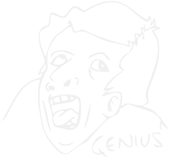
UNIVERSITÀ DI TRENTO

ProSVED
Projection of Security Vulnerabilities caused by Exploits in Dependencies

Formal and smart contracts—or maybe not

Not a presentation of results but of ~~terrible problems~~ opportunities



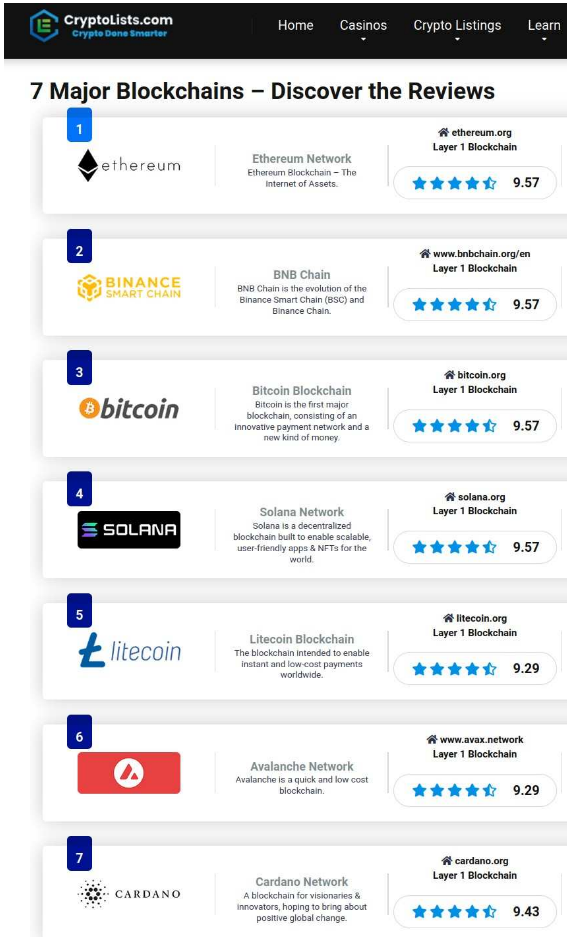
Carlos E. Budde

Security group @ Dipartimento di Ingegneria e Scienza dell'Informazione

carlosesteban.budde@unitn.it

Smartitude — 23.10.2023

Blockchain... what? which? why?

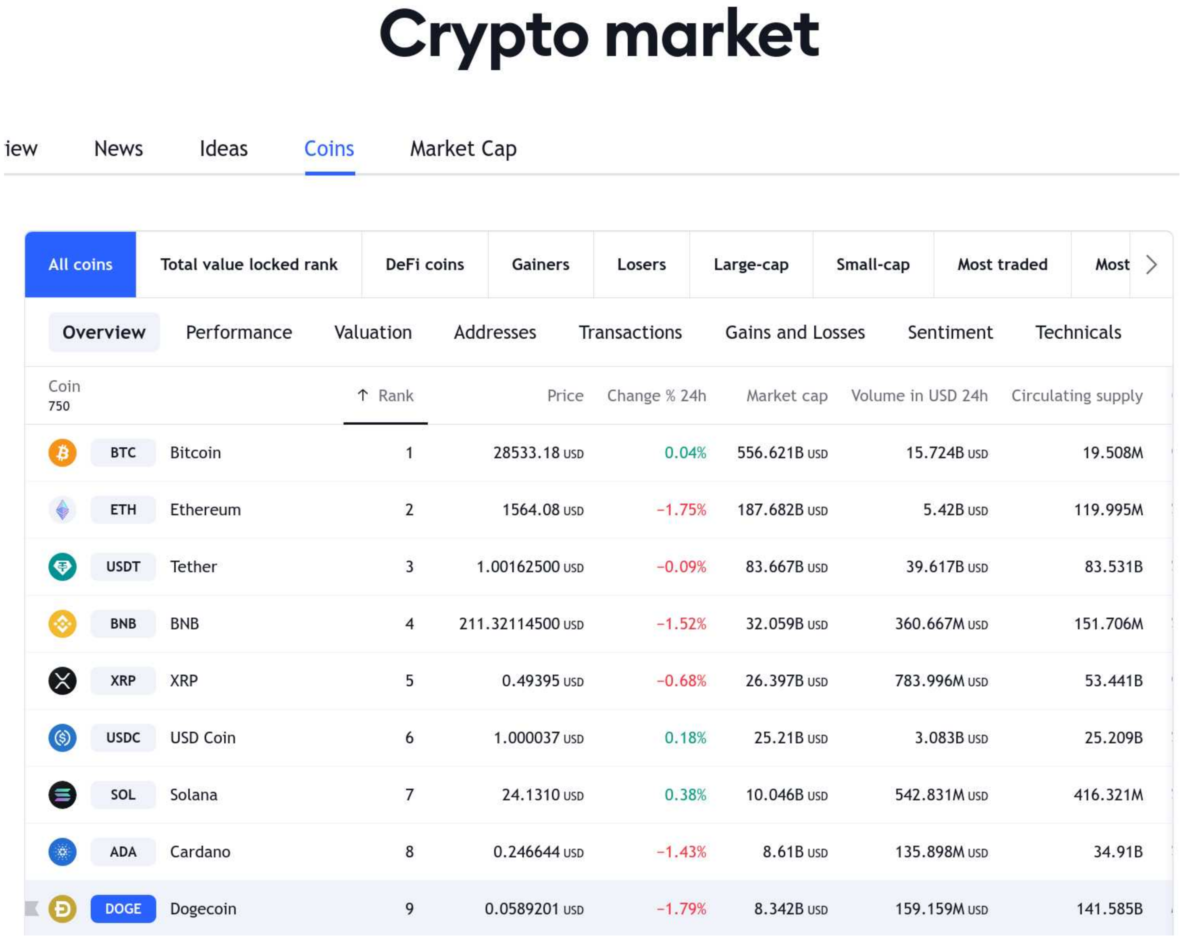


CryptoLists.com
Crypto Done Smarter

Home | Casinos | Crypto Listings | Learn

7 Major Blockchains – Discover the Reviews

1. **Ethereum Network** – Ethereum Blockchain – The Internet of Assets. – ethereum.org – Layer 1 Blockchain – ★★★★☆ 9.57
2. **BNB Chain** – BNB Chain is the evolution of the Binance Smart Chain (BSC) and Binance Chain. – www.bnbchain.org/en – Layer 1 Blockchain – ★★★★☆ 9.57
3. **Bitcoin Blockchain** – Bitcoin is the first major blockchain, consisting of an innovative payment network and a new kind of money. – bitcoin.org – Layer 1 Blockchain – ★★★★☆ 9.57
4. **Solana Network** – Solana is a decentralized blockchain built to enable scalable, user-friendly apps & NFTs for the world. – solana.org – Layer 1 Blockchain – ★★★★☆ 9.57
5. **Litecoin Blockchain** – The blockchain intended to enable instant and low-cost payments worldwide. – litecoin.org – Layer 1 Blockchain – ★★★★☆ 9.29
6. **Avalanche Network** – Avalanche is a quick and low cost blockchain. – www.avax.network – Layer 1 Blockchain – ★★★★☆ 9.29
7. **Cardano Network** – A blockchain for visionaries & innovators, hoping to bring about positive global change. – cardano.org – Layer 1 Blockchain – ★★★★☆ 9.43



Crypto market

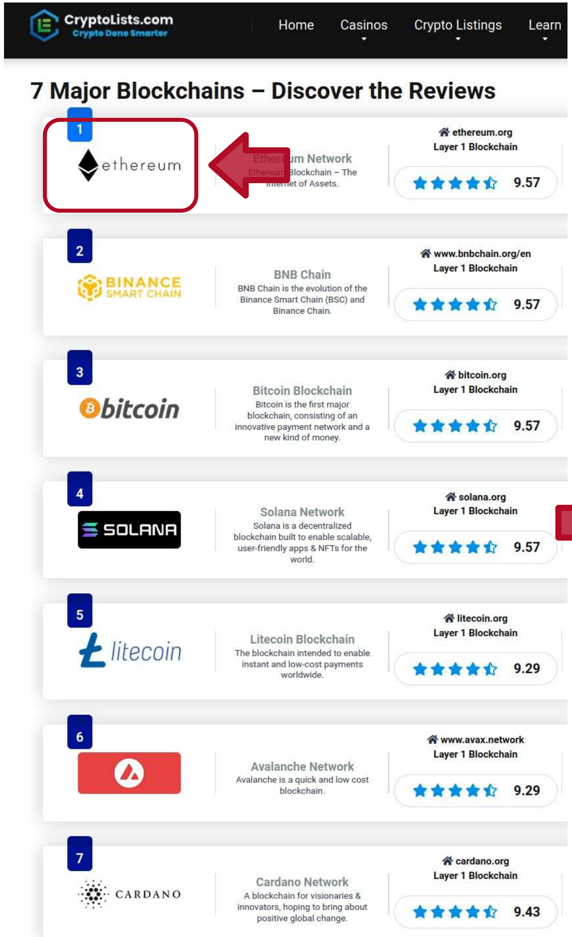
iew | News | Ideas | Coins | Market Cap

All coins | Total value locked rank | DeFi coins | Gainers | Losers | Large-cap | Small-cap | Most traded | Most

Overview | Performance | Valuation | Addresses | Transactions | Gains and Losses | Sentiment | Technicals

