# Enter Hydra

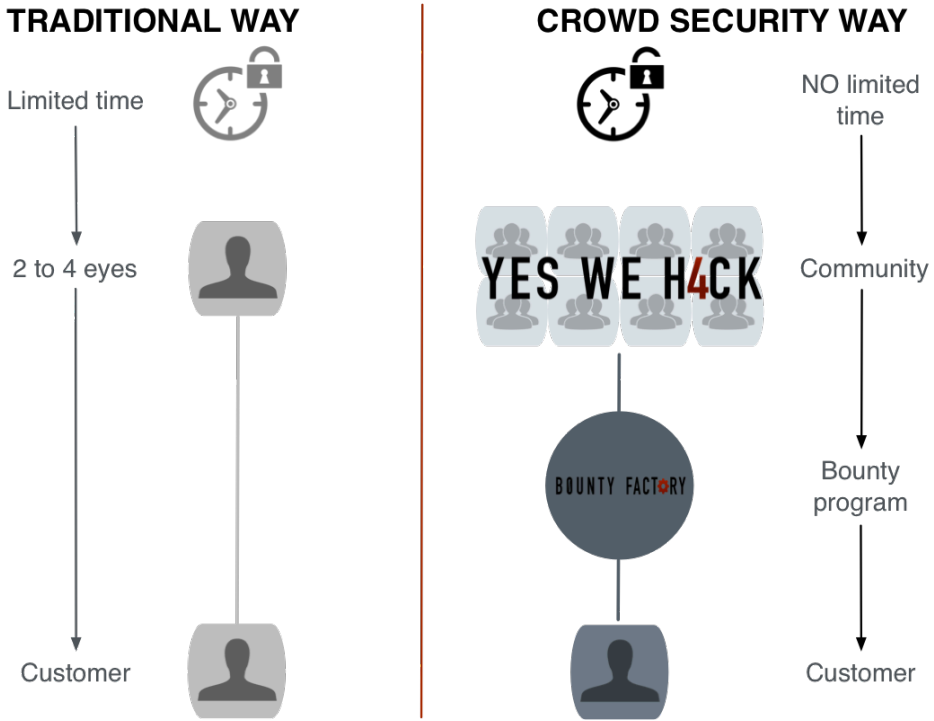
*towards (more) secure smart contracts*


Philip Daian, Ari Juels
*Cornell [Tech]*

Florian Tramer
*Stanford*

Lorenz Breidenbach
*ETH Zurich, Cornell [Tech]*

# Bug bounties



# Problems with Bug bounties

- Unaligned incentives (exploit $$$ > bounty $)
- Time lag between reporting and action
- No fair exchange: bounty admin may not pay!

