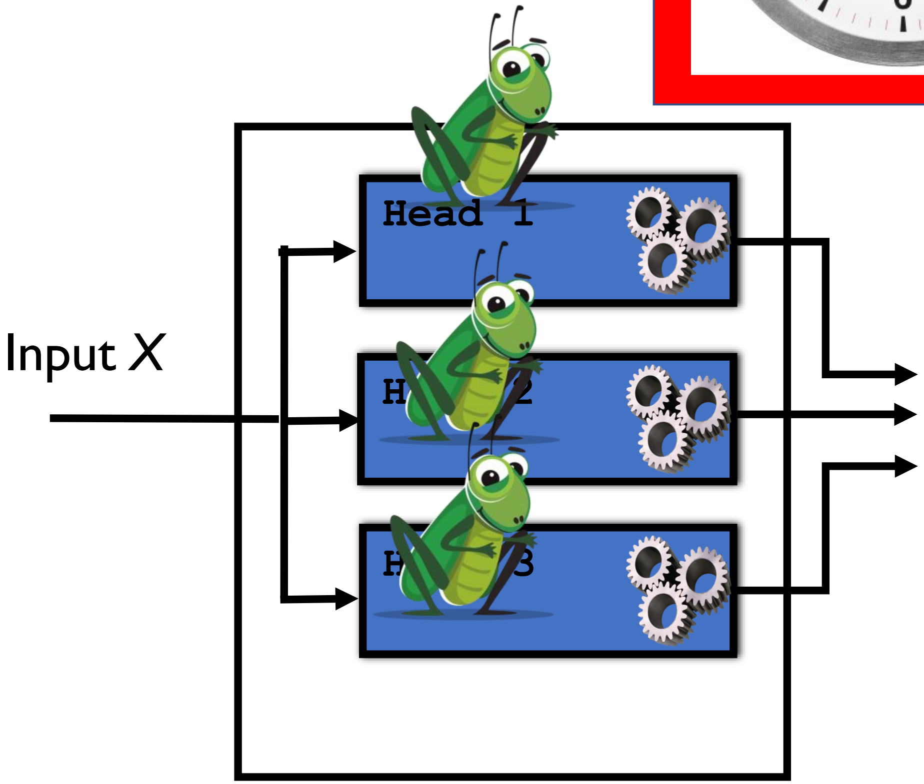
Claim bounty ($B) now?



Try to break all heads ($A)?

