THE UNIVERSITY OF UTAH

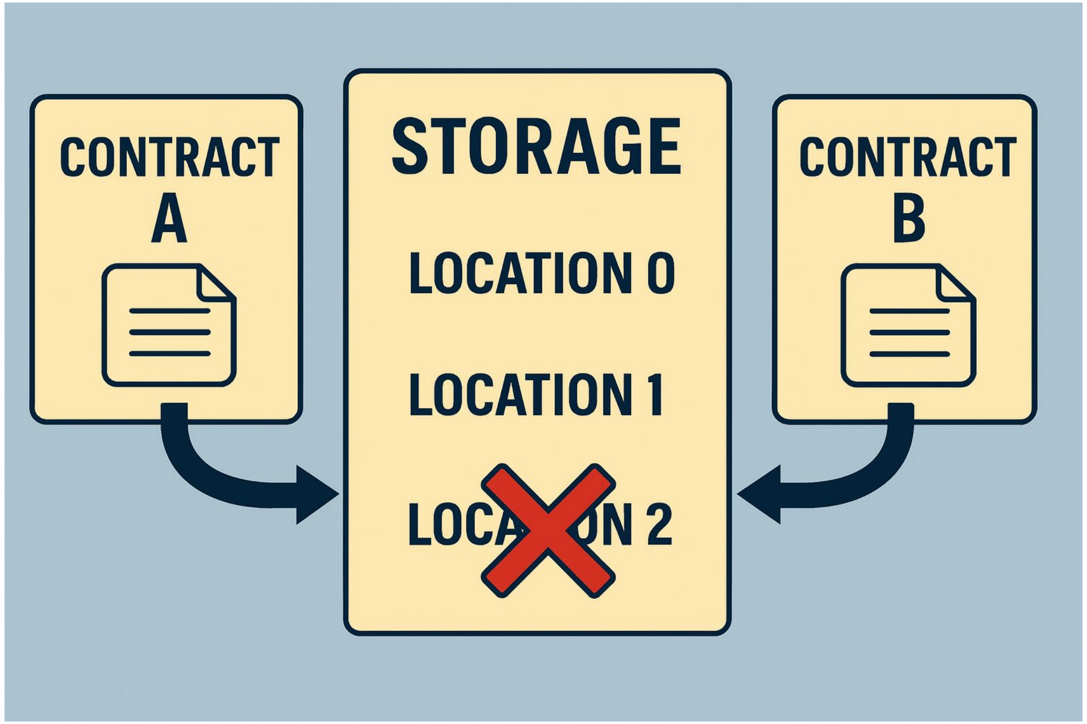
CollisionRepair: First-Aid and Automated Patching for Storage Collision Vulnerabilities in Smart Contracts

Yu Pan[1]*, Wanjing Han[1]*, Yue Duan[2], Mu Zhang[1]

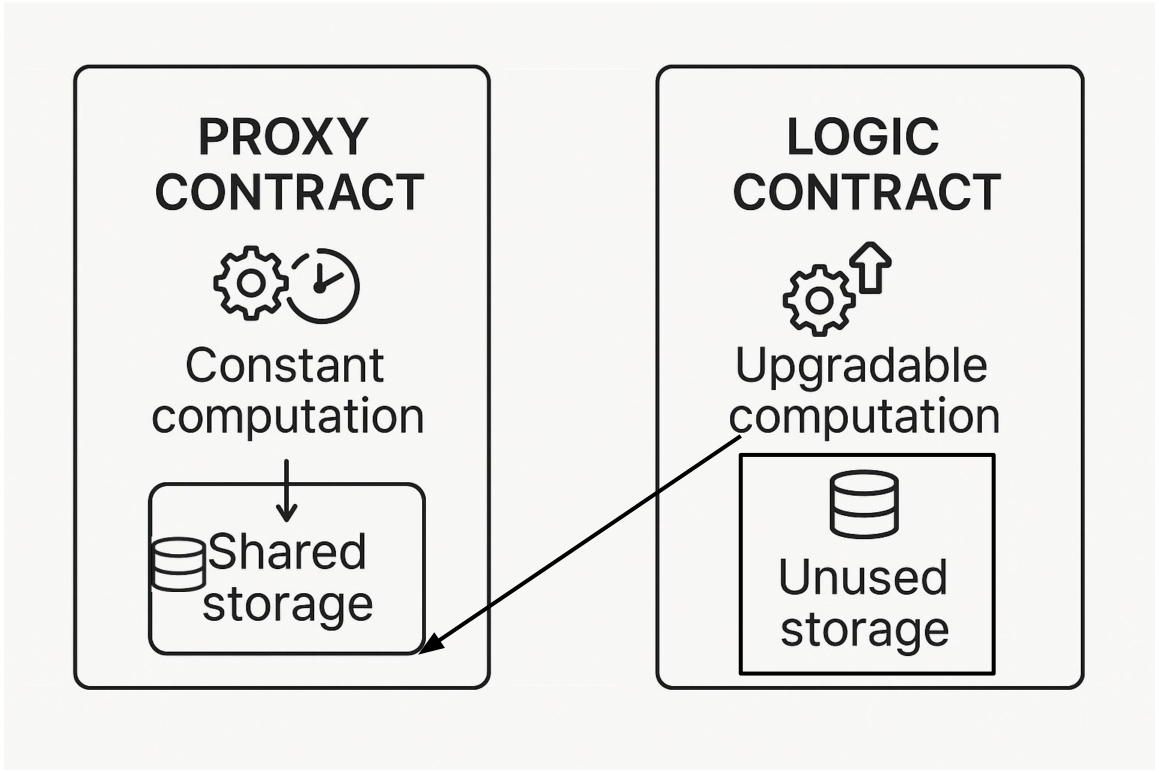
[1]University of Utah, Salt Lake City, United States
[2]Singapore Management University, Singapore

THE UNIVERSITY OF UTAH

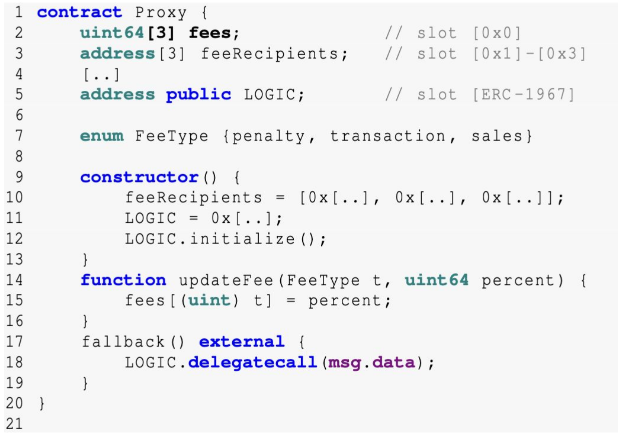
Storage Collision



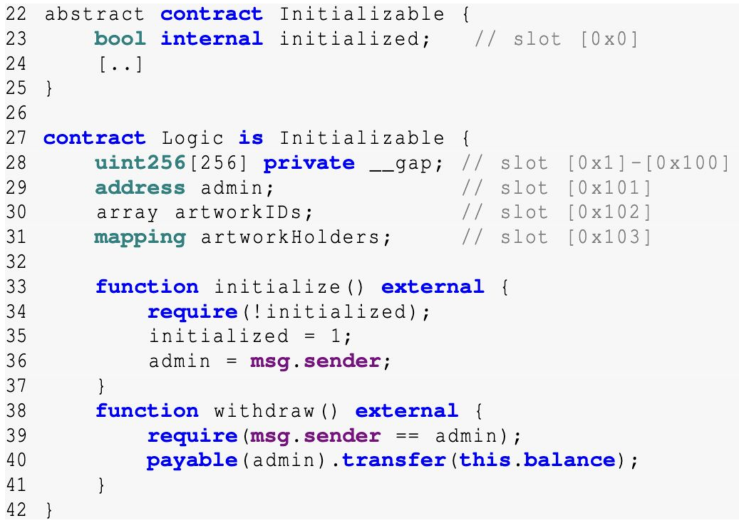
Upgradable Smart Contracts



Motivating Example



```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;
    }
    fallback() external {
        LOGIC.delegatecall(msg.data);
    }
}

```



```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Non-Fungible Token(NFT) Contract

Motivating Example

```
contract Proxy {
    uint64[3] fees;             // slot [0x0]
    address[3] feeRecipients;   // slot [0x1]-[0x3]
    [..]
    address public LOGIC;       // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;
    }
    fallback() external {
        LOGIC.delegatecall(msg.data);
    }
}

```

```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Stores fees and recipient info

Motivating Example

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;
    }
    fallback() external {
        LOGIC.delegatecall(msg.data);
    }
}

```

Stores fees and recipient info