# Problems with Bug bounties

- U
- T
- N

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

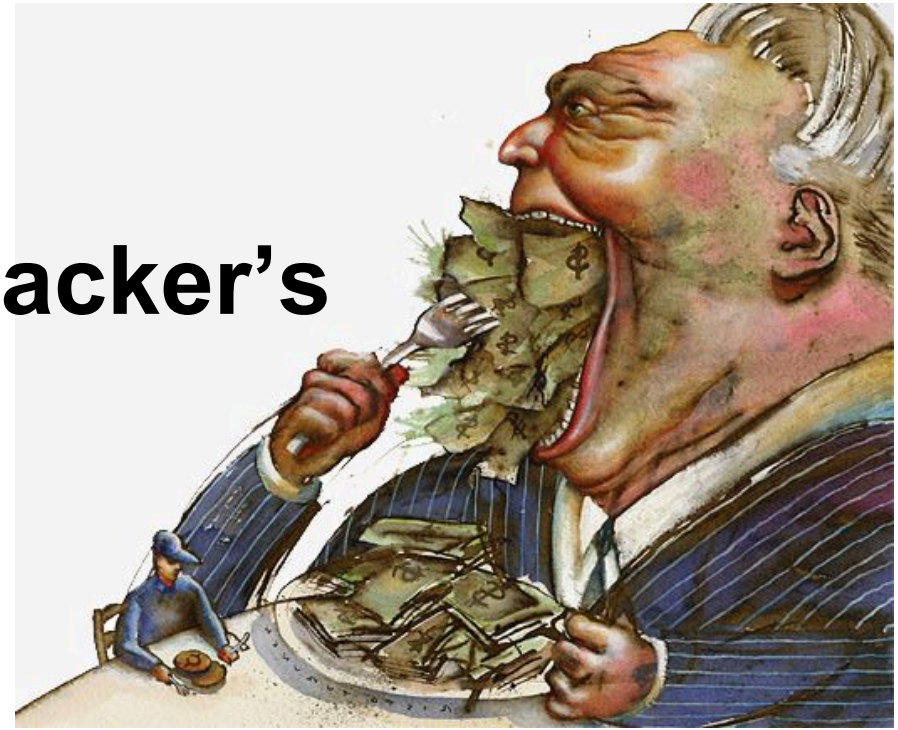
By Ionut Arghire on October 31, 2016

# The perfect bug bounty

1. **“Strong exploit gap”**: Small bounty incentivizes disclosure for valuable program
2. **Automatic remediation:** Immediate intervention in affected software
3. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable

# Why bug bounties?

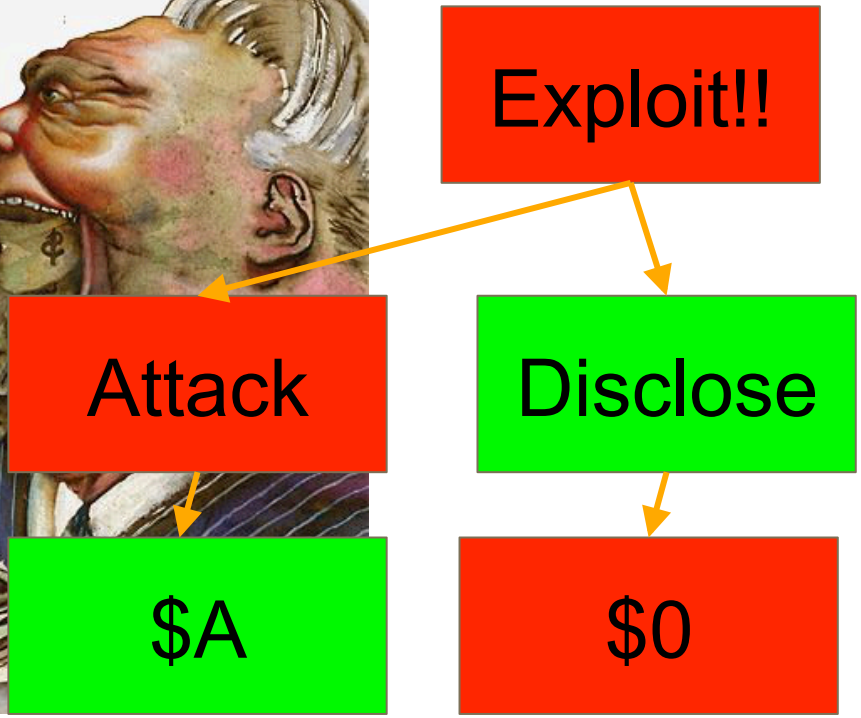
**The rational attacker's game**



# Why bug bounties?

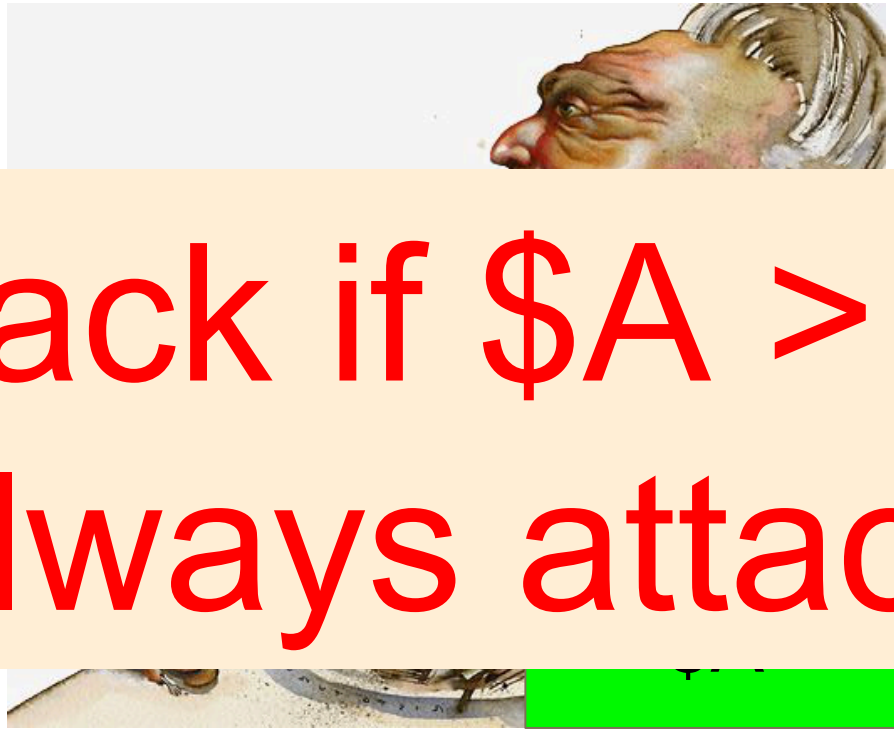
**The rational attacker's game**

**No bounties**



# Why bug bounties?

**The**
**ration**
**game**
**No bo**



Exploit!!

close

0

Attack if $A > $0

Always attack

# “Good enough” isn’t good enough

## The rational attacker’s game

Classic bounty
Unknown payout

Exploit!!

Attack

Disclose

$A

$??

# “Good enough” isn’t good enough

**The**
**ratio**
**game**

Attack if $A > $??

Classic bounty
Unknown payout

Exploit!!

se

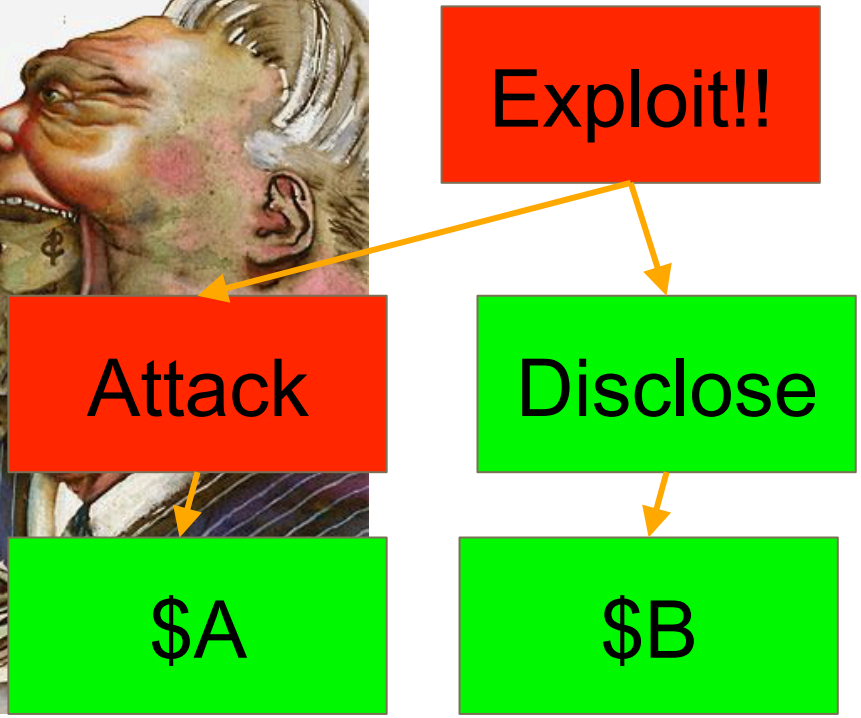
$A

$??

# Towards a better game

## The rational attacker's game

Classic bounty
Known payout



# Towards a better game

**The**
**ratio**
**game**

**Classic bounty**
Known payout

Exploit!!