The Hydra Hacker's Dilemma

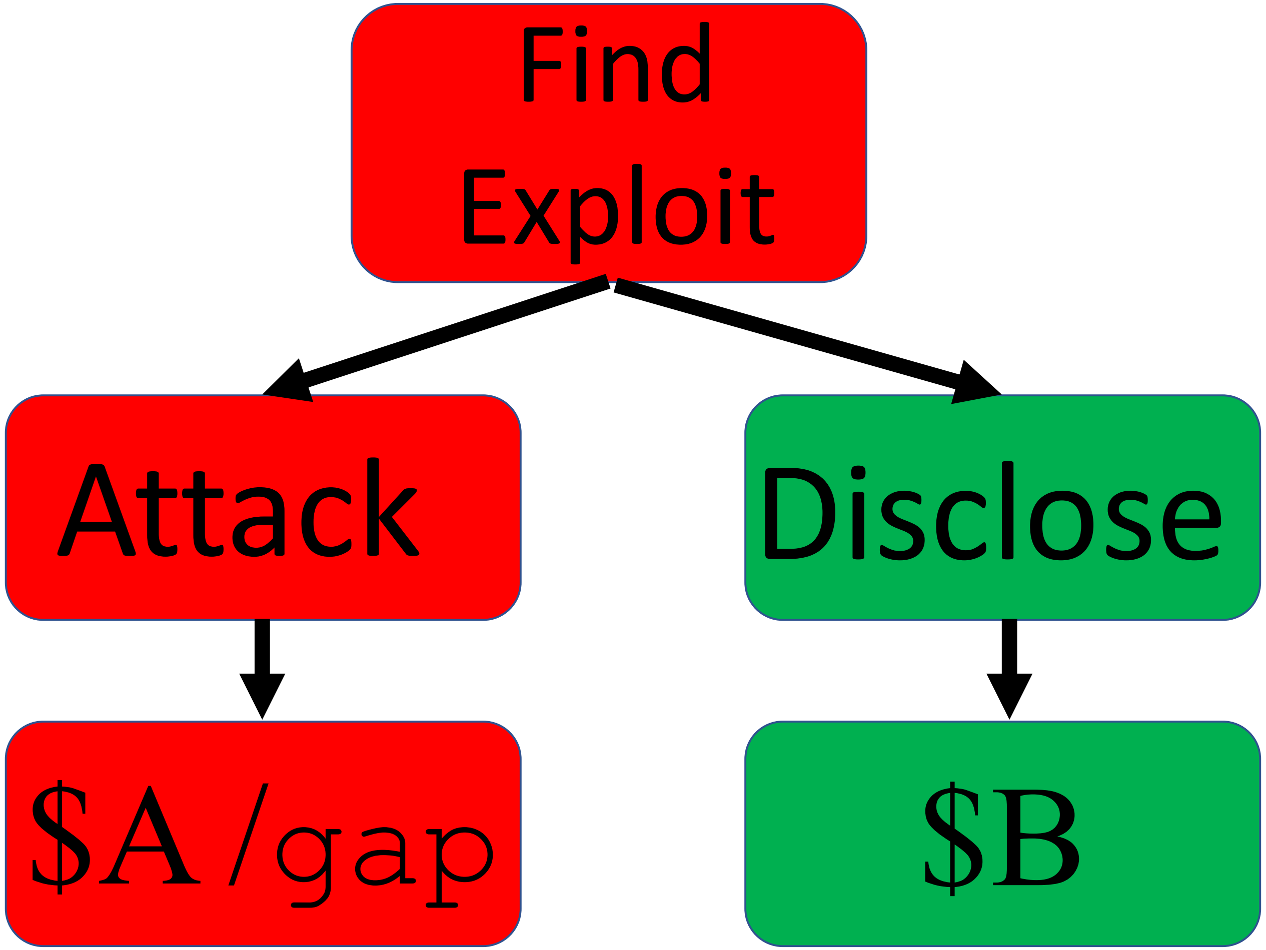




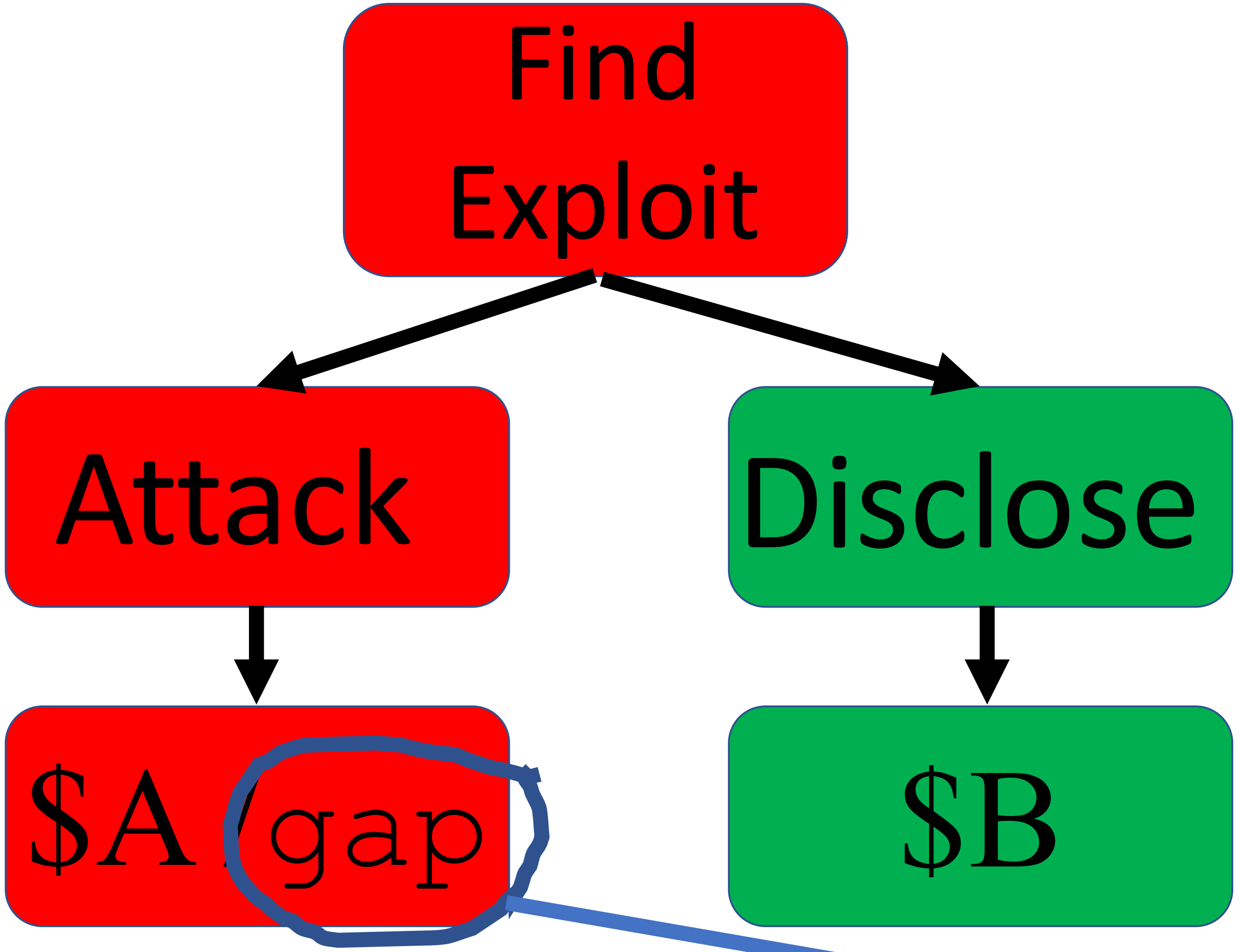
Claim bounty ($B) now?

Try to break all heads ($A)?

Our goal: High leverage



Our goal: High leverage



For gap $\gg 1$

Our goal: High leverage

Find Exploit

Attack

Disclose

$A/gap*

$B

Disclose if

$B > $A / gap!

Exploit gap

Our goal: High leverage

Find Exploit

Attack → $A/gap*

Disclose → $B

Disclose if

$B > $A / (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:
  - `gap` = 4,409 for *N* = 3 heads
  - `gap` = 34,546 for *N* = 4 heads
- So…
  - **Approx $1 billion** contract (e.g., OmiseGo)
  - *N* = 4
  - **$30k** $bounty incentivizes adversary to disclose!

