UC SANTA BARBARA

SEC LAB

Not your Type! Detecting Storage Collision Vulnerabilities in Ethereum Smart Contracts

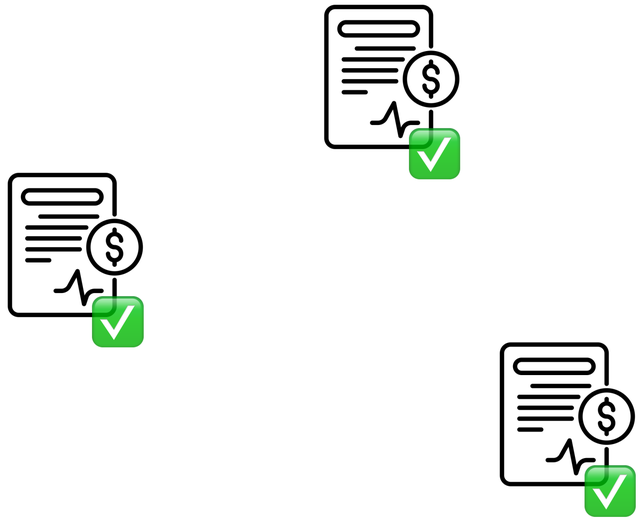
Nicola Ruaro

Nicola Ruaro, Fabio Gritti, Robert McLaughlin, Ilya Grishchenko,
Christopher Kruegel, Giovanni Vigna