```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Processes withdrawals

State Lives in Proxy Storage

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
```

```
uint64[3] fees;
```

```
    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;
    }
    fallback() external {
        LOGIC.delegatecall(msg.data);
    }
}
```

THE UNIVERSITY OF UTAH

State Lives in Proxy Storage

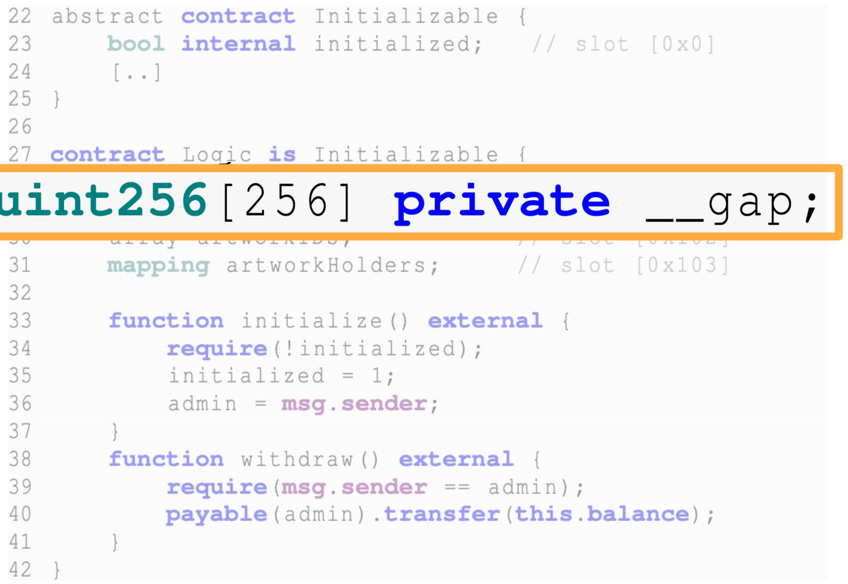
```
contract Proxy {
    uint64[3] fees;                    // slot [0x0]
```

```
uint64[3] fees;
```

```
    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;
    }
    fallback() external {
        LOGIC.delegatecall(msg.data);
    }
}
```

`fees` is Stored in Slot 0

Reserved Storage Slots

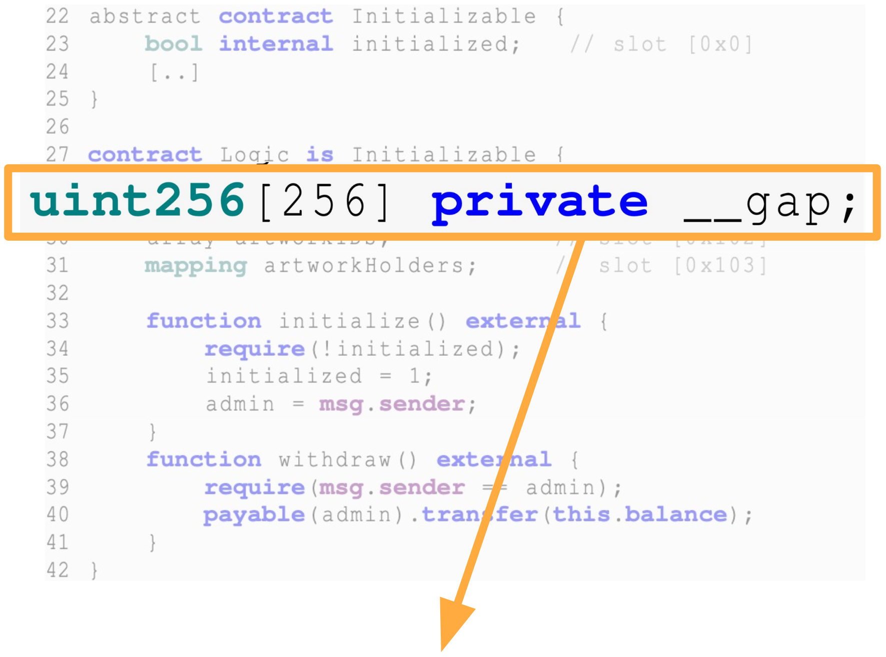


```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
```

```
uint256[256] private __gap;
```

```
    mapping artworkHolders;      // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

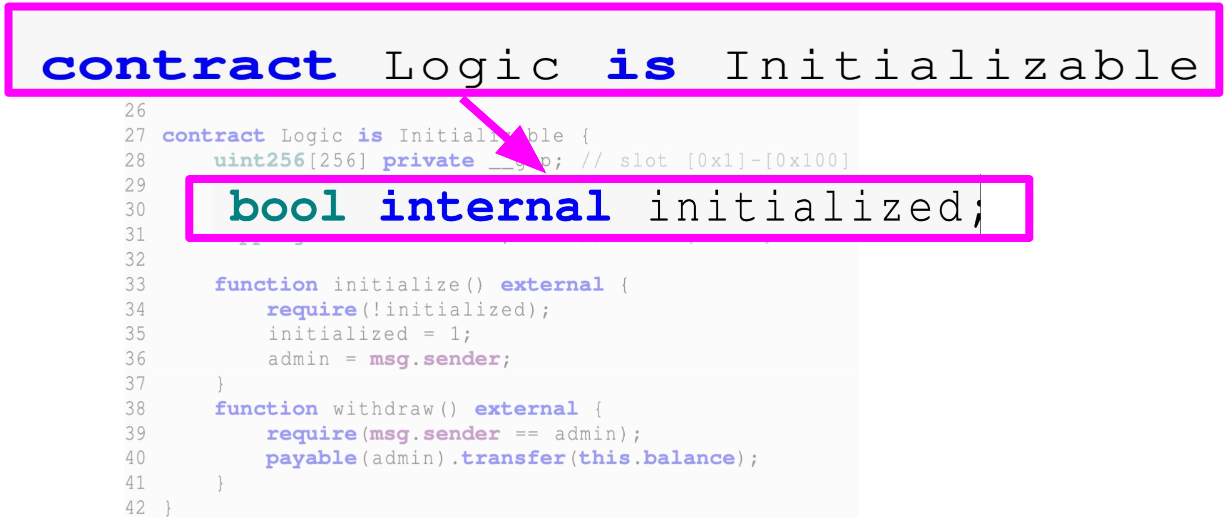
Reserved Storage Slots



```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
uint256[256] private __gap;
    array artworkIDs;             // slot [0x102]
    mapping artworkHolders;       // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Developers' Protection

Avoiding Storage Collision is Hard

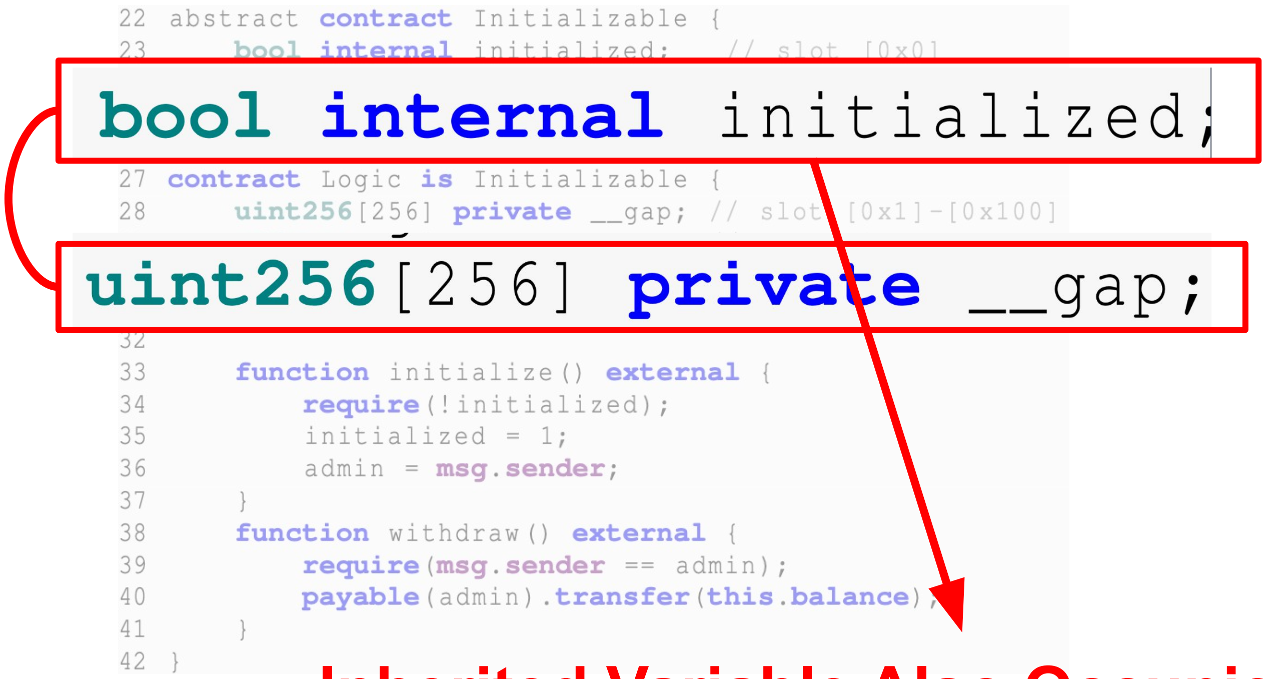


```
contract Logic is Initializable
```

```
bool internal initialized;
```

```
contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Avoiding Storage Collision is Hard



```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
```

```
bool internal initialized;
```

```
contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
```