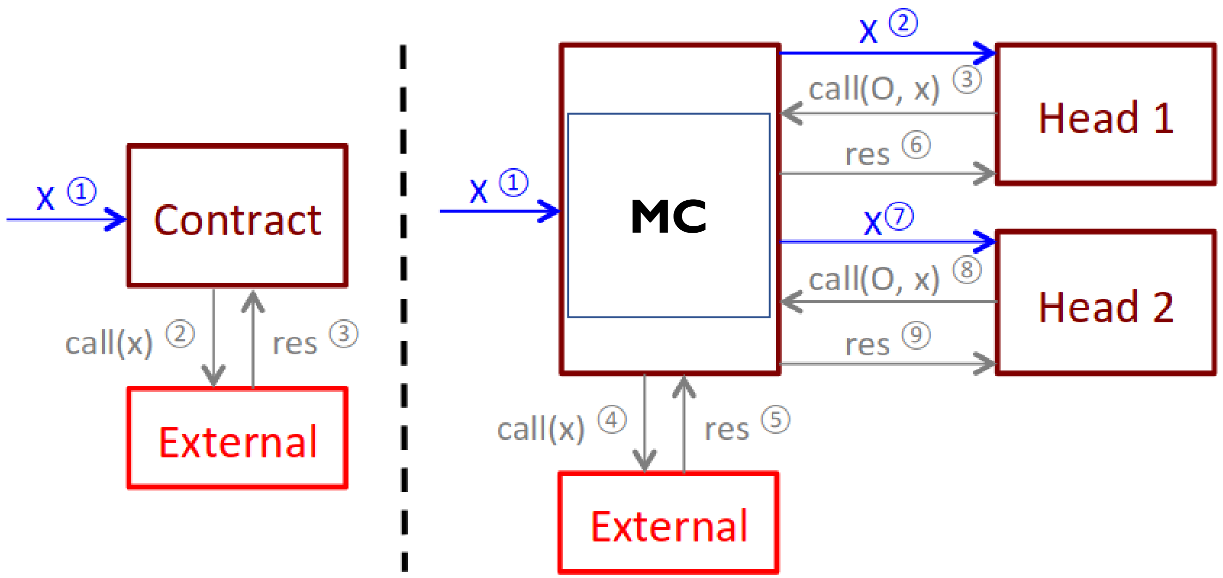
The perfect bug bounty

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
  - $bounty = 3ETH ~= $1k
  - **Deployed @** 0xf4ee935a3879ff0736251 4da69c64df80fa28622
- 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