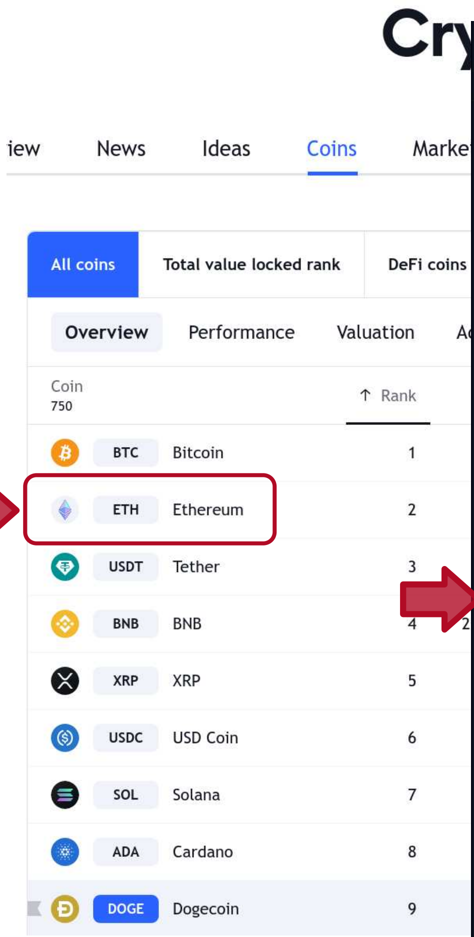
Coin 750		↑ Rank	Price	Change % 24h	Market cap	Volume in USD 24h	Circulating supply
BTC	Bitcoin	1	28533.18 USD	0.04%	556.621B USD	15.724B USD	19.508M
ETH	Ethereum	2	1564.08 USD	−1.75%	187.682B USD	5.42B USD	119.995M
USDT	Tether	3	1.00162500 USD	−0.09%	83.667B USD	39.617B USD	83.531B
BNB	BNB	4	211.32114500 USD	−1.52%	32.059B USD	360.667M USD	151.706M
XRP	XRP	5	0.49395 USD	−0.68%	26.397B USD	783.996M USD	53.441B
USDC	USD Coin	6	1.000037 USD	0.18%	25.21B USD	3.083B USD	25.209B
SOL	Solana	7	24.1310 USD	0.38%	10.046B USD	542.831M USD	416.321M
ADA	Cardano	8	0.246644 USD	−1.43%	8.61B USD	135.898M USD	34.91B
DOGE	Dogecoin	9	0.0589201 USD	−1.79%	8.342B USD	159.159M USD	141.585B

Blockchain... what? which? why?

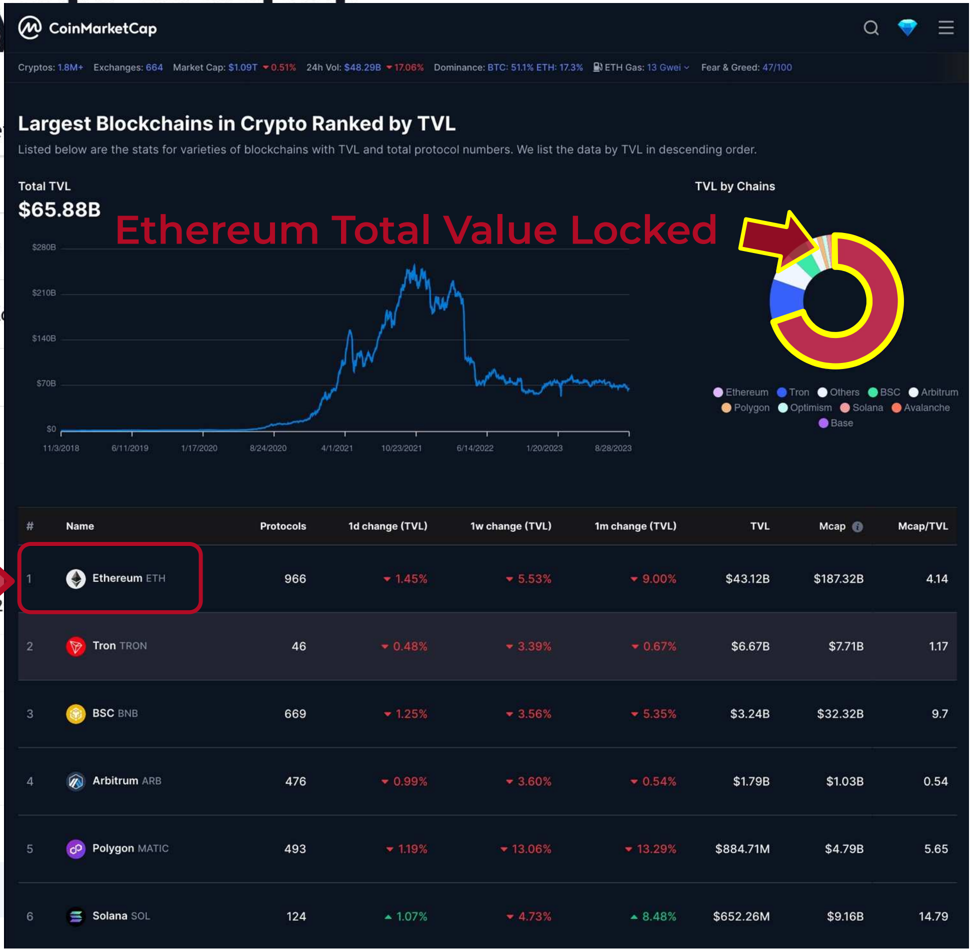


7 Major Blockchains – Discover the Reviews

#	Blockchain	Description	Website	Type	Rating
1	ethereum	**Ethereum Network** Ethereum Blockchain – The Internet of Assets.	ethereum.org	Layer 1 Blockchain	★★★★☆ 9.57
2	BINANCE SMART CHAIN	**BNB Chain** BNB Chain is the evolution of the Binance Smart Chain (BSC) and Binance Chain.	www.bnbchain.org/en	Layer 1 Blockchain	★★★★☆ 9.57
3	bitcoin	**Bitcoin Blockchain** Bitcoin is the first major blockchain, consisting of an innovative payment network and a new kind of money.	bitcoin.org	Layer 1 Blockchain	★★★★☆ 9.57
4	SOLANA	**Solana Network** Solana is a decentralized blockchain built to enable scalable, user-friendly apps & NFTs for the world.	solana.org	Layer 1 Blockchain	★★★★☆ 9.57
5	litecoin	**Litecoin Blockchain** The blockchain intended to enable instant and low-cost payments worldwide.	litecoin.org	Layer 1 Blockchain	★★★★☆ 9.29
6		**Avalanche Network** Avalanche is a quick and low cost blockchain.	www.avax.network	Layer 1 Blockchain	★★★★☆ 9.29
7	CARDANO	**Cardano Network** A blockchain for visionaries & innovators, hoping to bring about positive global change.	cardano.org	Layer 1 Blockchain	★★★★☆ 9.43



Coin 750		↑ Rank
BTC	Bitcoin	1
ETH	Ethereum	2
USDT	Tether	3
BNB	BNB	4
XRP	XRP	5
USDC	USD Coin	6
SOL	Solana	7
ADA	Cardano	8
DOGE	Dogecoin	9



Cryptos: 1.8M+ Exchanges: 664 Market Cap: $1.09T ▼0.51% 24h Vol: $48.29B ▼17.06% Dominance: BTC: 51.1% ETH: 17.3% ETH Gas: 13 Gwei Fear & Greed: 47/100

Largest Blockchains in Crypto Ranked by TVL

Listed below are the stats for varieties of blockchains with TVL and total protocol numbers. We list the data by TVL in descending order.

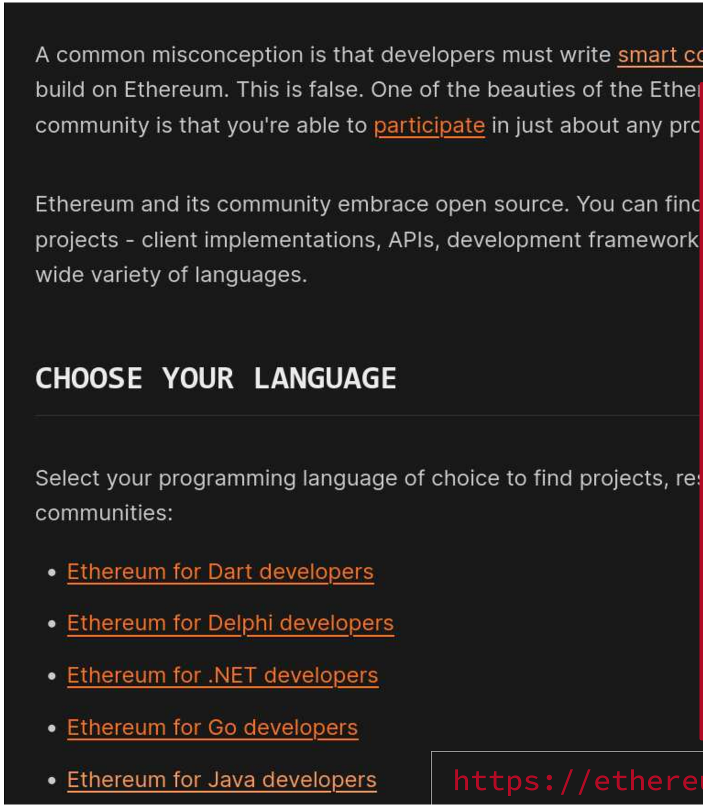
Total TVL
$65.88B

Ethereum Total Value Locked

#	Name	Protocols	1d change (TVL)	1w change (TVL)	1m change (TVL)	TVL	Mcap	Mcap/TVL
1	Ethereum ETH	966	▼1.45%	▼5.53%	▼9.00%	$43.12B	$187.32B	4.14
2	Tron TRON	46	▼0.48%	▼3.39%	▼0.67%	$6.67B	$7.71B	1.17
3	BSC BNB	669	▼1.25%	▼3.56%	▼5.35%	$3.24B	$32.32B	9.7
4	Arbitrum ARB	476	▼0.99%	▼3.60%	▼0.54%	$1.79B	$1.03B	0.54
5	Polygon MATIC	493	▼1.19%	▼13.06%	▼13.29%	$884.71M	$4.79B	5.65
6	Solana SOL	124	▲1.07%	▼4.73%	▲8.48%	$652.26M	$9.16B	14.79

Smart contracts... what? which? why?