NDSS SYMPOSIUM/2024

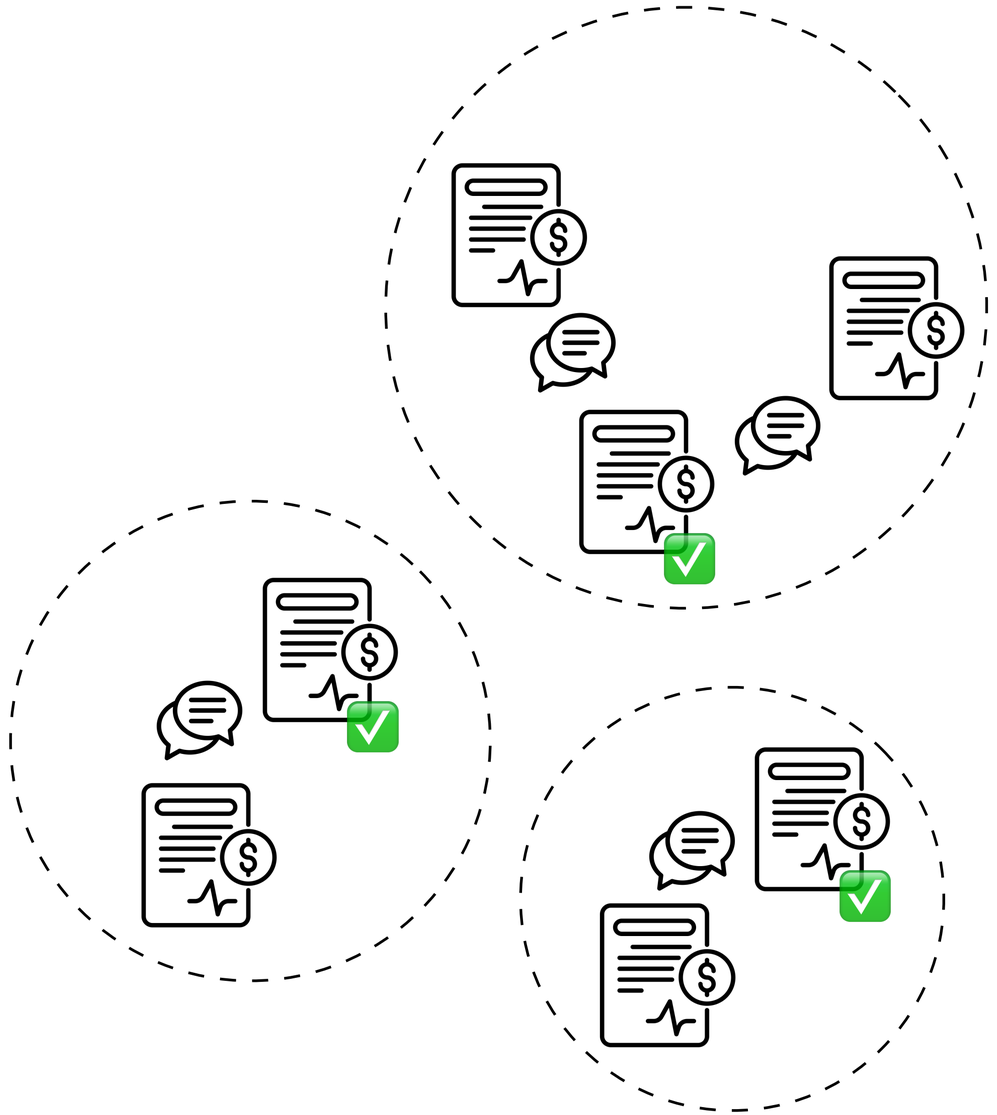
Motivation

- Classic contract attacks are well-studied (tainted calls, re-entrancy...)