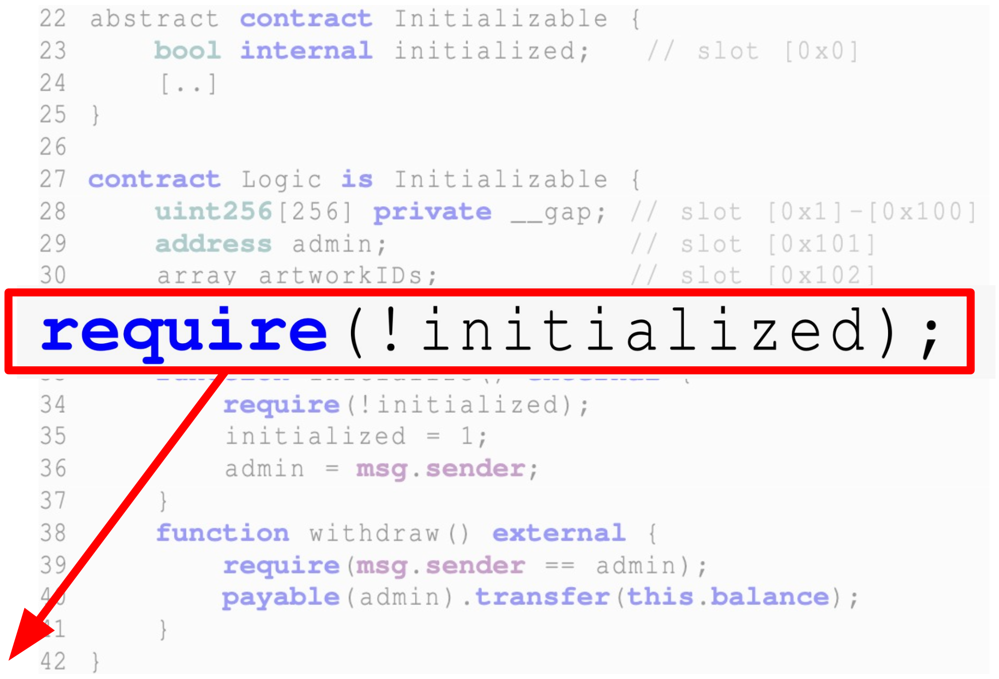
```
uint256[256] private __gap;
```

```
    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Inherited Variable Also Occupies Slot 0!

THE UNIVERSITY OF UTAH

Avoiding Storage Collision is Hard



```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
```

```
require(!initialized);
```

```
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Access Control Flag!

Avoiding Storage Collision is Hard

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
```

```
fees[(uint) t] = percent;
```

```
        LOGIC.delegatecall(msg.data);
    }
}
```

```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Avoiding Storage Collision is Hard

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;

        LOGIC.delegatecall(msg.data);
    }
}
```

may collide

```
fees[(uint) t] = percent;
```

```
bool internal initialized;
```

```
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Avoiding Storage Collision is Hard

```
contract Proxy {
    uint64[3] fees;               // slot [0x0]
    address[3] feeRecipients;     // slot [0x1]-[0x3]
    [..]
    address public LOGIC;         // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
        fees[(uint) t] = percent;

        LOGIC.delegatecall(msg.data);
    }
}
```

```
fees[(uint) t] = percent;
```

may collide

```
bool internal initialized;
```

```
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

Fail to protect; Collision happens!

Static Detection Falls Short

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
```

```
fees[(uint) t] = percent;
```

```
        LOGIC.delegatecall(msg.data);
    }
}
```

Index Resolved at Runtime

```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

THE UNIVERSITY OF UTAH

Static Detection Falls Short

```
contract Proxy {
    uint64[3] fees;              // slot [0x0]
    address[3] feeRecipients;    // slot [0x1]-[0x3]
    [..]
    address public LOGIC;        // slot [ERC-1967]

    enum FeeType {penalty, transaction, sales}

    constructor() {
        feeRecipients = [0x[..], 0x[..], 0x[..]];
        LOGIC = 0x[..];
        LOGIC.initialize();
    }
    function updateFee(FeeType t, uint64 percent) {
```

```
fees[(uint) t] = percent;
```

```
        LOGIC.delegatecall(msg.data);
    }
}
```

Index Resolved at Runtime

```
abstract contract Initializable {
    bool internal initialized;    // slot [0x0]
    [..]
}

contract Logic is Initializable {
    uint256[256] private __gap; // slot [0x1]-[0x100]
    address admin;              // slot [0x101]
    array artworkIDs;           // slot [0x102]
    mapping artworkHolders;     // slot [0x103]

    function initialize() external {
        require(!initialized);
        initialized = 1;
        admin = msg.sender;
    }
    function withdraw() external {
        require(msg.sender == admin);
        payable(admin).transfer(this.balance);
    }
}
```

USCHUNT(***USENIX Security '23***)
CRUSH(***NDSS '24***)

Our Solution

CollisionRepair patches code to precisely mitigate storage collision attacks at runtime

Existing Patching Can’t Solve this Problem!

Existing Patching Can’t Solve this Problem!

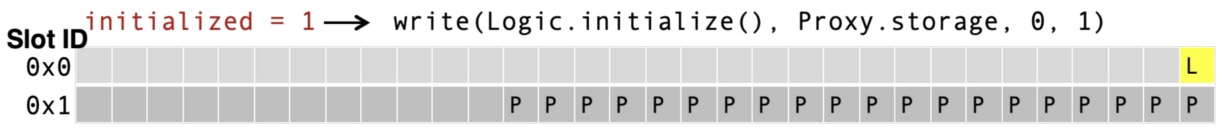
- SmartShield(***SANER '20***)
- SGuard(***Oakland '21***)
- EVMPATCH(***USENIX Security '21***)
- ELYSIUM(***RAID '22***)
- SmartFix(***FSE '23***)

Existing Patching Can’t Solve this Problem!

- SmartShield(***SANER '20***)
- SGuard(***Oakland '21***)
- EVMPATCH(***USENIX Security '21***)
- ELYSIUM(***RAID '22***)
- SmartFix(***FSE '23***)

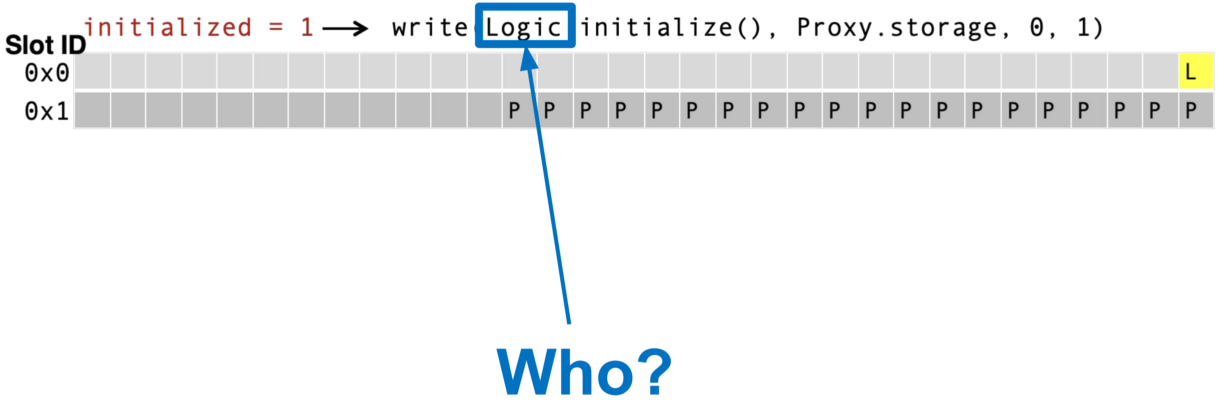
❌ **Single Point Detection**
Cannot Maintain Storage Access States Continuously

Continuous and Precise Monitoring – Solution: Ownership Changes Example

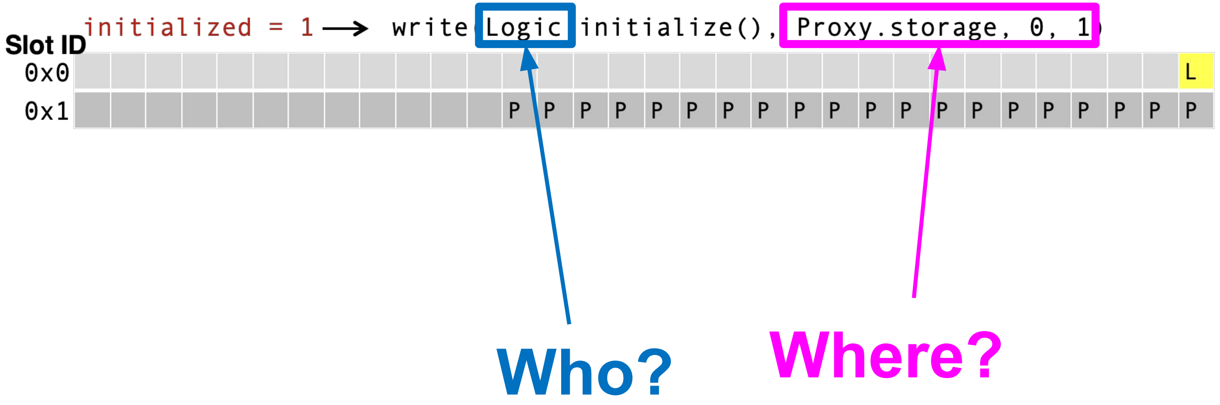


initialized = 1 ⟶ write(Logic.initialize(), Proxy.storage, 0, 1)
Slot ID
0x0
L
0x1
P P P P P P P P P P P P P P P P P P P P

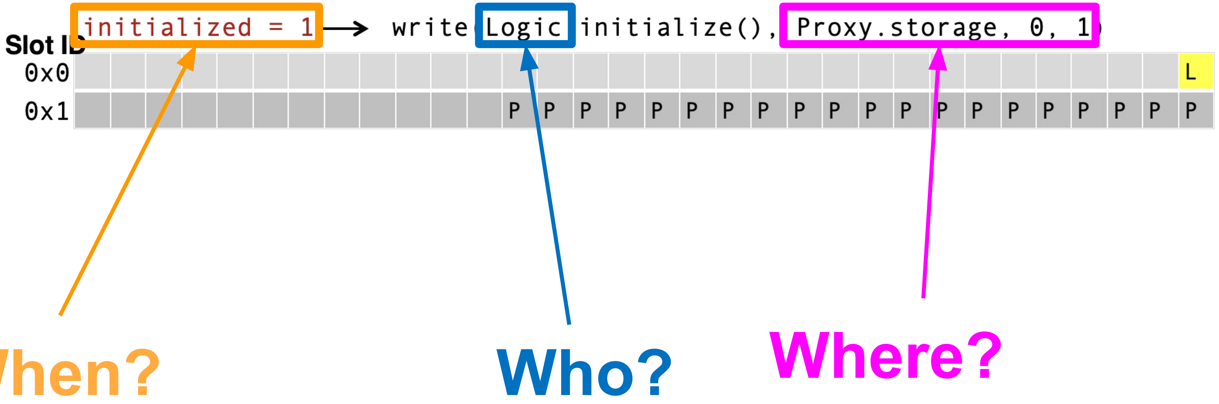
Continuous and Precise Monitoring – Solution: Ownership Changes Example


initialized = 1
write(Logic.initialize(), Proxy.storage, 0, 1)
Slot ID
0x0
0x1
L
P P P P P P P P P P P P P P P P P P P P
Who?

Continuous and Precise Monitoring – Solution: Ownership Changes Example

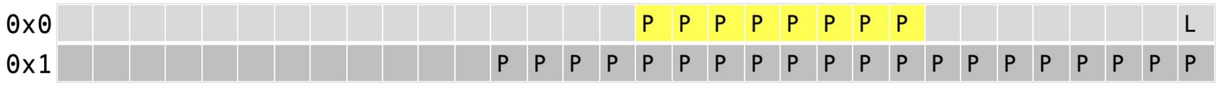

initialized = 1 ⟶ write(Logic.initialize(), Proxy.storage, 0, 1)
Slot ID
0x0
0x1
L
P P P P P P P P P P P P P P P P P P P P
Who?
Where?

Continuous and Precise Monitoring – Solution: Ownership Changes Example



Ownership Changes Example

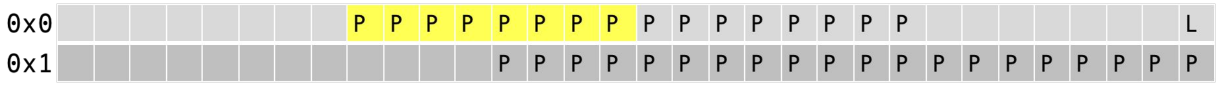
```
fees[1] = 10  ⟶  write(Proxy.updateFee(), Proxy.storage, 8, 8)
```



No collision

Ownership Changes Example