Attack if $A > $B

$A

$B

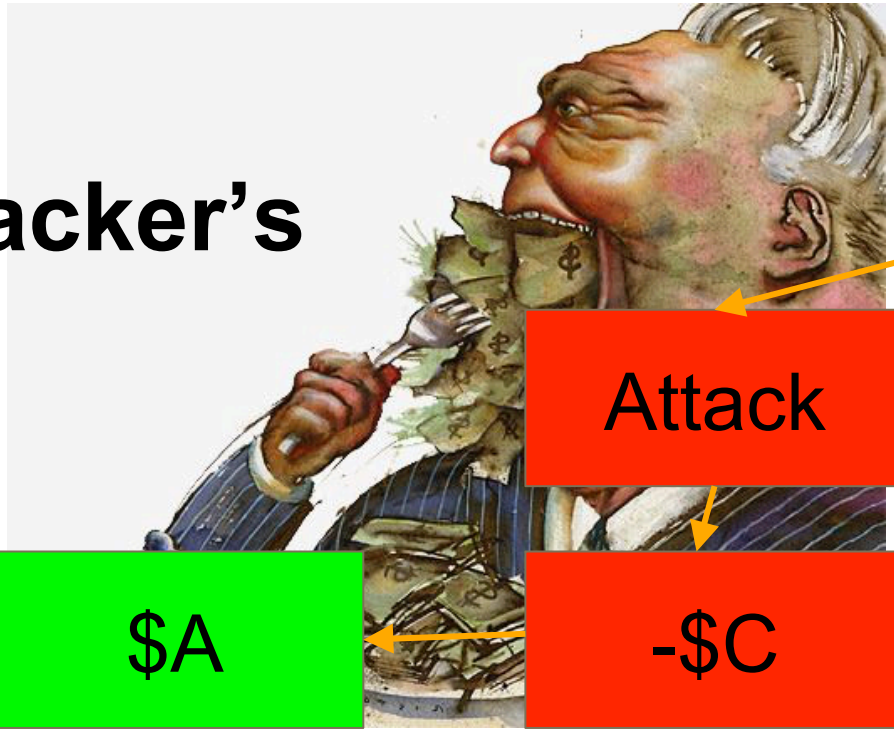
# The ideal game

## The rational attacker's game

**Hydra bounty**

**Known payout**

**Gap to exploit**



Exploit!!

Attack

Disclose

$A

-$C

$B

# The ideal game

**The**
**ra**
**ga**
**Hydra bounty**
**Known payout**
**Gap to exploit**

Exploit!!

Attack if $A-$C > $B

$A

-$C

$B

# The ideal game

**The**
**ra**
**ga**
**Hy**
**Kn**
**Ga**

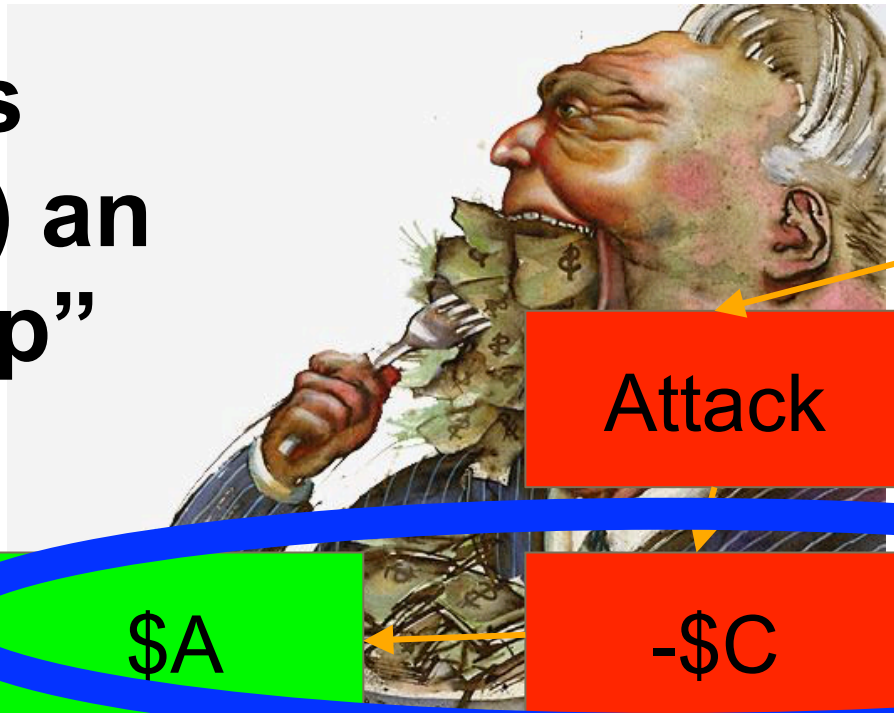
Exploit!!