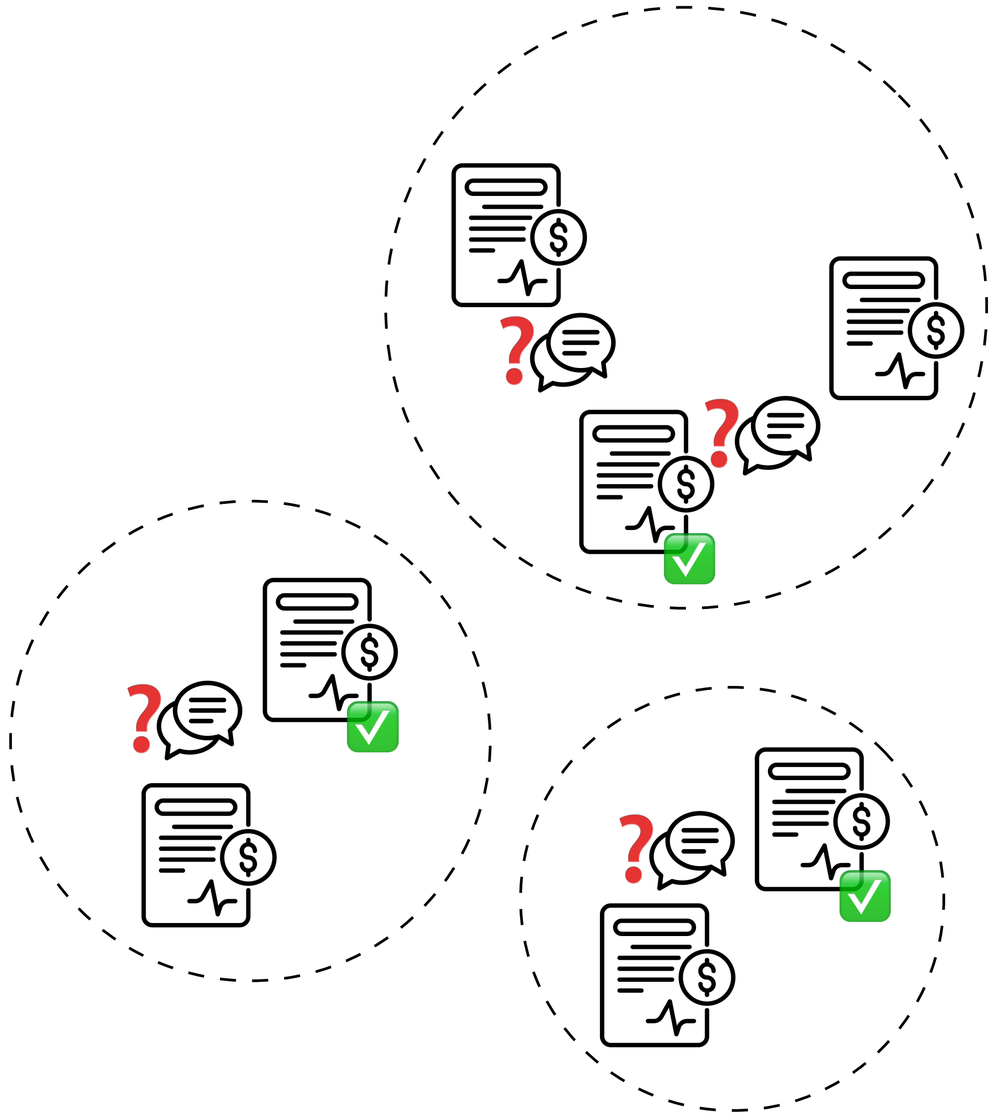
Motivation

- Classic contract attacks are well-studied (tainted calls, re-entrancy...)
- Decentralized Finance → **interactions** are increasingly complex (DeFi Protocols)



Motivation

- Classic contract attacks are well-studied (tainted calls, re-entrancy...)
- Decentralized Finance → **interactions** are increasingly complex (DeFi Protocols)
- This **attack surface** is under-studied:
  - Privilege escalation
  - Oracle manipulation
  - **Storage collision**



Smart Contracts

Blockchain State

Balance (ETH)

Storage

Code

Program Counter

Gas

Contract

Code

Stack

Memory

Contract

Code

Stack

Memory

EVM

Smart Contracts

Blockchain State

Balance (ETH)

Storage

Code

Tx

Data

Value (ETH)

Gas

Origin

Program Counter

Gas

Contract

Code

Stack

Memory

Contract

Code

Stack

Memory

EVM

Smart Contracts

Blockchain State

Balance (ETH)

Storage

Code

Tx

Data

Value (ETH)

Gas

Origin

```
CALLDATALOAD
PUSH 0x20
TIMESTAMP
MSTORE
PUSH 0X20
MLOAD
PUSH 0x1
SSTORE
...
(DELEGATE)CALL
```

EVM bytecode

Program Counter

Gas

Contract

Code

Stack

Memory

Contract

Code

Stack

Memory

EVM

Smart Contracts

Blockchain State

Balance (ETH)

Storage
{ slot : value }

Code

Tx

Data

Value (ETH)

Gas

Origin

```
CALLDATALOAD
PUSH 0x20
TIMESTAMP
MSTORE
PUSH 0X20
MLOAD
PUSH 0x1
SSTORE
...
(DELEGATE)CALL
```

EVM bytecode

Contract

Code

Stack

Memory

Contract

Code

Stack

Memory

Program Counter

Gas

EVM

Smart Contracts

Blockchain State

Balance (ETH)

Storage
{ slot : value }

Code

Tx

Data

Value (ETH)

Gas

Origin

```
CALLDATALOAD
PUSH 0x20
TIMESTAMP
MSTORE
PUSH 0X20
MLOAD
PUSH 0x1
SSTORE
...
(DELEGATE)CALL
```

EVM bytecode

Contract

Code

Stack

Memory

Contract

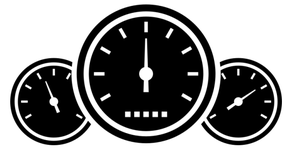
Code

Stack

Memory

Program Counter

Gas



EVM

Smart Contracts

Blockchain State

Balance (ETH)

Storage
{ slot : value }

Code

Tx

Data

Value (ETH)