```
fees[2] = 5   ⟶   write(Proxy.updateFee(), Proxy.storage, 16, 8)
```



0x0									P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P								L
0x1													P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P

No collision

Ownership Changes Example

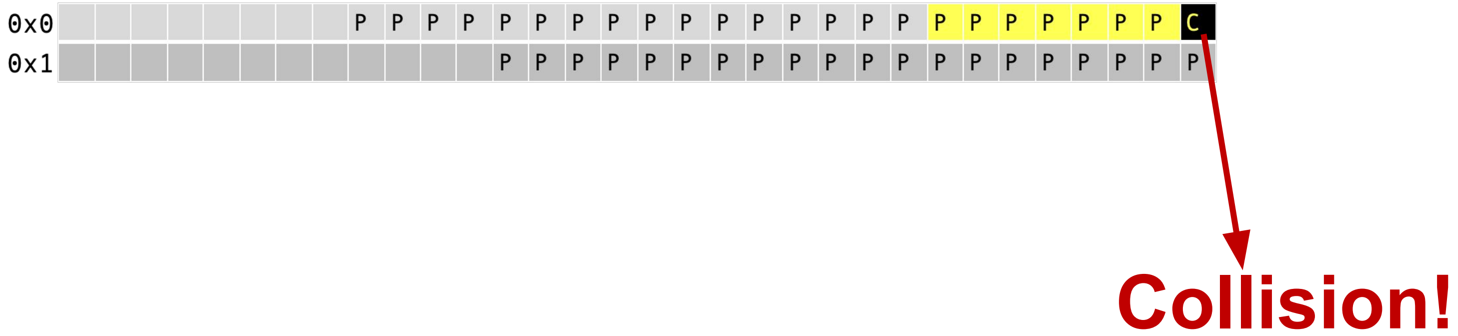
```
fees[1] = 5   ⟶   write(Proxy.updateFee(), Proxy.storage, 8, 8)
```

0x0									P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P								L
0x1													P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P

No collision

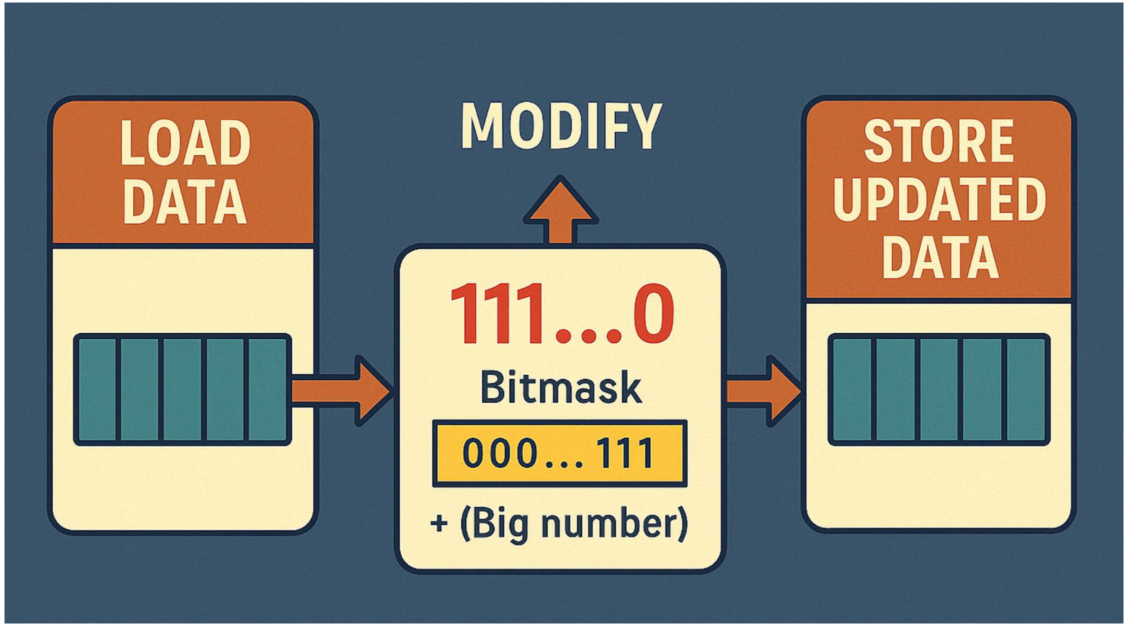
Ownership Changes Example