Ethereum's Blockchain Smart Contracts (BSC) in Solidity



A common misconception is that developers must write smart contracts in order to build on Ethereum. This is false. One of the beauties of the Ether community is that you're able to participate in just about any pro

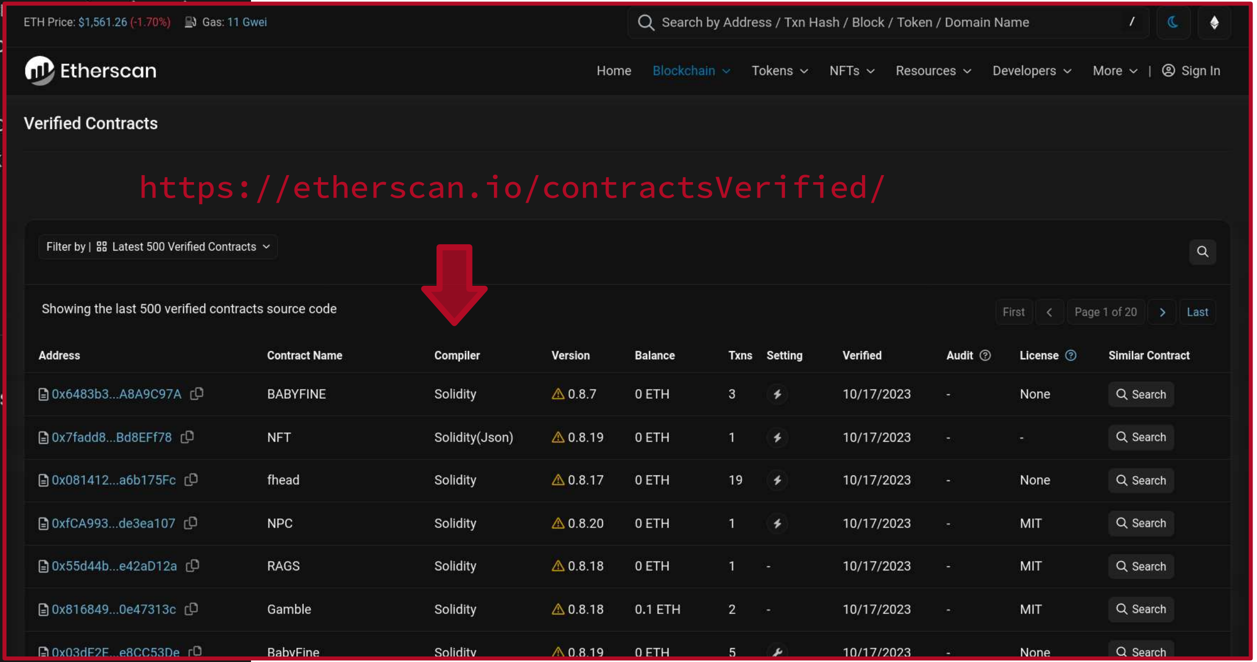
Ethereum and its community embrace open source. You can find projects - client implementations, APIs, development framework wide variety of languages.

CHOOSE YOUR LANGUAGE

Select your programming language of choice to find projects, re communities:

- Ethereum for Dart developers
- Ethereum for Delphi developers
- Ethereum for .NET developers
- Ethereum for Go developers
- Ethereum for Java developers

https://ethereum.org/en/developers/docs/programming-languages/



ETH Price: $1,561.26 (-1.70%)
Gas: 11 Gwei
Search by Address / Txn Hash / Block / Token / Domain Name
Etherscan
Home
Blockchain
Tokens
NFTs
Resources
Developers
More
Sign In

Verified Contracts

https://etherscan.io/contractsVerified/

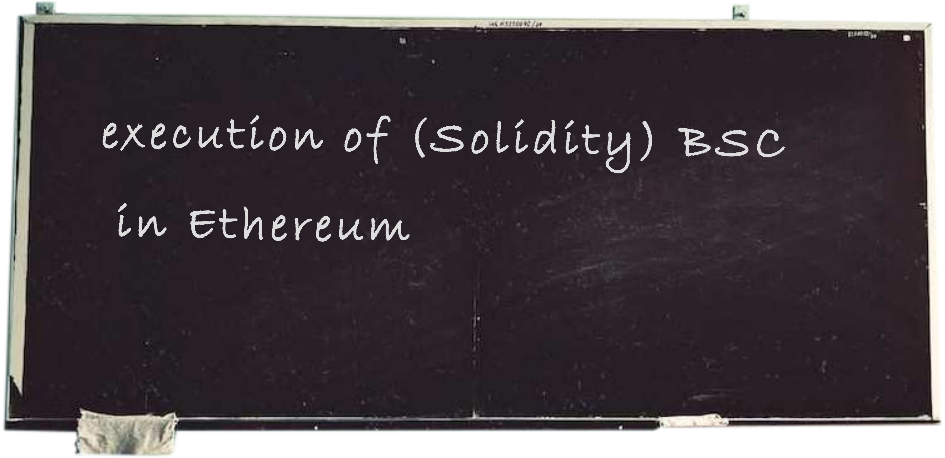
Filter by | Latest 500 Verified Contracts

Showing the last 500 verified contracts source code

Address	Contract Name	Compiler	Version	Balance	Txns	Setting	Verified	Audit	License	Similar Contract
0x6483b3...A8A9C97A	BABYFINE	Solidity	⚠ 0.8.7	0 ETH	3	⚡	10/17/2023	-	None	Search
0x7fadd8...Bd8EFf78	NFT	Solidity(Json)	⚠ 0.8.19	0 ETH	1	⚡	10/17/2023	-	-	Search
0x081412...a6b175Fc	fhead	Solidity	⚠ 0.8.17	0 ETH	19	⚡	10/17/2023	-	None	Search
0xfCA993...de3ea107	NPC	Solidity	⚠ 0.8.20	0 ETH	1	⚡	10/17/2023	-	MIT	Search
0x55d44b...e42aD12a	RAGS	Solidity	⚠ 0.8.18	0 ETH	1	-	10/17/2023	-	MIT	Search
0x816849...0e47313c	Gamble	Solidity	⚠ 0.8.18	0.1 ETH	2	-	10/17/2023	-	MIT	Search
0x03dF2E...e8CC53De	BabyFine	Solidity	⚠ 0.8.19	0 ETH	5	⚡	10/17/2023	-	None	Search

Common vulnerabilities

Security vulnerabilities in BSC written in Solidity



execution of (Solidity) BSC

in Ethereum

Common vulnerabilities

Security vulnerabilities in BSC written in Solidity

`https://dasp.co/`

- Reentrancy
- Access Control
- Arithmetic
- Unchecked external call
- Denial of Service
- Bad Randomness
- Front Running
- Time Manipulation
- Short Addresses

`https://github.com/crytic/not-so-smart-contracts`

- Bad randomness
- Denial of service
- Forced Ether reception
- HoneyPots
- Incorrect interface
- Integer overflow
- Race condition
- Reentrancy
- Unchecked external call
- Unprotected function
- Variable shadowing
- Wrong constructor name

Example 1

Unchecked external call

Certain Solidity operations known as “external calls”, require the developer to check that the operation succeeded—in contrast to operations which throw an exception on failure.

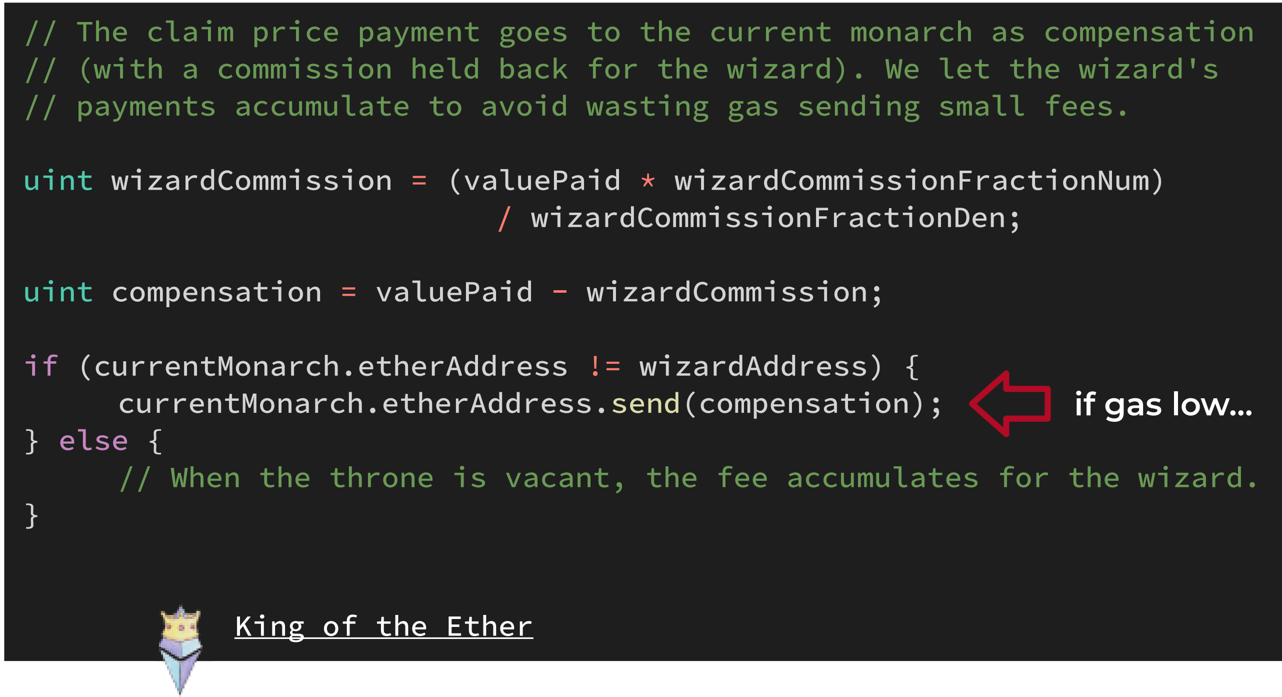
If an external call fails, the contract will continue execution “as if the call succeeded.”

Example 1

Unchecked external call

Certain Solidity operations known as "external calls", require the developer to check that the operation succeeded—in contrast to operations which throw an exception on failure.

If an external call fails, the contract will continue execution "as if the call succeeded."