Gas

Origin

```
CALLDATALOAD
PUSH 0x20
TIMESTAMP
MSTORE
PUSH 0X20
MLOAD
PUSH 0x1
SSTORE
...
(DELEGATE) CALL
```

EVM bytecode

Contract

Code

Stack

Memory

Contract

Code

Stack

Memory

Program Counter

Gas

EVM

Blockchain Streaming Platform Audius Victim of $6M Hack

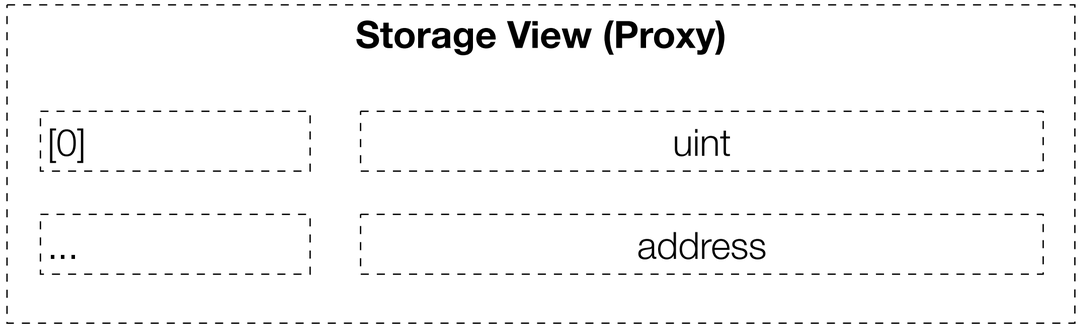
The hacker was able to exploit a bug in the company's smart contract.