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

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

⇓ (⇑)

```
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
```

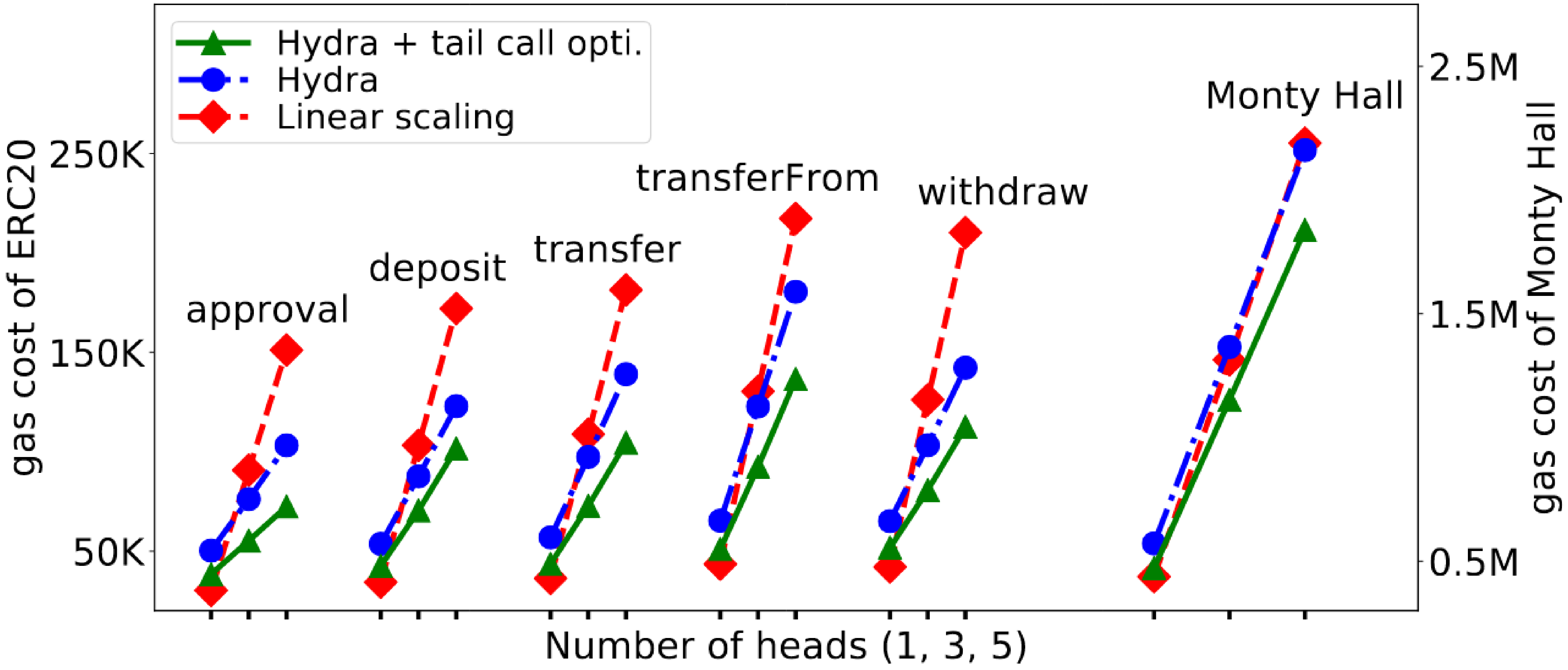
Instrumenter: EVM -> EVM transpiler

Is it practical?

Opcode	Contracts		Transactions		Difficulty
CODECOPY	50,147	(14%)	5,646,607	(27%)	medium
CALLCODE	30,109	(8%)	1,213,064	(6%)	hard
SELFDESTRUCT	24,707	(7%)	739,249	(4%)	easy
DELEGATECALL	19,749	(6%)	2,695,326	(13%)	hard
CREATE	11,559	(3%)	1,143,961	(5%)	easy
Other	6681	(2%)	195,569	(1%)	-
None	268,652	(76%)	12,780,929	(61%)	supported

Blocks 4690101 to 5049100
(Dec-07-2017 -- Feb-07-2018)

Does it scale?



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

$\mathcal{F}_{\mathsf{withhold}}$ with $\mathcal{P} = \{P_0, P_1, \ldots, P_m\}$, (δ, ρ)-adversary $\mathcal{A}$, blocksize s, target height n

Init: $\mathcal{B} \leftarrow \emptyset$, $\mathcal{B}.\mathsf{Height} \leftarrow 0$, $\mathsf{MaxHeight} \leftarrow 0$, $\mathsf{Mempool} \leftarrow \emptyset$

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

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

On receive ("rewind", r) from $\mathcal{A}$ // $\mathcal{A}$ rewinds by r blocks
  assert $\mathsf{MaxHeight} - (\mathcal{B}.\mathsf{Height} - r) \leq \rho$
    // Ensure that $\mathcal{A}$ rewinds by no more than ρ
  $\mathsf{Mempool} \leftarrow \mathsf{Mempool} \bigcup \{B_i\}_{i \in [\mathcal{B}.\mathsf{Height} - r + 1, \mathcal{B}.\mathsf{Height}]}$
    // Return rewound transactions to Mempool
  $\mathcal{B}.\mathsf{Height} \leftarrow \mathcal{B}.\mathsf{Height} - r$

Figure 2: Ideal functionality $\mathcal{F}_{\mathsf{withhold}}$ for (δ, ρ)-adversary $\mathcal{A}$

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! :))

www.thehydra.io

Initiative for CryptoCurrencies and Contracts (IC3)

IC3 The Initiative For CryptoCurrencies & Contracts

HOME ABOUT PEOPLE PARTNERS PROJECTS PUBLICATIONS BLOGS PRESS EVENTS

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.

www.initc3.org

Thanks!

thehydra.io

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 -