Attack if $A-$C > $B

*So, raise $C….*

# … mind the gap!

**We call this barrier ($C) an "exploit gap"**



Exploit!!

Attack

Disclose

$A

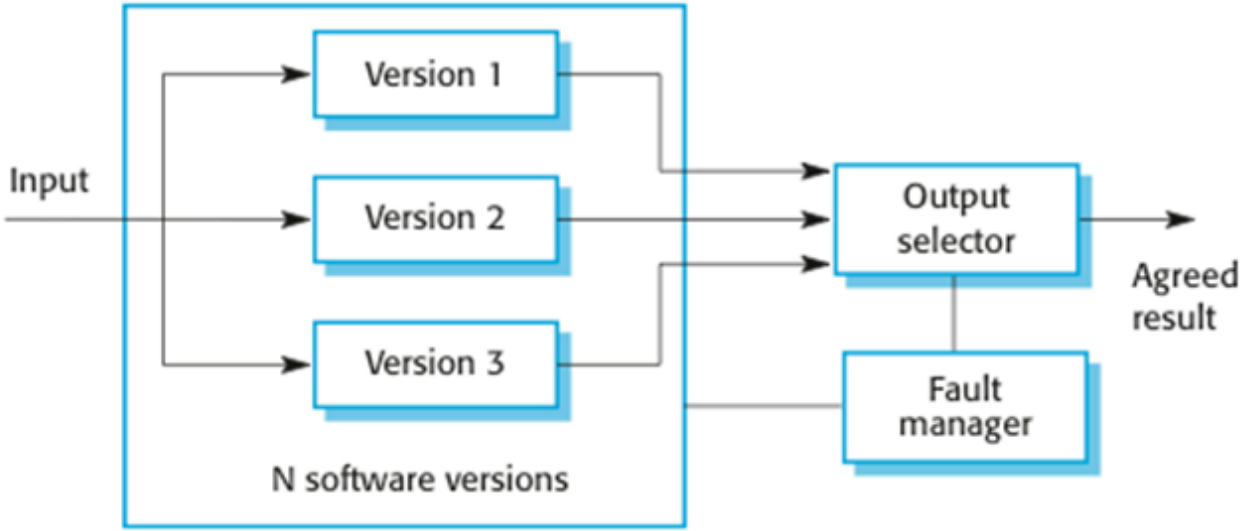
-$C

$B

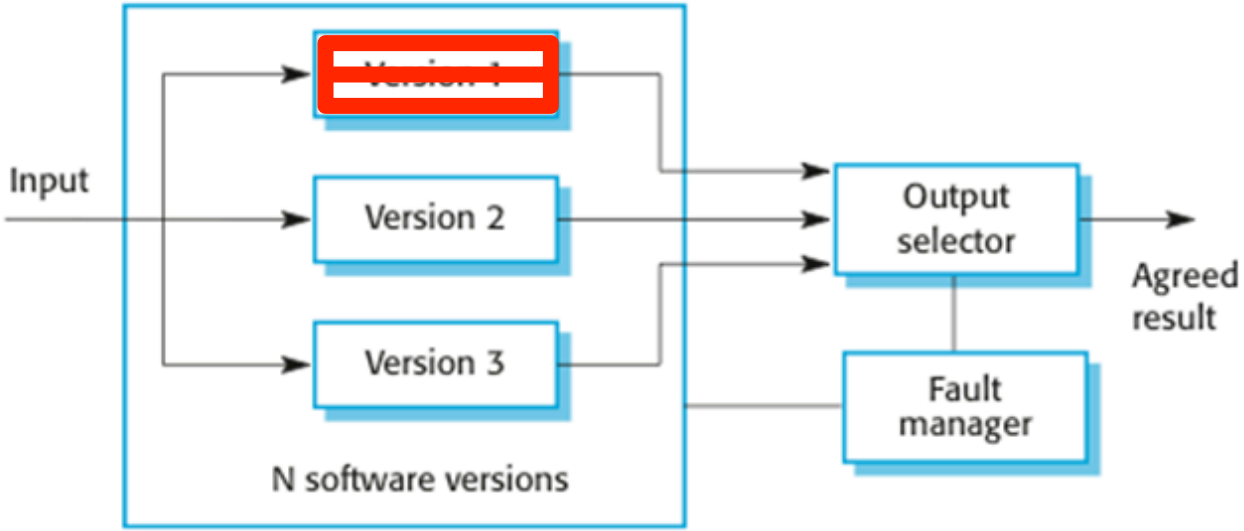

BACK
TO
THE FUTURE
A ROBERT ZEMECKIS FILM
DOCTAVINE

# Exploit Gap through Hydra Contracts

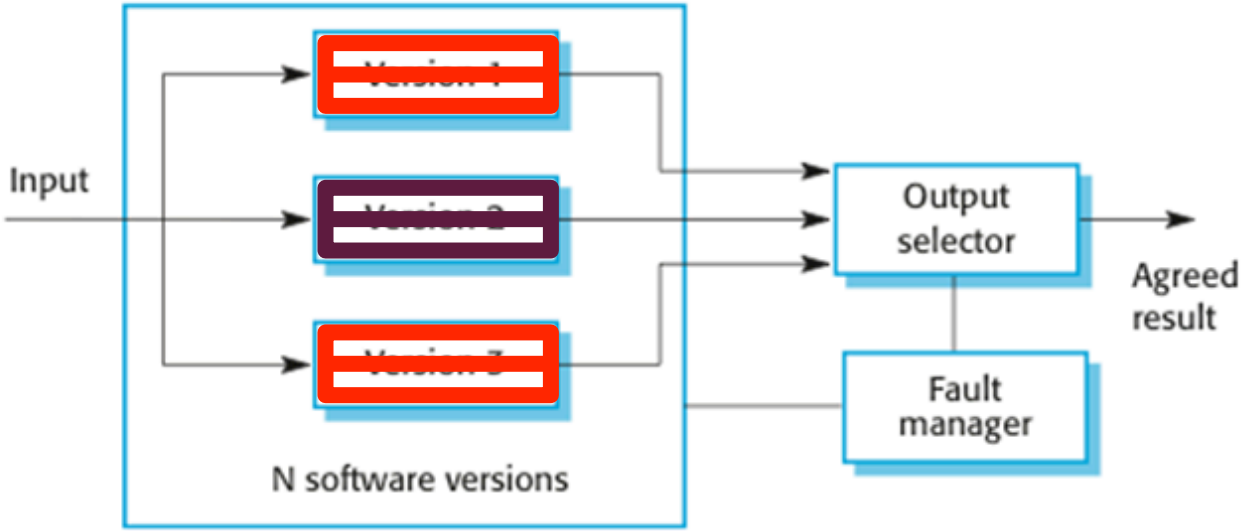
Chen & Avizienis, ‘78



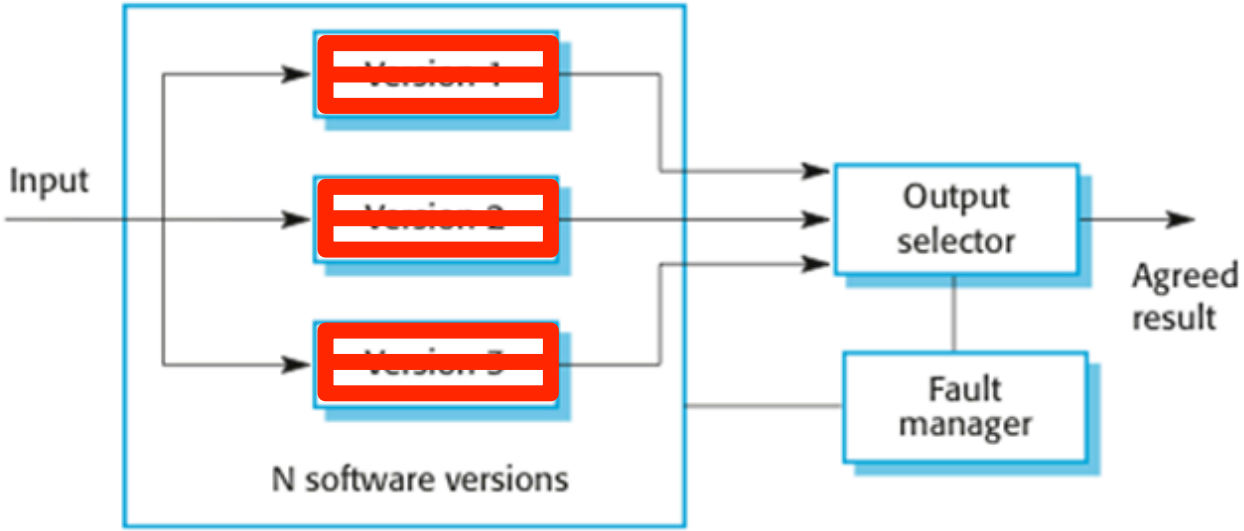

# … Houston we have a gap (only one contract has bug)



# … Houston we have a gap (contracts have different bugs)

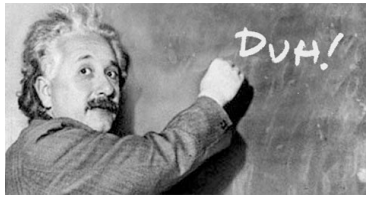


# … Houston we have no gap! Hydra fails! (all contracts have same bug)



# N-Version Programming Criticism

- Analysis assumes full independence of faults (correlations are annoying!)
- Knight-Leveson (‘86):
  « We reject the null hypothesis of full independence at a p-level of 5% »




- Eckhardt et al. (’91):
  « 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!

- «Classical» N-Version Programming: **Availability >> Reliability**
  - **Majority Voting**: Always available, but may fail often

- Smart contracts: do we really car if it's down for a while?
  - N-out-of-N agreement: ***better no answer than the wrong one***

- *Numbers from Eckhardt et al. look much better:*