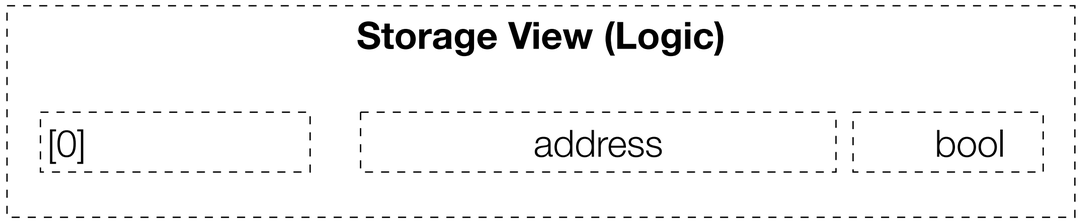
Storage Collisions

```
contract Proxy {
    uint visits;        // storage slot [0x0]
    address LOGIC;      // storage slot [...]

    constructor() {
        LOGIC = [...];
        visits = 0;
        LOGIC.initialize();
    }
    fallback() external {
        visits += 1;
        LOGIC.delegatecall(msg.data);
    }
}

contract Logic {
    bool initialized;  // storage slot [0x0]
    address admin;     // storage slot [0x0]

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

Storage Collisions

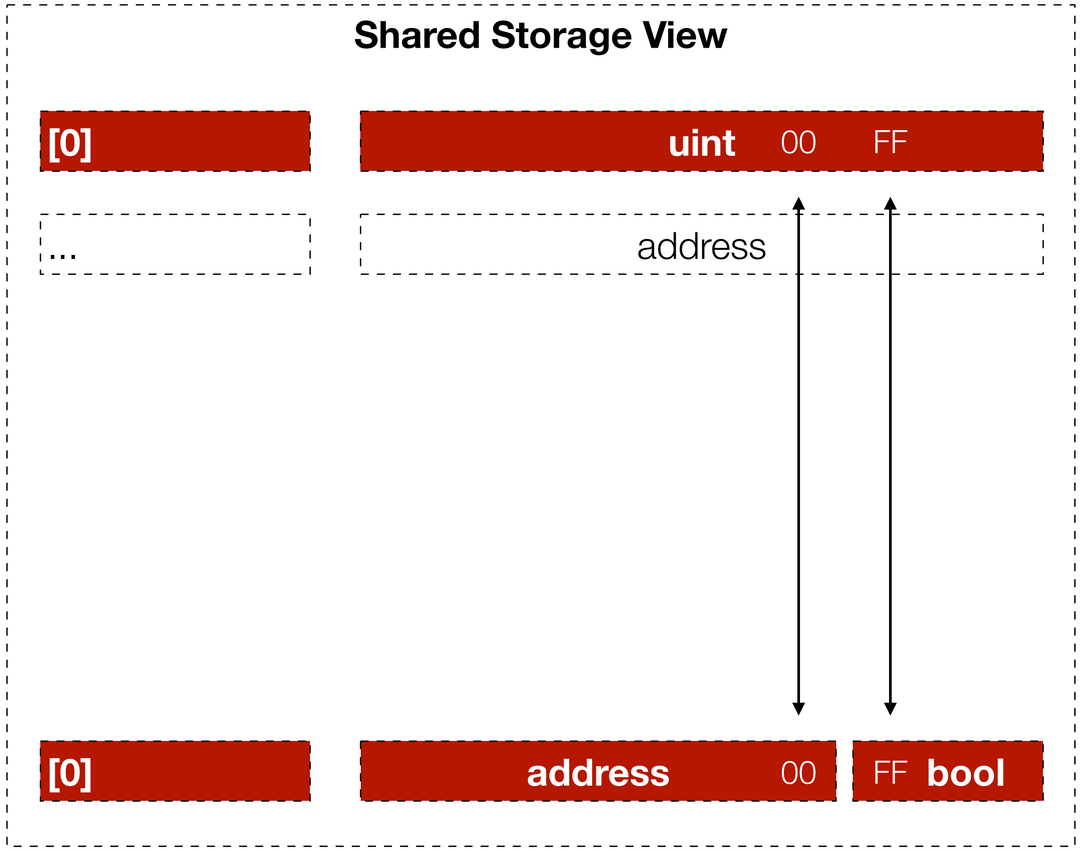
```
1  contract Proxy {
2      uint visits;         // storage slot [0x0]
3      address LOGIC;       // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract Logic {
17     bool initialized;  // storage slot [0x0]
18     address admin;     // storage slot [0x0]
19
20     function initialize() external {
21         require(!initialized);
22         initialized = 1;
23         admin = msg.sender;
24     }
25     function withdraw() external {
26         require(msg.sender == admin);
27         payable(admin).transfer(this.balance);
28     }
29 }
```





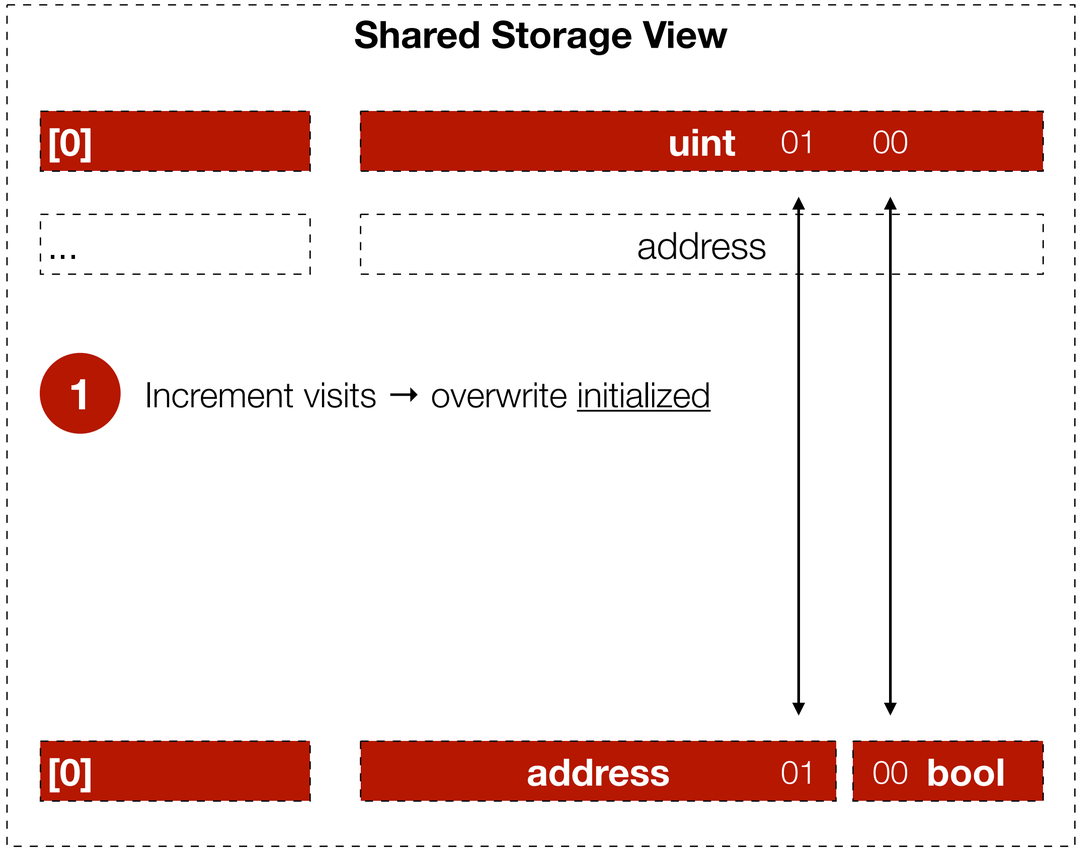
Storage Collisions