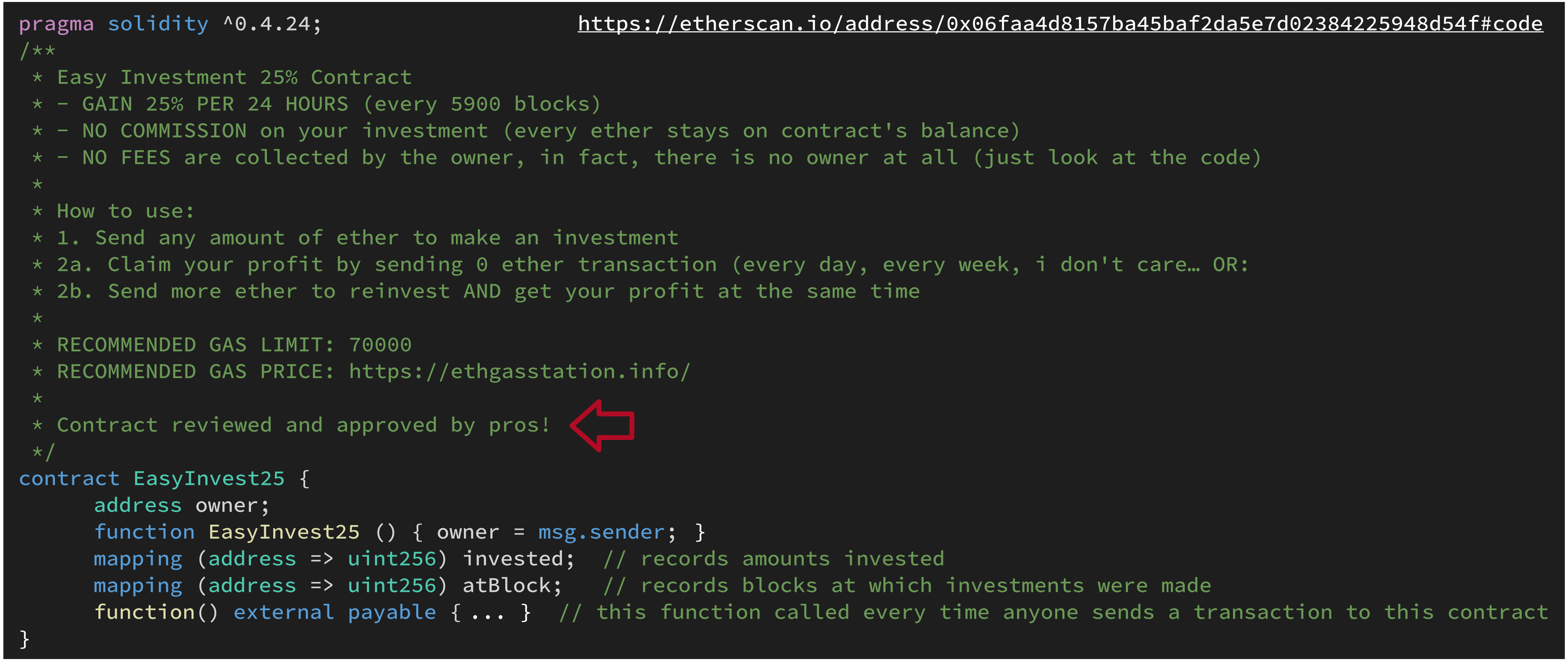
```
// The claim price payment goes to the current monarch as compensation
// (with a commission held back for the wizard). We let the wizard's
// payments accumulate to avoid wasting gas sending small fees.

uint wizardCommission = (valuePaid * wizardCommissionFractionNum)
                           / wizardCommissionFractionDen;

uint compensation = valuePaid - wizardCommission;

if (currentMonarch.etherAddress != wizardAddress) {
     currentMonarch.etherAddress.send(compensation);
} else {
     // When the throne is vacant, the fee accumulates for the wizard.
}
```

if gas low...

King of the Ether

Example 1’

Unchecked external call

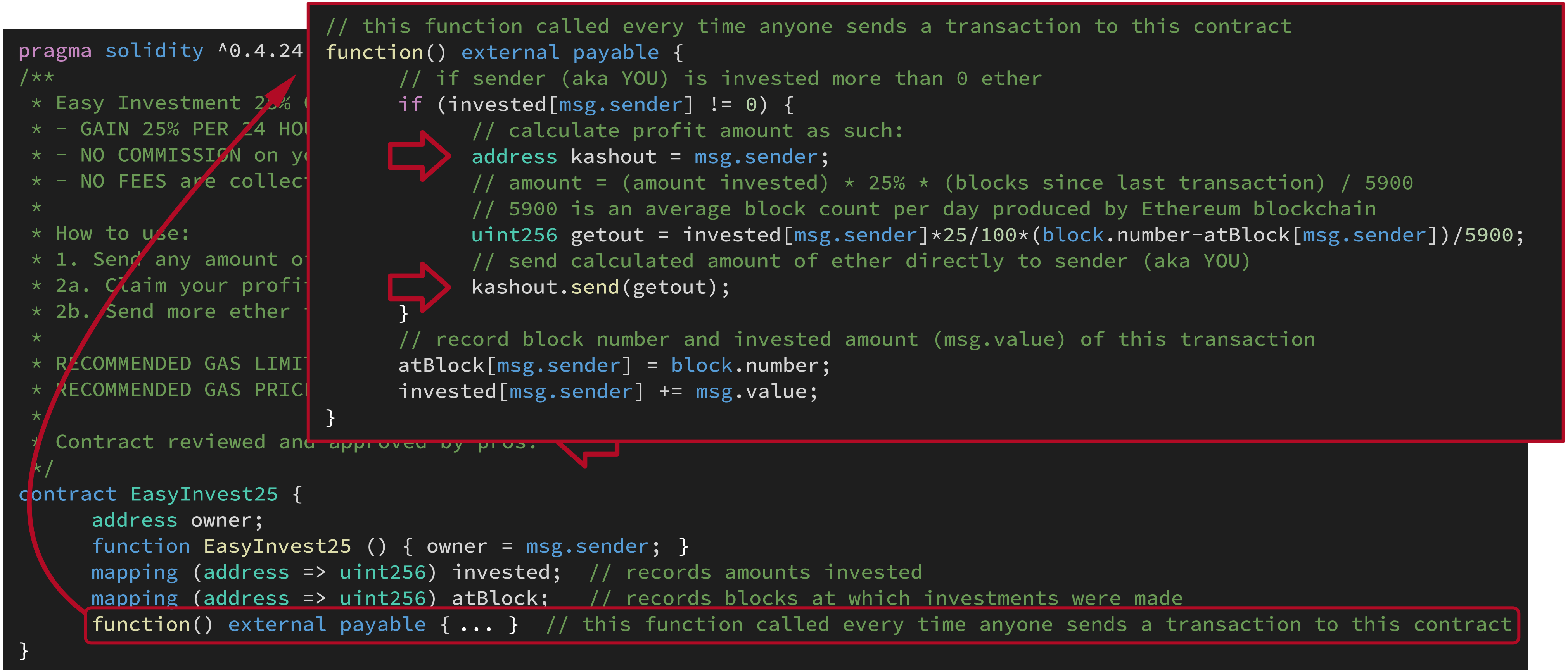


```
pragma solidity ^0.4.24;                    https://etherscan.io/address/0x06faa4d8157ba45baf2da5e7d02384225948d54f#code
/**
 * Easy Investment 25% Contract
 * - GAIN 25% PER 24 HOURS (every 5900 blocks)
 * - NO COMMISSION on your investment (every ether stays on contract's balance)
 * - NO FEES are collected by the owner, in fact, there is no owner at all (just look at the code)
 *
 * How to use:
 * 1. Send any amount of ether to make an investment
 * 2a. Claim your profit by sending 0 ether transaction (every day, every week, i don't care… OR:
 * 2b. Send more ether to reinvest AND get your profit at the same time
 *
 * RECOMMENDED GAS LIMIT: 70000
 * RECOMMENDED GAS PRICE: https://ethgasstation.info/
 *
 * Contract reviewed and approved by pros!
 */
contract EasyInvest25 {
      address owner;
      function EasyInvest25 () { owner = msg.sender; }
      mapping (address => uint256) invested;  // records amounts invested
      mapping (address => uint256) atBlock;   // records blocks at which investments were made
      function() external payable { ... }  // this function called every time anyone sends a transaction to this contract
}
```

Example 1’

Unchecked external call

https://etherscan.io/address/0x06faa4d8157ba45baf2da5e7d02384225948d54f#code



```
pragma solidity ^0.4.24
/**
 * Easy Investment 25% (
 * - GAIN 25% PER 24 HOU
 * - NO COMMISSION on yo
 * - NO FEES are collect
 *
 * How to use:
 * 1. Send any amount of
 * 2a. Claim your profit
 * 2b. Send more ether t
 *
 * RECOMMENDED GAS LIMIT
 * RECOMMENDED GAS PRICE
 *
 * Contract reviewed and approved by pros!
 */
contract EasyInvest25 {
      address owner;
      function EasyInvest25 () { owner = msg.sender; }
      mapping (address => uint256) invested;  // records amounts invested
      mapping (address => uint256) atBlock;   // records blocks at which investments were made
      function() external payable { ... }  // this function called every time anyone sends a transaction to this contract
}
```