```
fees[0] = 0   ⟶   write(Proxy.updateFee(), Proxy.storage, 0, 8)
```

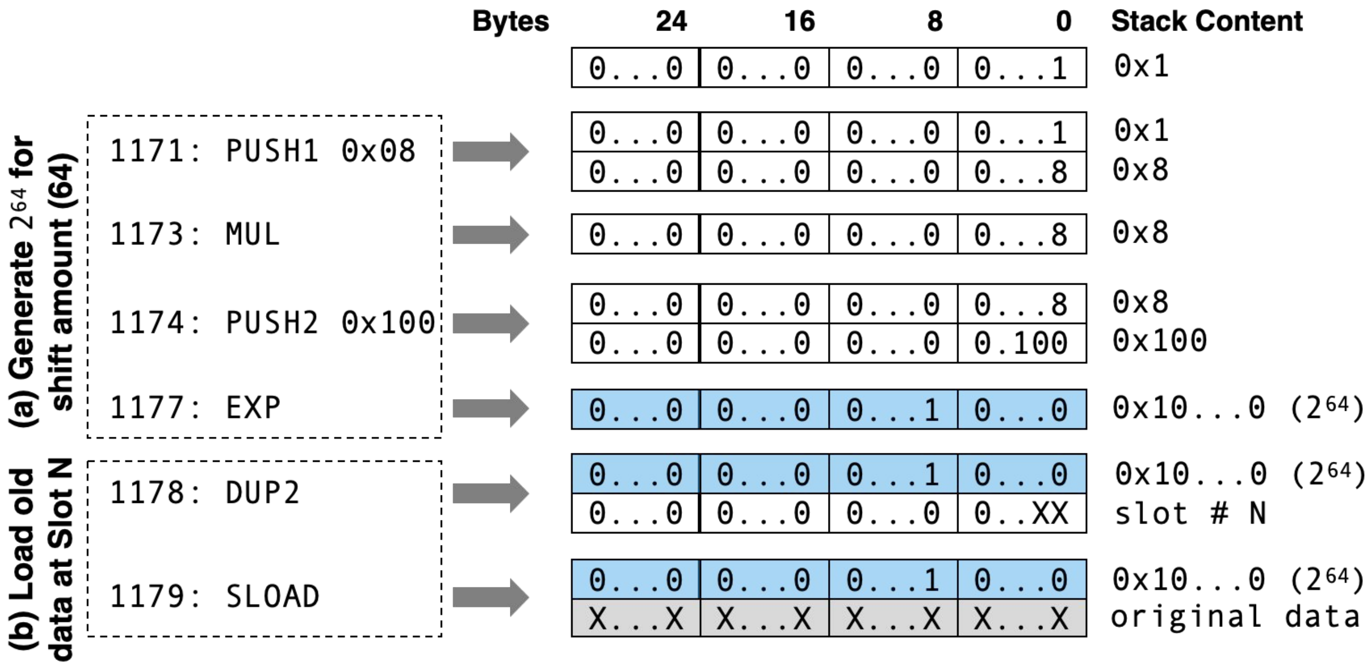


0x0
P P P P P P P P P P P P P P P P P P P P P P P C
0x1
P P P P P P P P P P P P P P P P P P P P
Collision!

Low-level Implementation Insight: Mask & Shift Pattern



Load Data

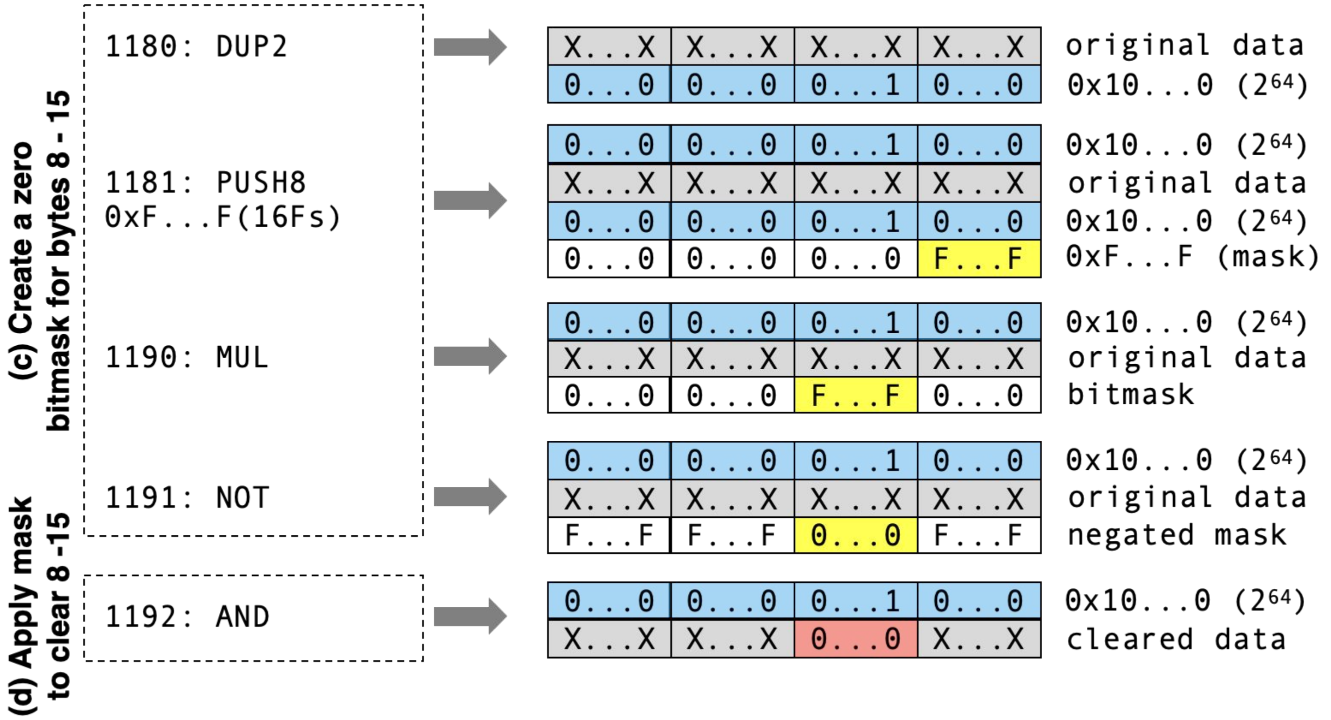


Bytes
24
16
8
0
Stack Content
(a) Generate 2^{64} for shift amount (64)
(b) Load old data at Slot N

```
1171:  PUSH1  0x08
1173:  MUL
1174:  PUSH2  0x100
1177:  EXP
1178:  DUP2
1179:  SLOAD
```

	24	16	8	0	Stack Content
	0...0	0...0	0...0	0...1	0x1
1171: PUSH1 0x08	0...0	0...0	0...0	0...1	0x1
	0...0	0...0	0...0	0...8	0x8
1173: MUL	0...0	0...0	0...0	0...8	0x8
1174: PUSH2 0x100	0...0	0...0	0...0	0...8	0x8
	0...0	0...0	0...0	0.100	0x100
1177: EXP	0...0	0...0	0...1	0...0	0x10...0 (2^{64})
1178: DUP2	0...0	0...0	0...1	0...0	0x10...0 (2^{64})
	0...0	0...0	0...0	0..XX	slot # N
1179: SLOAD	0...0	0...0	0...1	0...0	0x10...0 (2^{64})
	X...X	X...X	X...X	X...X	original data

Create and Apply Bitmask



(c) Create a zero bitmask for bytes 8 - 15

(d) Apply mask to clear 8 -15

```
1180: DUP2

1181: PUSH8
0xF...F(16Fs)