```
1  contract Proxy {
2      uint visits;          // storage slot [0x0]
3      address LOGIC;        // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract Logic {
17     bool initialized;  // storage slot [0x0]
18     address admin;     // storage slot [0x0]
19
20     function initialize() external {
21         require(!initialized);
22         initialized = 1;
23         admin = msg.sender;
24     }
25     function withdraw() external {
26         require(msg.sender == admin);
27         payable(admin).transfer(this.balance);
28     }
29 }
```



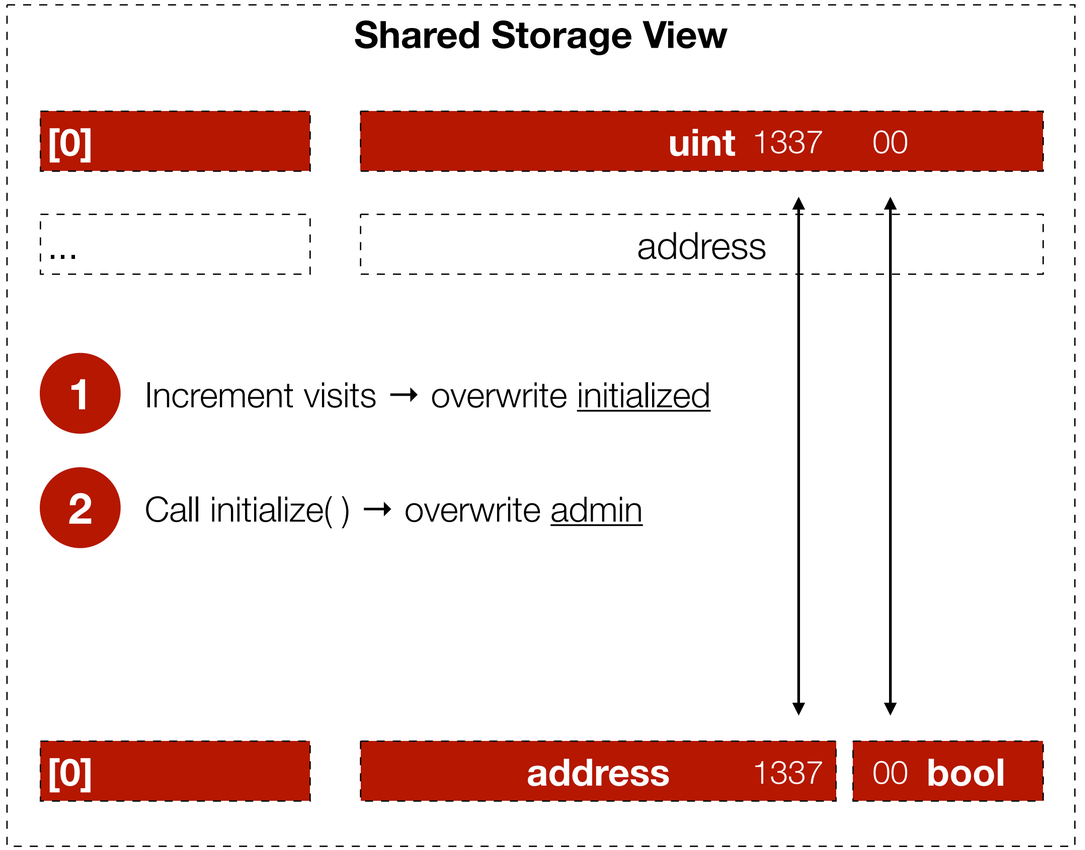
Storage Collisions