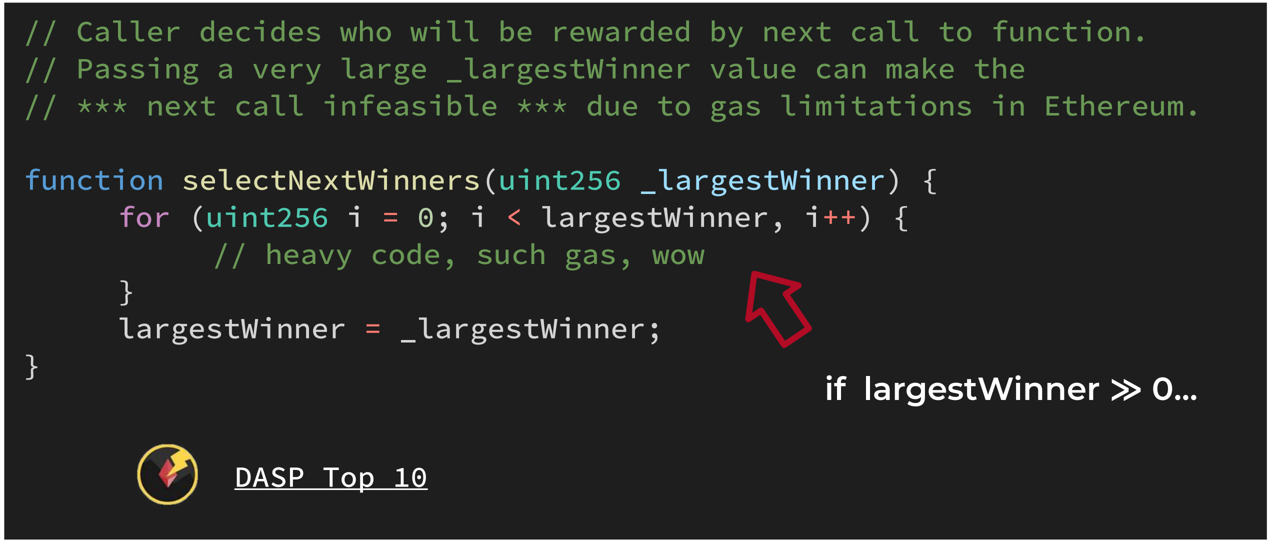
```
// this function called every time anyone sends a transaction to this contract
function() external payable {
      // if sender (aka YOU) is invested more than 0 ether
      if (invested[msg.sender] != 0) {
            // calculate profit amount as such:
            address kashout = msg.sender;
            // amount = (amount invested) * 25% * (blocks since last transaction) / 5900
            // 5900 is an average block count per day produced by Ethereum blockchain
            uint256 getout = invested[msg.sender]*25/100*(block.number-atBlock[msg.sender])/5900;
            // send calculated amount of ether directly to sender (aka YOU)
            kashout.send(getout);
      }
      // record block number and invested amount (msg.value) of this transaction
      atBlock[msg.sender] = block.number;
      invested[msg.sender] += msg.value;
}
```

Example 2

Denial of Service

Denial of service is deadly in the world of Ethereum: while other types of applications can eventually recover,

smart contracts can be taken offline forever by just one of these attacks.



```
// Caller decides who will be rewarded by next call to function.
// Passing a very large _largestWinner value can make the
// *** next call infeasible *** due to gas limitations in Ethereum.

function selectNextWinners(uint256 _largestWinner) {
     for (uint256 i = 0; i < largestWinner, i++) {
          // heavy code, such gas, wow
     }
     largestWinner = _largestWinner;
}
```

if largestWinner ≫ 0...

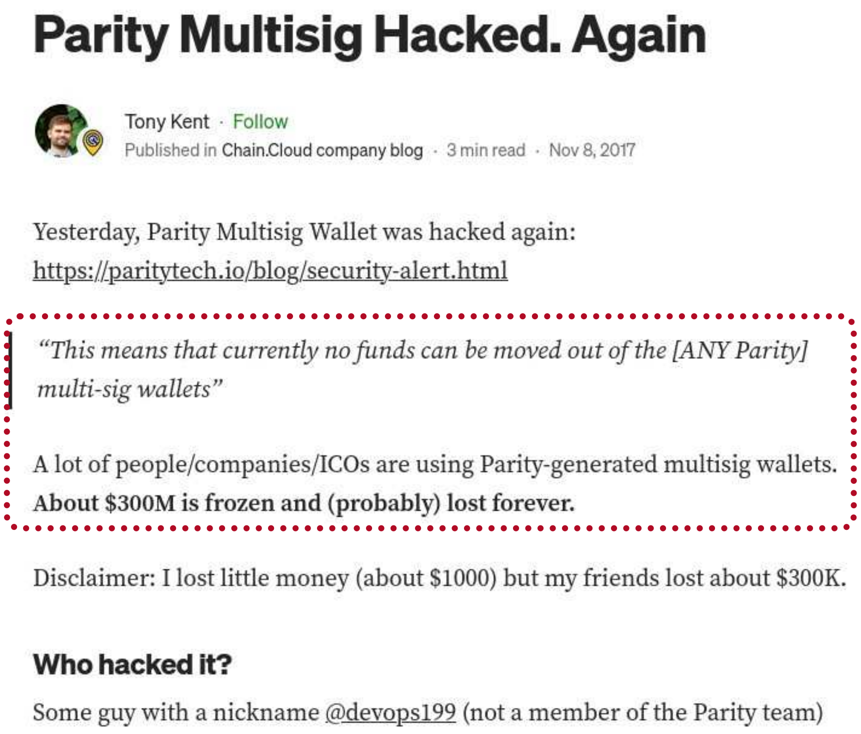
DASP Top 10

Example 2′

Denial of Service

All legit execution (perhaps should be “access control”)

Poor guy even did it accidentally!



Parity Multisig Hacked. Again

Tony Kent · Follow

Published in Chain.Cloud company blog · 3 min read · Nov 8, 2017

Yesterday, Parity Multisig Wallet was hacked again: https://paritytech.io/blog/security-alert.html

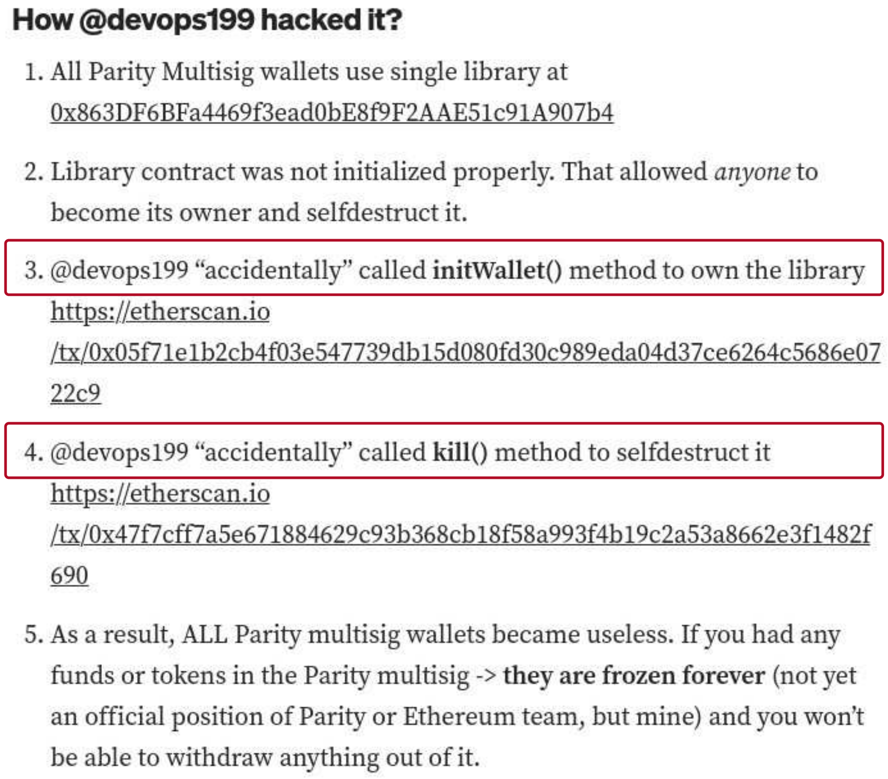
> *“This means that currently no funds can be moved out of the [ANY Parity] multi-sig wallets”*

A lot of people/companies/ICOs are using Parity-generated multisig wallets. **About $300M is frozen and (probably) lost forever.**

Disclaimer: I lost little money (about $1000) but my friends lost about $300K.

Who hacked it?

Some guy with a nickname @devops199 (not a member of the Parity team)



How @devops199 hacked it?

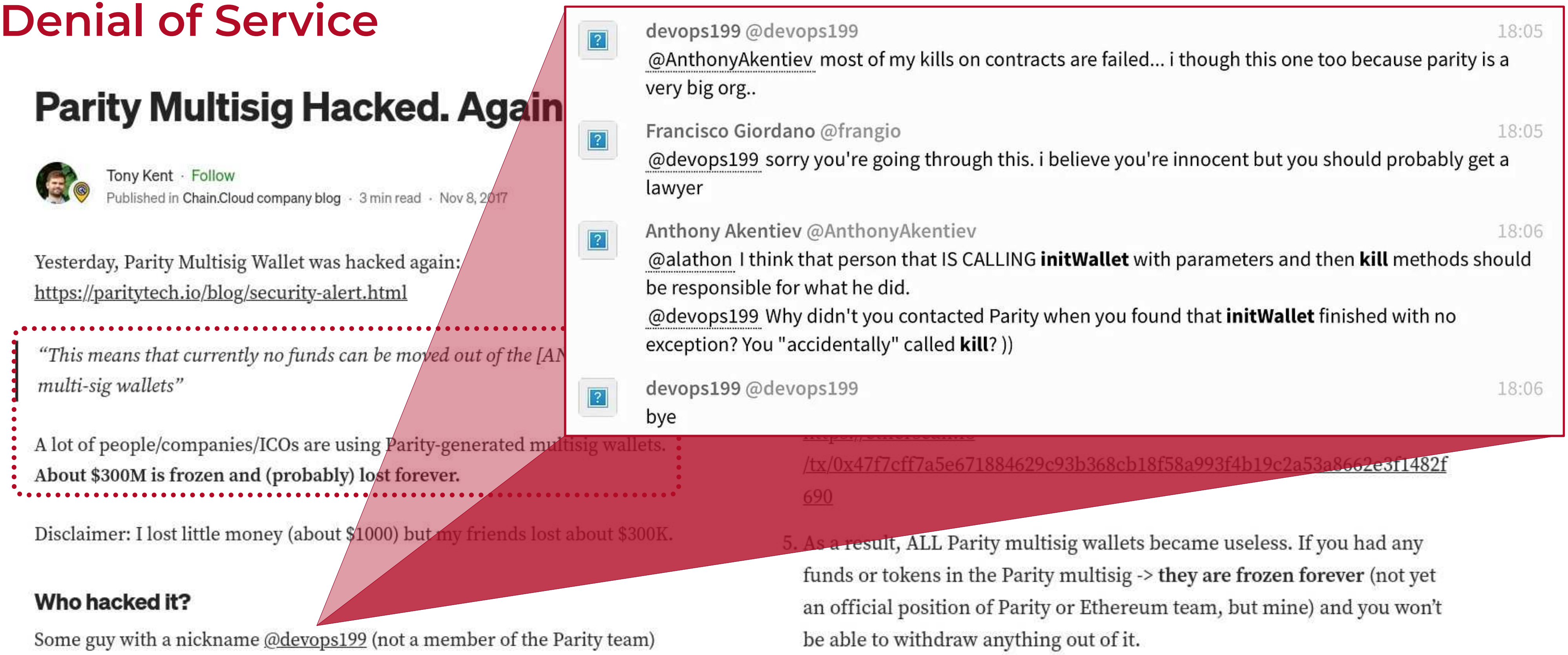
1. All Parity Multisig wallets use single library at 0x863DF6BFa4469f3ead0bE8f9F2AAE51c91A907b4
2. Library contract was not initialized properly. That allowed *anyone* to become its owner and selfdestruct it.
3. @devops199 “accidentally” called **initWallet()** method to own the library https://etherscan.io/tx/0x05f71e1b2cb4f03e547739db15d080fd30c989eda04d37ce6264c5686e0722c9
4. @devops199 “accidentally” called **kill()** method to selfdestruct it https://etherscan.io/tx/0x47f7cff7a5e671884629c93b368cb18f58a993f4b19c2a53a8662e3f1482f690
5. As a result, ALL Parity multisig wallets became useless. If you had any funds or tokens in the Parity multisig -> **they are frozen forever** (not yet an official position of Parity or Ethereum team, but mine) and you won’t be able to withdraw anything out of it.