1190: MUL

1191: NOT

1192: AND
```

X...X	X...X	X...X	X...X	original data
0...0	0...0	0...1	0...0	0x10...0 (2^{64})

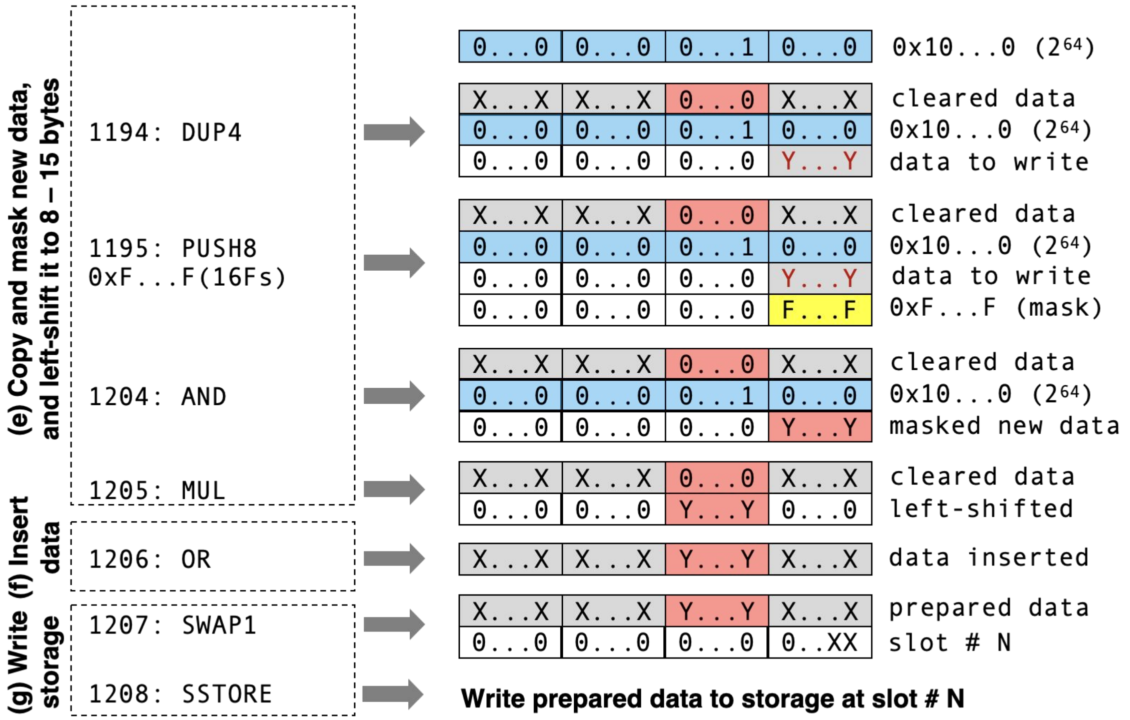
0...0	0...0	0...1	0...0	0x10...0 (2^{64})
X...X	X...X	X...X	X...X	original data
0...0	0...0	0...1	0...0	0x10...0 (2^{64})
0...0	0...0	0...0	F...F	0xF...F (mask)

0...0	0...0	0...1	0...0	0x10...0 (2^{64})
X...X	X...X	X...X	X...X	original data
0...0	0...0	F...F	0...0	bitmask

0...0	0...0	0...1	0...0	0x10...0 (2^{64})
X...X	X...X	X...X	X...X	original data
F...F	F...F	0...0	F...F	negated mask

0...0	0...0	0...1	0...0	0x10...0 (2^{64})
X...X	X...X	0...0	X...X	cleared data

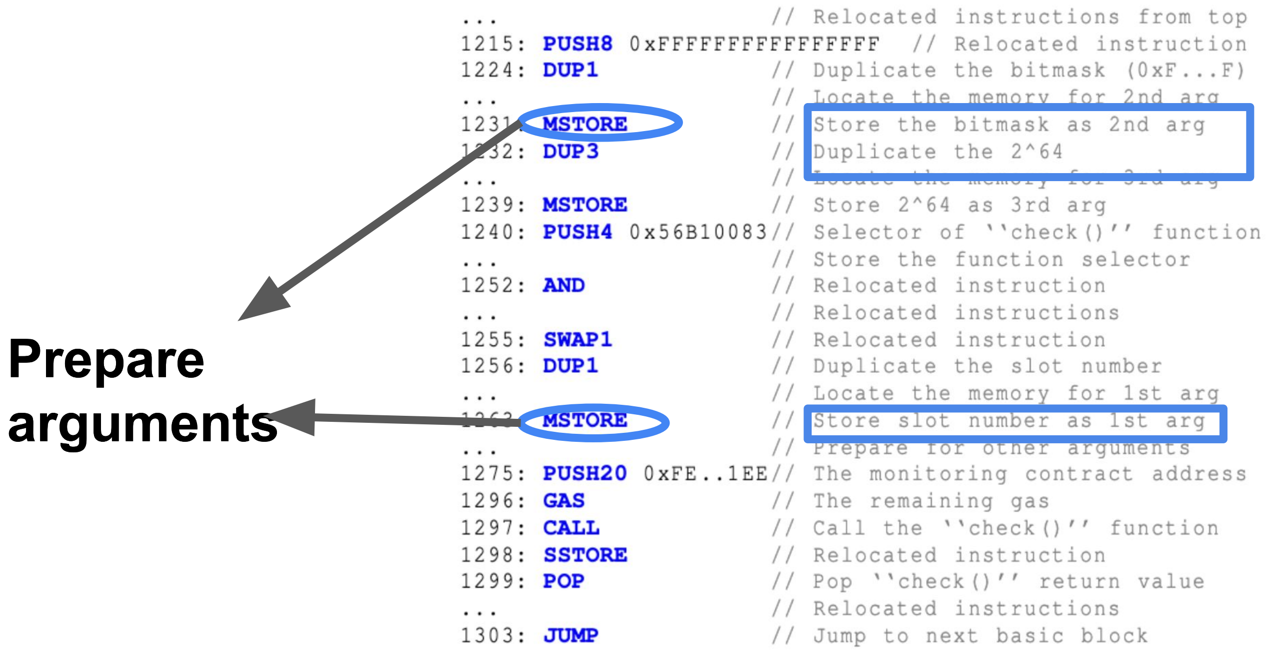
Write New Data


(e) Copy and mask new data, and left-shift it to 8 – 15 bytes
(f) Insert data
(g) Write storage
1194: DUP4
1195: PUSH8 0xF...F(16Fs)
1204: AND
1205: MUL
1206: OR
1207: SWAP1
1208: SSTORE
0...0 | 0...0 | 0...1 | 0...0 — 0x10...0 (2^{64})
X...X | X...X | 0...0 | X...X — cleared data
0...0 | 0...0 | 0...1 | 0...0 — 0x10...0 (2^{64})
0...0 | 0...0 | 0...0 | Y...Y — data to write
X...X | X...X | 0...0 | X...X — cleared data
0...0 | 0...0 | 0...1 | 0...0 — 0x10...0 (2^{64})
0...0 | 0...0 | 0...0 | Y...Y — data to write
0...0 | 0...0 | 0...0 | F...F — 0xF...F (mask)
X...X | X...X | 0...0 | X...X — cleared data
0...0 | 0...0 | 0...1 | 0...0 — 0x10...0 (2^{64})
0...0 | 0...0 | 0...0 | Y...Y — masked new data
X...X | X...X | 0...0 | X...X — cleared data
0...0 | 0...0 | Y...Y | 0...0 — left-shifted
X...X | X...X | Y...Y | X...X — data inserted
X...X | X...X | Y...Y | X...X — prepared data
0...0 | 0...0 | 0...0 | 0..XX — slot # N
Write prepared data to storage at slot # N

Partial Patch Bytecode

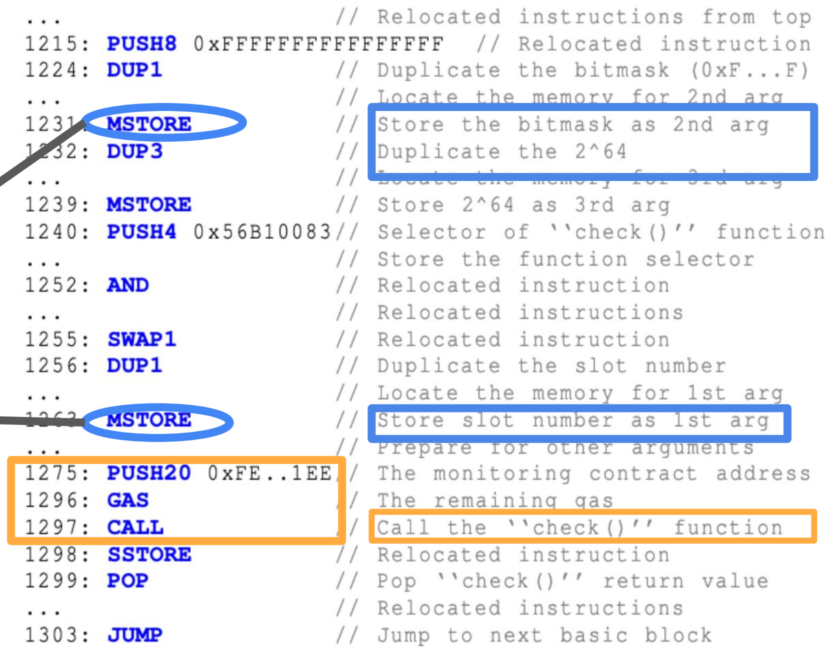
```
...                           // Relocated instructions from top
1215: PUSH8 0xFFFFFFFFFFFFFFFF   // Relocated instruction
1224: DUP1                    // Duplicate the bitmask (0xF...F)
...                           // Locate the memory for 2nd arg
1231: MSTORE                  // Store the bitmask as 2nd arg
1232: DUP3                    // Duplicate the 2^64
...                           // Locate the memory for 3rd arg
1239: MSTORE                  // Store 2^64 as 3rd arg
1240: PUSH4 0x56B10083// Selector of ``check()'' function
...                           // Store the function selector
1252: AND                     // Relocated instruction
...                           // Relocated instructions
1255: SWAP1                   // Relocated instruction
1256: DUP1                    // Duplicate the slot number
...                           // Locate the memory for 1st arg
1263: MSTORE                  // Store slot number as 1st arg
...                           // Prepare for other arguments
1275: PUSH20 0xFE..1EE// The monitoring contract address
1296: GAS                     // The remaining gas
1297: CALL                    // Call the ``check()'' function
1298: SSTORE                  // Relocated instruction
1299: POP                     // Pop ``check()'' return value
...                           // Relocated instructions
1303: JUMP                    // Jump to next basic block
```

Partial Patch Bytecode



```
...                           // Relocated instructions from top
1215: PUSH8 0xFFFFFFFFFFFFFFFF  // Relocated instruction
1224: DUP1                    // Duplicate the bitmask (0xF...F)
...                           // Locate the memory for 2nd arg
1231: MSTORE                  // Store the bitmask as 2nd arg
1232: DUP3                    // Duplicate the 2^64
...                           // Locate the memory for 3rd arg
1239: MSTORE                  // Store 2^64 as 3rd arg
1240: PUSH4 0x56B10083// Selector of ''check()'' function
...                           // Store the function selector
1252: AND                     // Relocated instruction
...                           // Relocated instructions
1255: SWAP1                   // Relocated instruction
1256: DUP1                    // Duplicate the slot number
...                           // Locate the memory for 1st arg
1263: MSTORE                  // Store slot number as 1st arg
...                           // Prepare for other arguments
1275: PUSH20 0xFE..1EE// The monitoring contract address
1296: GAS                     // The remaining gas
1297: CALL                    // Call the ''check()'' function
1298: SSTORE                  // Relocated instruction
1299: POP                     // Pop ''check()'' return value
...                           // Relocated instructions
1303: JUMP                    // Jump to next basic block
```

Partial Patch Bytecode