  - *For 3 versions,* **30 − 5087** times fewer failures (but some loss in availability…)

# The perfect bug bounty

1. **“Strong exploit gap”**: Small bounty incentivizes disclosure for valuable program
2. **Automatic remediation:** Immediate intervention in affected software
3. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable

# Target Application: Smart Contracts

# Smart contracts are the perfect target

- Small programs with astonishing value per line of code

| Token | Lines of Code | Value per line |
|---|---|---|
| OmiseGo | 396 | ~$1.59M |
| Tether | 423 | ~$1.11M |
| EOS | 584 | ~$1.01M |

*Sources: coinmarketcap.com, 3 Nov., 8:20 a.m. and published contract source code*

- Hydra friendly bug remediation (return money, put in escrow etc)
- Automatic bounty payment possible
- Bonus: automatic assesment of value at risk

# The perfect bug bounty

1. **“Strong exploit gap”**: Small bounty incentivizes disclosure for valuable program
2. **Automatic remediation:** Immediate intervention in affected software
3. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable

# Development Challenges

- Coordinating multiple smart contracts:
  - The coordinator should be bug free => simple proxy behavior
  - Maintain consistent blockchain state
  - How to recover from a discovered bug => escape hatches

- Frontrunning (as always…)
  - Attacker can break the exploit gap by *witholding* bugs
  - Search for full exploit until someone tries to claim a bounty
  - Solution: Submarine sends!
    http://hackingdistributed.com/2017/08/28/submarine-sends/

# Bug Withholding and Commit-Reveal

Sol 1: To claim bounty at time T, must *commit to bug* at time T- 1

Problem: Attacker commits in every round and only reveals if someone else does

Sol 2: To commit, you must pay $$ (in a verifiable way)

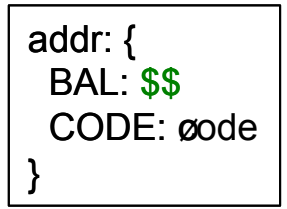
Problem: Attacker commits if someone else also commits

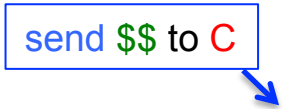
Sol 3: Hide commitments (e.g., proof of burn to random address)

Problem: Wasteful

# Submarine Sends (post-metropolis version)

Goals: (1) only allow *committed* users to send a transaction to *C*
(2) being *eternally committed* is expensive
(3) attacker *can't know* if someone has committed
(4) money isn't wasted




Submarine sends:

Phase 1: compute *addr* = *H(C* || *nonce* || *code)* and send $$ to *addr*

Phase 2: reveal *addr* to *C*.

*C verifies that addr got $$ in Phase 1*

*C* creates a contract with the specified nonce and code

*C collects $$ and allows transaction*


# www.thehydra.io