https://medium.com/chain-cloud-company-blog/parity-multisig-hack-again-b46771eaa838

Example 2′

All legit execution (perhaps should be “access control”)

Denial of Service



Parity Multisig Hacked. Again

Tony Kent · Follow
Published in Chain.Cloud company blog · 3 min read · Nov 8, 2017

Yesterday, Parity Multisig Wallet was hacked again:
https://paritytech.io/blog/security-alert.html

> *“This means that currently no funds can be moved out of the [AN*
> *multi-sig wallets”*

A lot of people/companies/ICOs are using Parity-generated multisig wallets. **About $300M is frozen and (probably) lost forever.**

Disclaimer: I lost little money (about $1000) but my friends lost about $300K.

Who hacked it?

Some guy with a nickname @devops199 (not a member of the Parity team)

/tx/0x47f7cff7a5e671884629c93b368cb18f58a993f4b19c2a53a8662e3f1482f
690

5. As a result, ALL Parity multisig wallets became useless. If you had any funds or tokens in the Parity multisig -> **they are frozen forever** (not yet an official position of Parity or Ethereum team, but mine) and you won’t be able to withdraw anything out of it.

devops199 @devops199 18:05
@AnthonyAkentiev most of my kills on contracts are failed... i though this one too because parity is a very big org..

Francisco Giordano @frangio 18:05
@devops199 sorry you're going through this. i believe you're innocent but you should probably get a lawyer

Anthony Akentiev @AnthonyAkentiev 18:06
@alathon I think that person that IS CALLING **initWallet** with parameters and then **kill** methods should be responsible for what he did.
@devops199 Why didn't you contacted Parity when you found that **initWallet** finished with no exception? You "accidentally" called **kill**?))

devops199 @devops199 18:06
bye

https://medium.com/chain-cloud-company-blog/parity-multisig-hack-again-b46771eaa838

Is all lost?

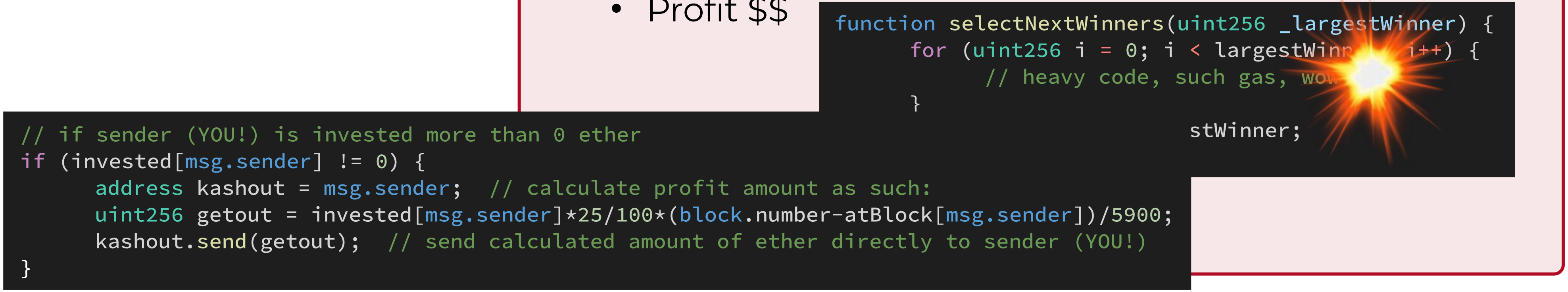
inSecurity in Solidity

State-of-the-art in security for BSC

- Symbolic execution
- Formal methods
- Fuzz testing
- Deep learning (ML)

Static / dynamic code analysis:

- Build Control Flow Graph (CFG)
- Variables inputs as symbolic expressions in CFG
- Symbolic path has condition over those expressions
- Feed full thing to SMT solver, e.g. Z3
- Profit $$



```
function selectNextWinners(uint256 _largestWinner) {
      for (uint256 i = 0; i < largestWinn    i++) {
            // heavy code, such gas, wo
      }
                                          stWinner;
```

```
// if sender (YOU!) is invested more than 0 ether
if (invested[msg.sender] != 0) {
      address kashout = msg.sender;  // calculate profit amount as such:
      uint256 getout = invested[msg.sender]*25/100*(block.number-atBlock[msg.sender])/5900;
      kashout.send(getout);  // send calculated amount of ether directly to sender (YOU!)
}
```

State-of-the-art in security for BSC

- Symbolic execution
- Formal methods
- Fuzz testing
- Deep learning (ML)

Theorem proving & model checking (mainly)

- Create specification/model of desired behaviour
- Create model available implementation
- Prove/check whether current implementation "refines"/"simulates" the specification
- Some flavours:
  - Formal code semantics (denotational, small step, …)
  - Abstract interpretation e.g. via function decorations
  - EVM bytecode to Prolog (!)
  - Etc.

Pretty hard
(if not impossible)
to automate

Correctness mathematically guaranteed!

State-of-the-art in security for BSC

- Symbolic execution
- Formal methods
- Fuzz testing
- Deep learning (ML)

Generate invalid input and monitor execution

inSecurity in Solidity'''

State-of-the-art in security for BSC

- Symbolic execution
- Formal methods
- Fuzz testing
- Deep learning (ML)

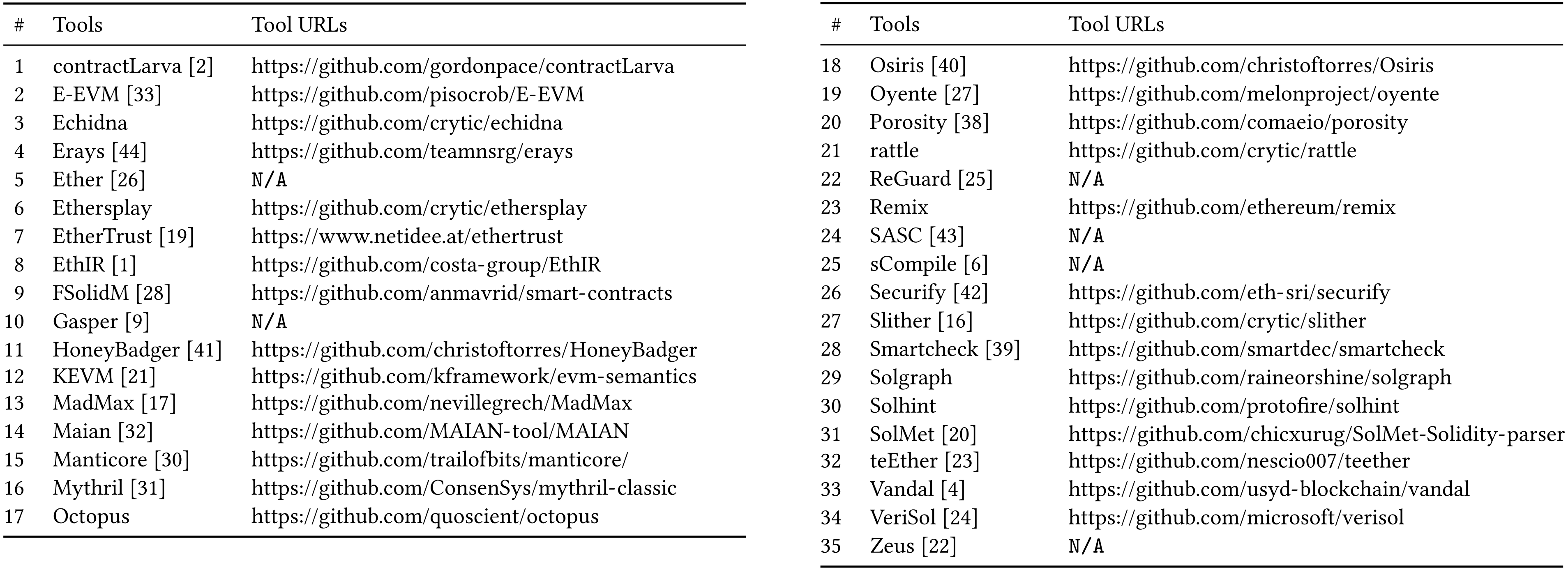
Usual arsenal of black-box methods, now with BSC

- Supervised-learn bytecode of buggy contracts
- Detect fishy patterns in source code
- …

```
Write an exploit for this
contract, that transfers
ETH to address 0x863DF6BFa4…
```

Security-enhancing tools

What about implementing those approaches?