```
...                           // Relocated instructions from top
1215: PUSH8 0xFFFFFFFFFFFFFFFF  // Relocated instruction
1224: DUP1                    // Duplicate the bitmask (0xF...F)
...                           // Locate the memory for 2nd arg
1231: MSTORE                  // Store the bitmask as 2nd arg
1232: DUP3                    // Duplicate the 2^64
...                           // Locate the memory for 3rd arg
1239: MSTORE                  // Store 2^64 as 3rd arg
1240: PUSH4 0x56B10083// Selector of ''check()'' function
...                           // Store the function selector
1252: AND                     // Relocated instruction
...                           // Relocated instructions
1255: SWAP1                   // Relocated instruction
1256: DUP1                    // Duplicate the slot number
...                           // Locate the memory for 1st arg
1263: MSTORE                  // Store slot number as 1st arg
...                           // Prepare for other arguments
1275: PUSH20 0xFE..1EE        // The monitoring contract address
1296: GAS                     // The remaining gas
1297: CALL                    // Call the ''check()'' function
1298: SSTORE                  // Relocated instruction
1299: POP                     // Pop ''check()'' return value
...                           // Relocated instructions
1303: JUMP                    // Jump to next basic block
```

Prepare arguments

Send to check()

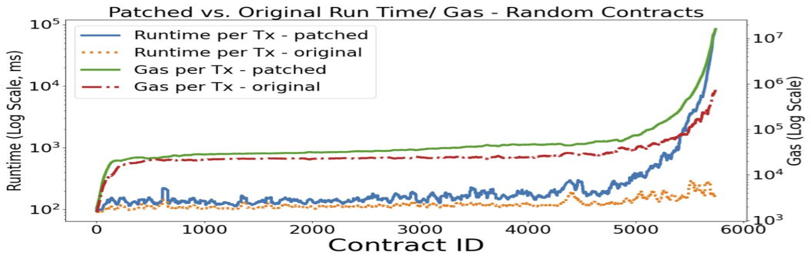
THE UNIVERSITY OF UTAH

Evaluation Dataset

- Vulnerable: 12,526 vulnerable contracts from CRUSH(***NDSS ’24***)

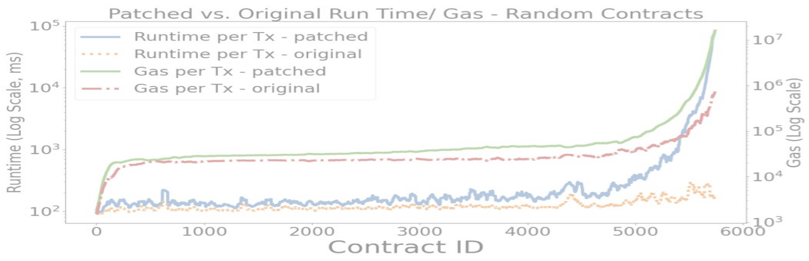
- Random: 6,018 upgradeable contracts randomly selected from Etherscan

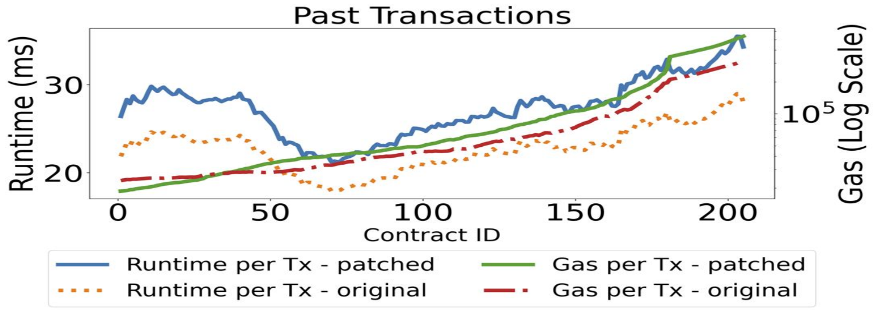
Random Sample: Past Transactions Replayed



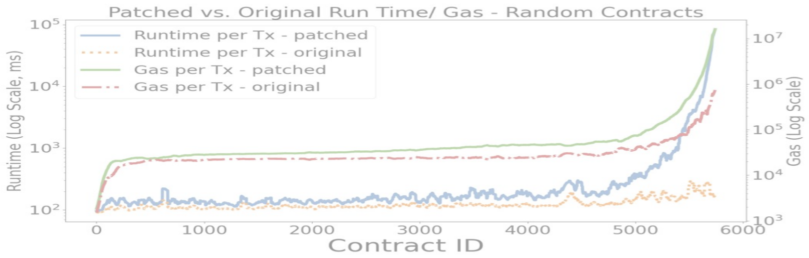
THE UNIVERSITY OF UTAH

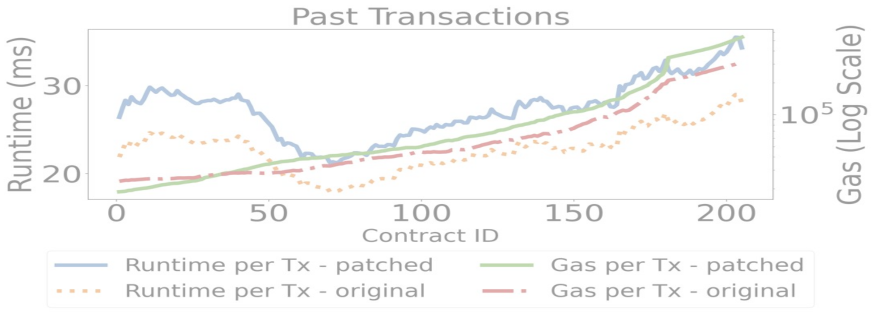
Vulnerable: Past Transactions Replayed

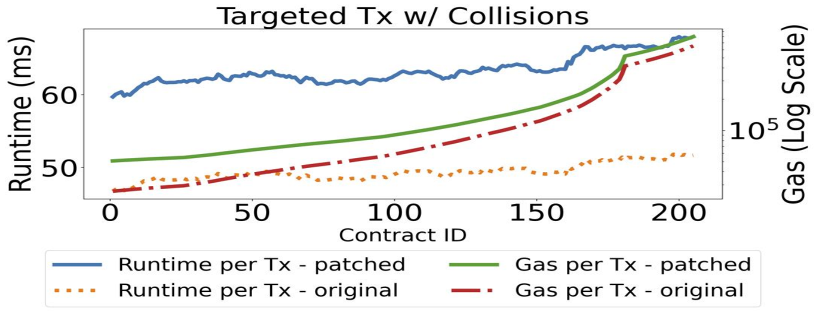




Vulnerable: Attack Transactions Replayed

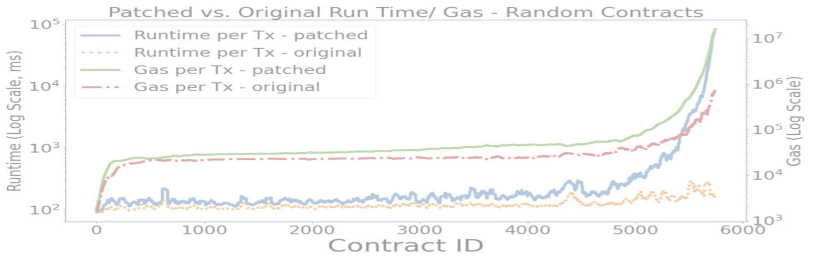


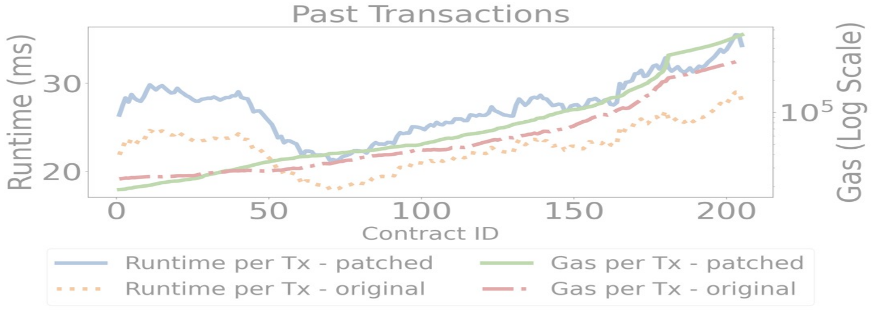


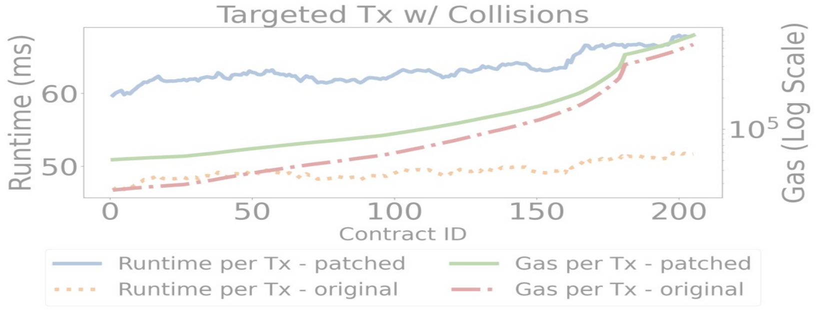


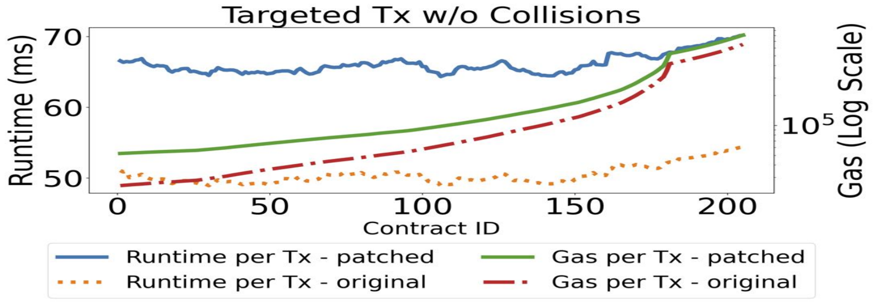
THE UNIVERSITY OF UTAH

Vulnerable: Safe Transactions Replayed

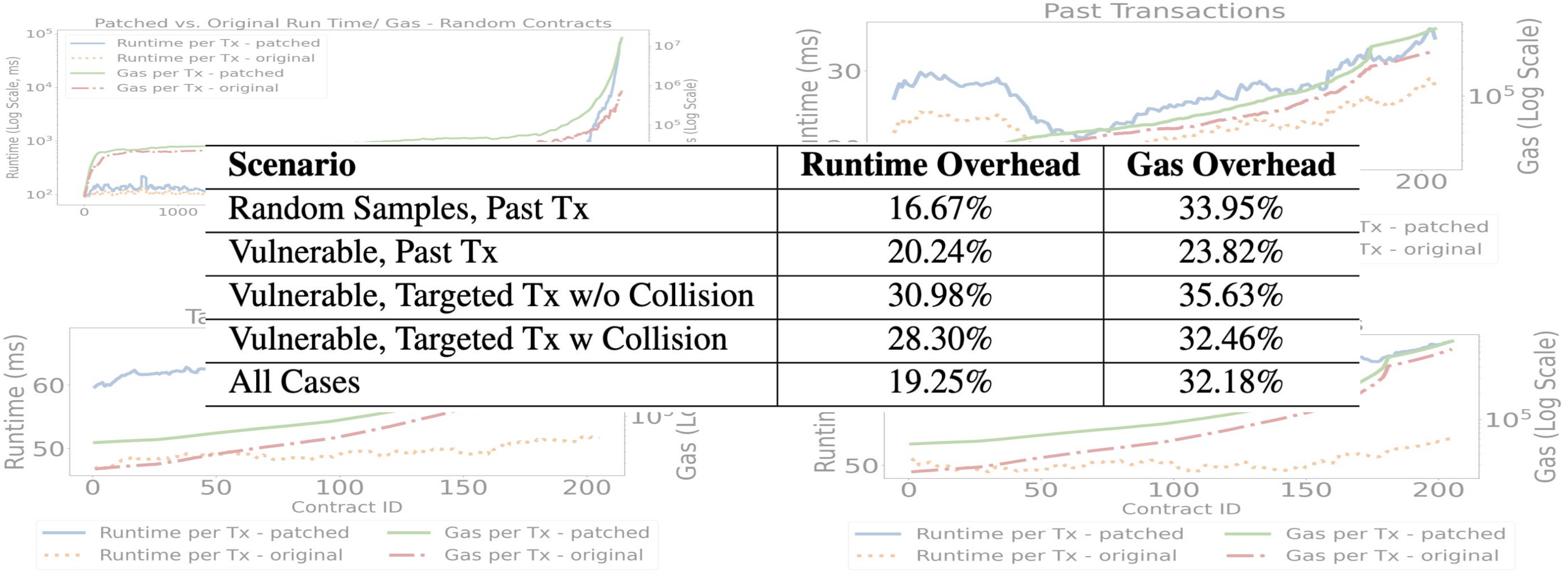






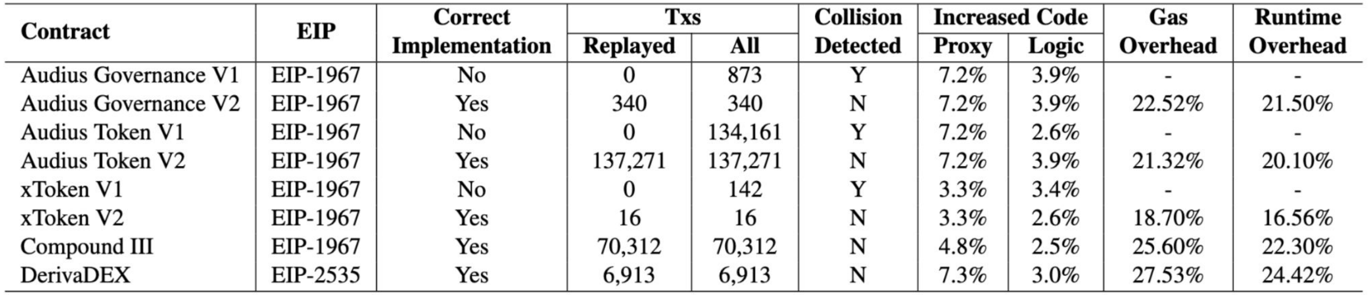


Runtime and Gas Overhead



Scenario	Runtime Overhead	Gas Overhead
Random Samples, Past Tx	16.67%	33.95%