#	Tools	Tool URLs
1	contractLarva [2]	https://github.com/gordonpace/contractLarva
2	E-EVM [33]	https://github.com/pisocrob/E-EVM
3	Echidna	https://github.com/crytic/echidna
4	Erays [44]	https://github.com/teamnsrg/erays
5	Ether [26]	N/A
6	Ethersplay	https://github.com/crytic/ethersplay
7	EtherTrust [19]	https://www.netidee.at/ethertrust
8	EthIR [1]	https://github.com/costa-group/EthIR
9	FSolidM [28]	https://github.com/anmavrid/smart-contracts
10	Gasper [9]	N/A
11	HoneyBadger [41]	https://github.com/christoftorres/HoneyBadger
12	KEVM [21]	https://github.com/kframework/evm-semantics
13	MadMax [17]	https://github.com/nevillegrech/MadMax
14	Maian [32]	https://github.com/MAIAN-tool/MAIAN
15	Manticore [30]	https://github.com/trailofbits/manticore/
16	Mythril [31]	https://github.com/ConsenSys/mythril-classic
17	Octopus	https://github.com/quoscient/octopus
18	Osiris [40]	https://github.com/christoftorres/Osiris
19	Oyente [27]	https://github.com/melonproject/oyente
20	Porosity [38]	https://github.com/comaeio/porosity
21	rattle	https://github.com/crytic/rattle
22	ReGuard [25]	N/A
23	Remix	https://github.com/ethereum/remix
24	SASC [43]	N/A
25	sCompile [6]	N/A
26	Securify [42]	https://github.com/eth-sri/securify
27	Slither [16]	https://github.com/crytic/slither
28	Smartcheck [39]	https://github.com/smartdec/smartcheck
29	Solgraph	https://github.com/raineorshine/solgraph
30	Solhint	https://github.com/protofire/solhint
31	SolMet [20]	https://github.com/chicxurug/SolMet-Solidity-parser
32	teEther [23]	https://github.com/nescio007/teether
33	Vandal [4]	https://github.com/usyd-blockchain/vandal
34	VeriSol [24]	https://github.com/microsoft/verisol
35	Zeus [22]	N/A

* Durieux et al.: "Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts" (ICSE 2020)

Has all been done?

Vulnerabilities in Solidity BSC

Collect (and classify) true-positive vulnerabilities

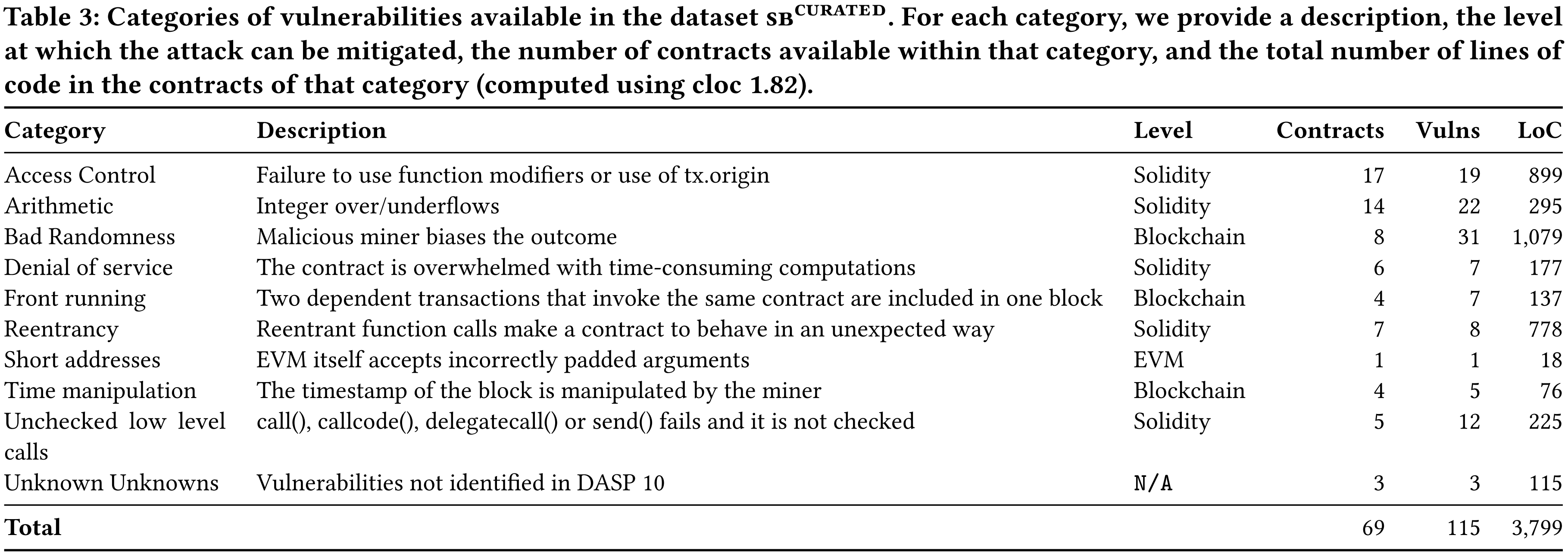


Table 3: Categories of vulnerabilities available in the dataset SB$^{\text{CURATED}}$. For each category, we provide a description, the level at which the attack can be mitigated, the number of contracts available within that category, and the total number of lines of code in the contracts of that category (computed using cloc 1.82).

Category	Description	Level	Contracts	Vulns	LoC
Access Control	Failure to use function modifiers or use of tx.origin	Solidity	17	19	899
Arithmetic	Integer over/underflows	Solidity	14	22	295
Bad Randomness	Malicious miner biases the outcome	Blockchain	8	31	1,079
Denial of service	The contract is overwhelmed with time-consuming computations	Solidity	6	7	177
Front running	Two dependent transactions that invoke the same contract are included in one block	Blockchain	4	7	137
Reentrancy	Reentrant function calls make a contract to behave in an unexpected way	Solidity	7	8	778
Short addresses	EVM itself accepts incorrectly padded arguments	EVM	1	1	18
Time manipulation	The timestamp of the block is manipulated by the miner	Blockchain	4	5	76
Unchecked low level calls	call(), callcode(), delegatecall() or send() fails and it is not checked	Solidity	5	12	225
Unknown Unknowns	Vulnerabilities not identified in DASP 10	N/A	3	3	115
Total			69	115	3,799

* Durieux et al.: "Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts" (ICSE 2020)

Vulnerabilities in Solidity BSC

Collect (and classify) true-positive vulnerabilities

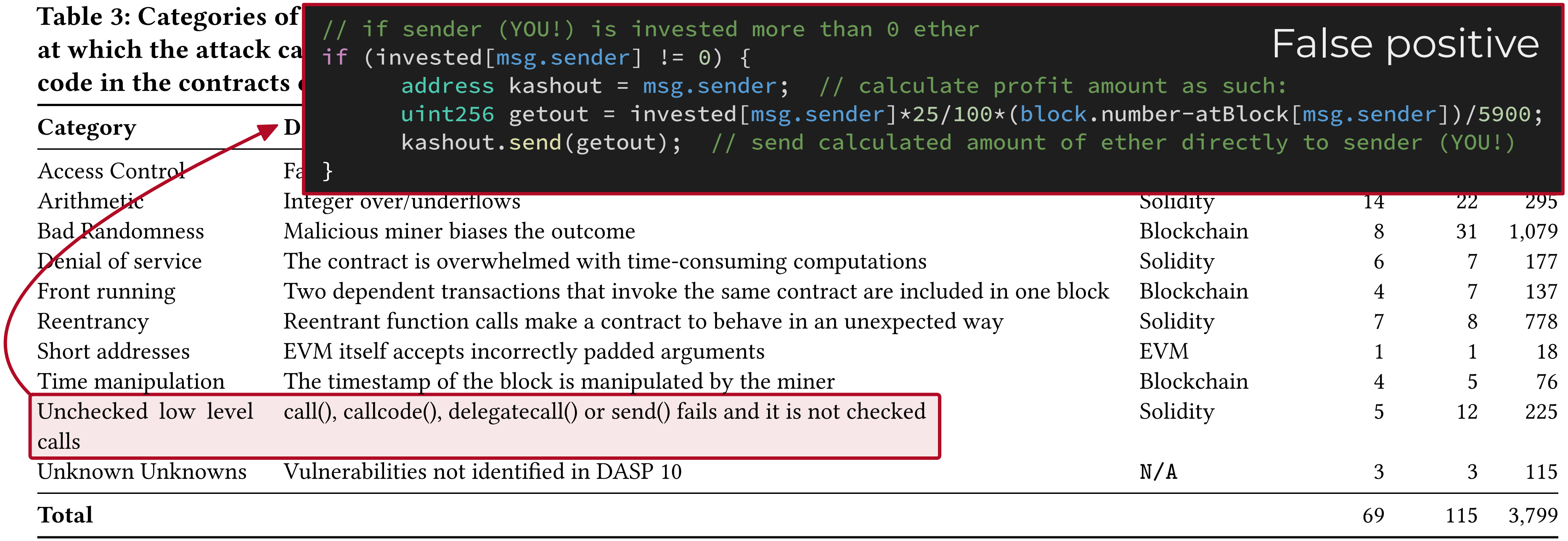


Table 3: Categories of
at which the attack ca
code in the contracts

```
// if sender (YOU!) is invested more than 0 ether
if (invested[msg.sender] != 0) {
     address kashout = msg.sender;  // calculate profit amount as such:
     uint256 getout = invested[msg.sender]*25/100*(block.number-atBlock[msg.sender])/5900;
     kashout.send(getout);  // send calculated amount of ether directly to sender (YOU!)
}
```

False positive

Category	D				
Access Control	Fa				
Arithmetic	Integer over/underflows	Solidity	14	22	295
Bad Randomness	Malicious miner biases the outcome	Blockchain	8	31	1,079
Denial of service	The contract is overwhelmed with time-consuming computations	Solidity	6	7	177
Front running	Two dependent transactions that invoke the same contract are included in one block	Blockchain	4	7	137
Reentrancy	Reentrant function calls make a contract to behave in an unexpected way	Solidity	7	8	778
Short addresses	EVM itself accepts incorrectly padded arguments	EVM	1	1	18
Time manipulation	The timestamp of the block is manipulated by the miner	Blockchain	4	5	76
Unchecked low level calls	call(), callcode(), delegatecall() or send() fails and it is not checked	Solidity	5	12	225
Unknown Unknowns	Vulnerabilities not identified in DASP 10	N/A	3	3	115
Total			69	115	3,799

* Durieux et al.: "Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts" (ICSE 2020)

Vulnerabilities in Solidity BSC’

Collect (and classify) “true-positive” vulnerabilities

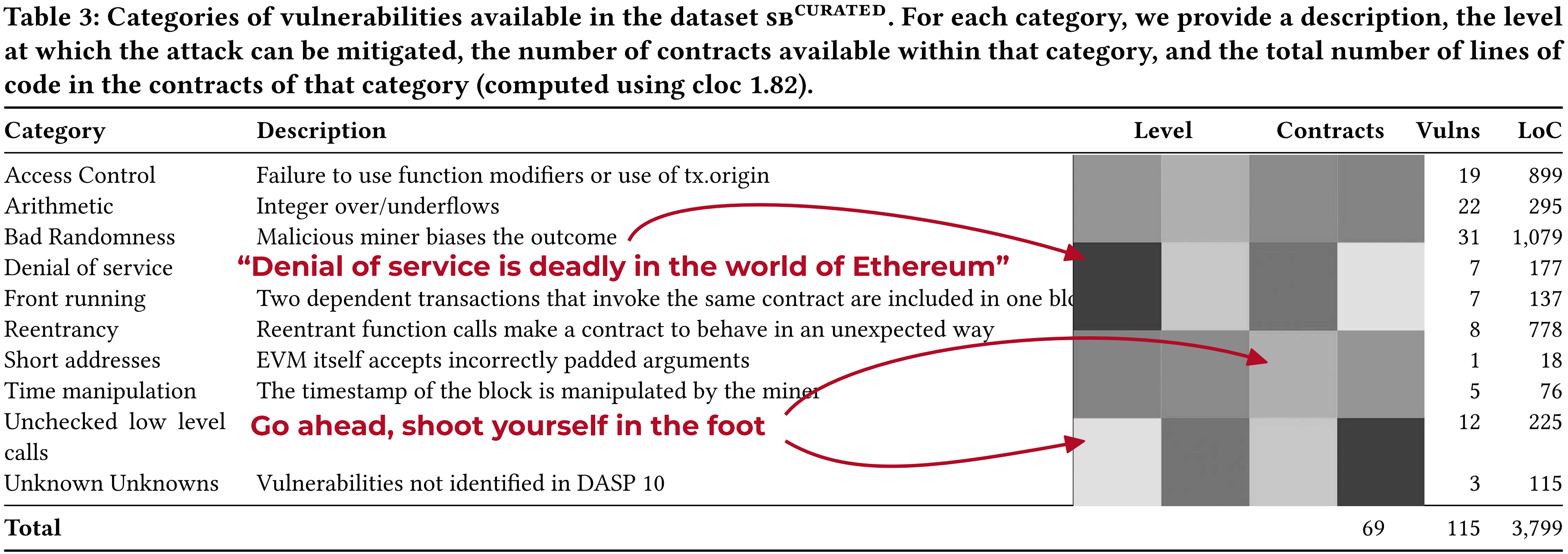


Table 3: Categories of vulnerabilities available in the dataset SB$^{\text{CURATED}}$. For each category, we provide a description, the level at which the attack can be mitigated, the number of contracts available within that category, and the total number of lines of code in the contracts of that category (computed using cloc 1.82).

Category	Description	Level	Contracts	Vulns	LoC
Access Control	Failure to use function modifiers or use of tx.origin	[redacted]	[redacted]	19	899
Arithmetic	Integer over/underflows	[redacted]	[redacted]	22	295
Bad Randomness	Malicious miner biases the outcome	[redacted]	[redacted]	31	1,079
Denial of service	**“Denial of service is deadly in the world of Ethereum”**	[redacted]	[redacted]	7	177
Front running	Two dependent transactions that invoke the same contract are included in one bl	[redacted]	[redacted]	7	137
Reentrancy	Reentrant function calls make a contract to behave in an unexpected way	[redacted]	[redacted]	8	778
Short addresses	EVM itself accepts incorrectly padded arguments	[redacted]	[redacted]	1	18
Time manipulation	The timestamp of the block is manipulated by the miner	[redacted]	[redacted]	5	76
Unchecked low level calls	**Go ahead, shoot yourself in the foot**	[redacted]	[redacted]	12	225
Unknown Unknowns	Vulnerabilities not identified in DASP 10	[redacted]	[redacted]	3	115
Total			69	115	3,799

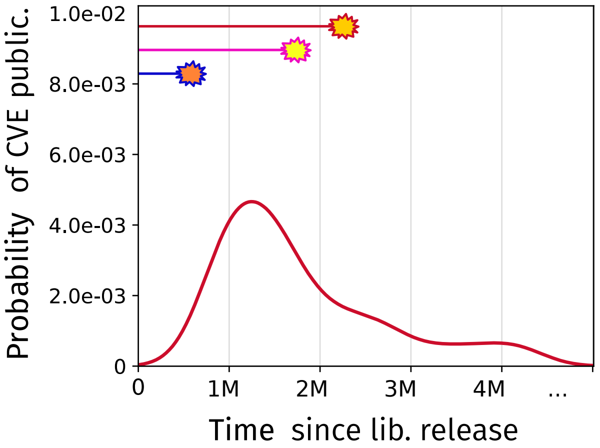
* Durieux et al.: "Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts" (ICSE 2020)

Vulnerabilities in Solidity BSC”

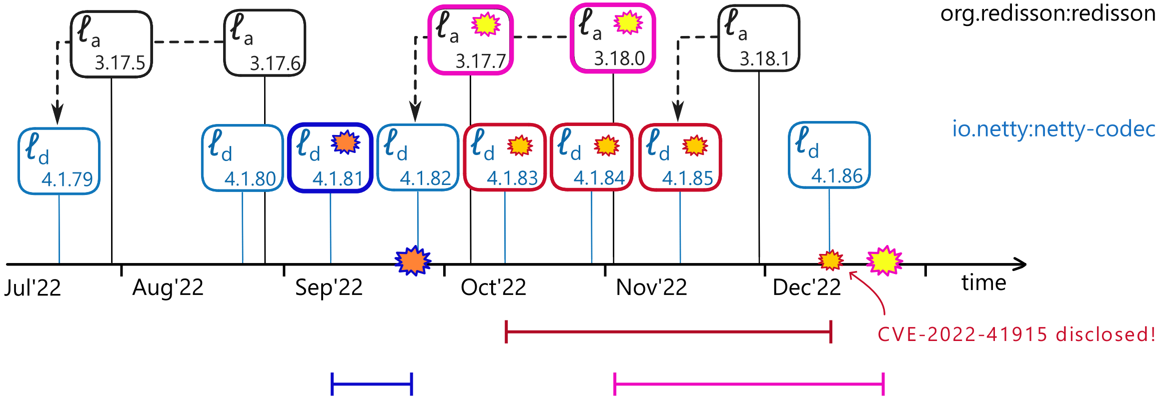
Vulnerability introduction-discovery correlations

ProSVED

Projection of Security Vulnerabilities caused by Exploits in Dependencies

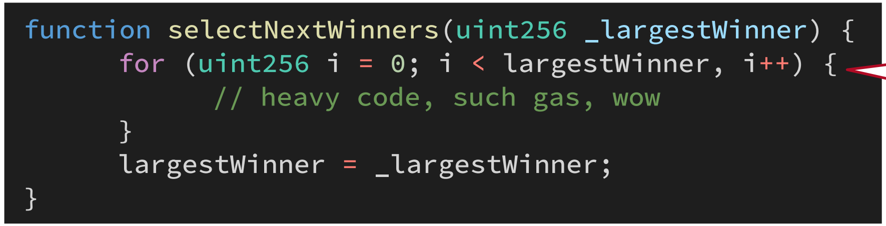


Relies on quantitative data like CVSS of CVEs



Security in BSC

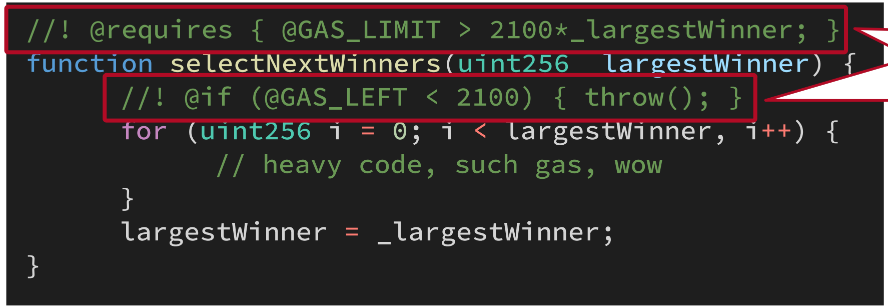
Do smart contracts *really* need Turing completeness?



```
function selectNextWinners(uint256 _largestWinner) {
      for (uint256 i = 0; i < largestWinner, i++) {
            // heavy code, such gas, wow
      }
      largestWinner = _largestWinner;
}
```

```
do {
      break(havoc);
} while (still_works);
```

But now they have already tasted blood...



```
//! @requires { @GAS_LIMIT > 2100*_largestWinner; }
function selectNextWinners(uint256 _largestWinner) {
      //! @if (@GAS_LEFT < 2100) { throw(); }
      for (uint256 i = 0; i < largestWinner, i++) {
            // heavy code, such gas, wow
      }
      largestWinner = _largestWinner;
}
```

Code annotations, some could be automated from meta-parameters

UNIVERSITÀ
DI TRENTO

ProSVED
Projection of Security Vulnerabilities
caused by Exploits in Dependencies

Formal and smart contracts—or maybe not

Not a presentation of results but of ~~terrible problems~~ opportunities

Carlos E. Budde

Security group @ DISI

carlosesteban.budde@unitn.it

Smartitude

Por suerte,Mendieta,el gaucho tiene algo que nunca caerá bajo la piqueta del progreso: ¡La siesta!
¡Vamos a lo nuestro!