Vulnerable, Past Tx	20.24%	23.82%
Vulnerable, Targeted Tx w/o Collision	30.98%	35.63%
Vulnerable, Targeted Tx w Collision	28.30%	32.46%
All Cases	19.25%	32.18%

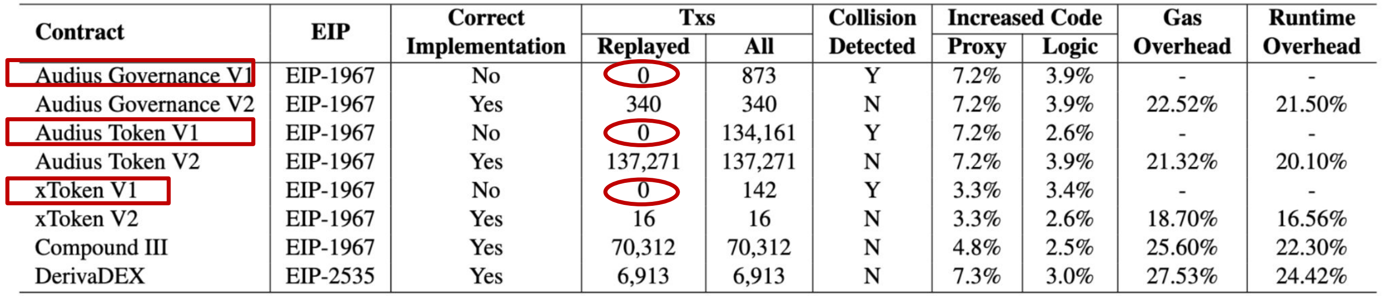
Case Study



Contract	EIP	Correct Implementation	Txs		Collision Detected	Increased Code		Gas Overhead	Runtime Overhead
			Replayed	All		Proxy	Logic		
Audius Governance V1	EIP-1967	No	0	873	Y	7.2%	3.9%	-	-
Audius Governance V2	EIP-1967	Yes	340	340	N	7.2%	3.9%	22.52%	21.50%
Audius Token V1	EIP-1967	No	0	134,161	Y	7.2%	2.6%	-	-
Audius Token V2	EIP-1967	Yes	137,271	137,271	N	7.2%	3.9%	21.32%	20.10%
xToken V1	EIP-1967	No	0	142	Y	3.3%	3.4%	-	-
xToken V2	EIP-1967	Yes	16	16	N	3.3%	2.6%	18.70%	16.56%
Compound III	EIP-1967	Yes	70,312	70,312	N	4.8%	2.5%	25.60%	22.30%
DerivaDEX	EIP-2535	Yes	6,913	6,913	N	7.3%	3.0%	27.53%	24.42%

THE UNIVERSITY OF UTAH

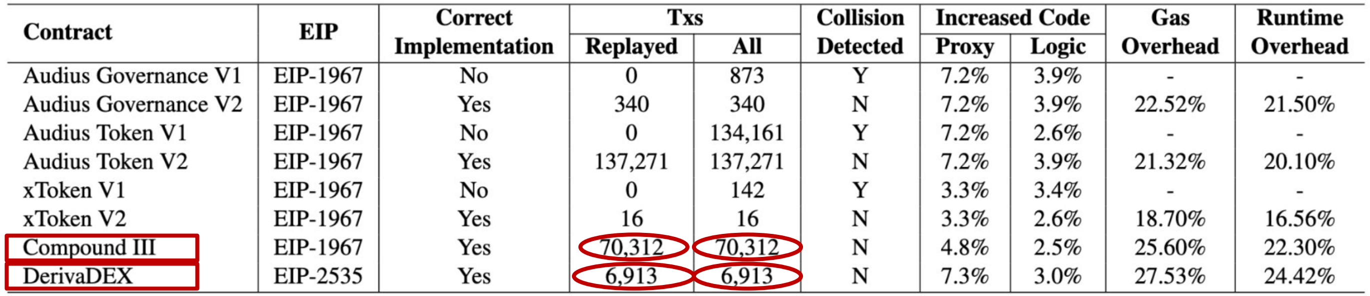
Case Study



Contract	EIP	Correct Implementation	Txs		Collision Detected	Increased Code		Gas Overhead	Runtime Overhead
			Replayed	All		Proxy	Logic		
Audius Governance V1	EIP-1967	No	0	873	Y	7.2%	3.9%	-	-
Audius Governance V2	EIP-1967	Yes	340	340	N	7.2%	3.9%	22.52%	21.50%
Audius Token V1	EIP-1967	No	0	134,161	Y	7.2%	2.6%	-	-
Audius Token V2	EIP-1967	Yes	137,271	137,271	N	7.2%	3.9%	21.32%	20.10%
xToken V1	EIP-1967	No	0	142	Y	3.3%	3.4%	-	-
xToken V2	EIP-1967	Yes	16	16	N	3.3%	2.6%	18.70%	16.56%
Compound III	EIP-1967	Yes	70,312	70,312	N	4.8%	2.5%	25.60%	22.30%
DerivaDEX	EIP-2535	Yes	6,913	6,913	N	7.3%	3.0%	27.53%	24.42%

THE UNIVERSITY OF UTAH

Case Study



Contract	EIP	Correct Implementation	Txs		Collision Detected	Increased Code		Gas Overhead	Runtime Overhead
			Replayed	All		Proxy	Logic		
Audius Governance V1	EIP-1967	No	0	873	Y	7.2%	3.9%	-	-
Audius Governance V2	EIP-1967	Yes	340	340	N	7.2%	3.9%	22.52%	21.50%
Audius Token V1	EIP-1967	No	0	134,161	Y	7.2%	2.6%	-	-
Audius Token V2	EIP-1967	Yes	137,271	137,271	N	7.2%	3.9%	21.32%	20.10%
xToken V1	EIP-1967	No	0	142	Y	3.3%	3.4%	-	-
xToken V2	EIP-1967	Yes	16	16	N	3.3%	2.6%	18.70%	16.56%
Compound III	EIP-1967	Yes	70,312	70,312	N	4.8%	2.5%	25.60%	22.30%
DerivaDEX	EIP-2535	Yes	6,913	6,913	N	7.3%	3.0%	27.53%	24.42%

Conclusion

- We present CollisionRepair, an automated patching system for mitigating storage collision vulnerabilities.
- CollisionRepair defines ownership model to track storage usage.
- Evaluated on 12,526 real-world contracts, CollisionRepair effectively prevents storage collisions while preserving normal functionality.

Thank You!