```
1  contract Proxy {
2      uint visits;          // storage slot [0x0]
3      address LOGIC;        // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract Logic {
17     bool initialized;  // storage slot [0x0]
18     address admin;     // storage slot [0x0]
19
20     function initialize() external {
21         require(!initialized);
22         initialized = 1;
23         admin = msg.sender;
24     }
25     function withdraw() external {
26         require(msg.sender == admin);
27         payable(admin).transfer(this.balance);
28     }
29 }
```



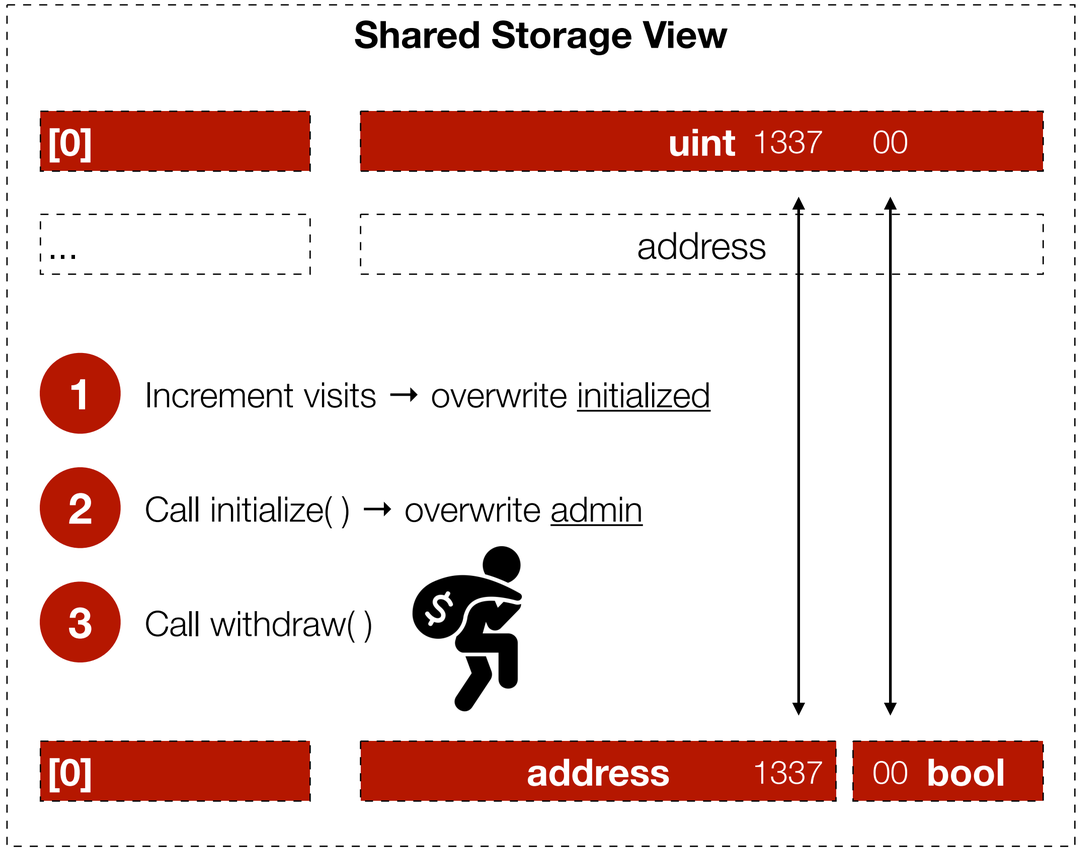
Storage Collisions

```
1  contract Proxy {
2      uint visits;          // storage slot [0x0]
3      address LOGIC;        // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract Logic {
17     bool initialized;  // storage slot [0x0]
18     address admin;     // storage slot [0x0]
19
20     function initialize() external {
21         require(!initialized);
22         initialized = 1;
23         admin = msg.sender;
24     }
25     function withdraw() external {
26         require(msg.sender == admin);
27         payable(admin).transfer(this.balance);
28     }
29 }
```

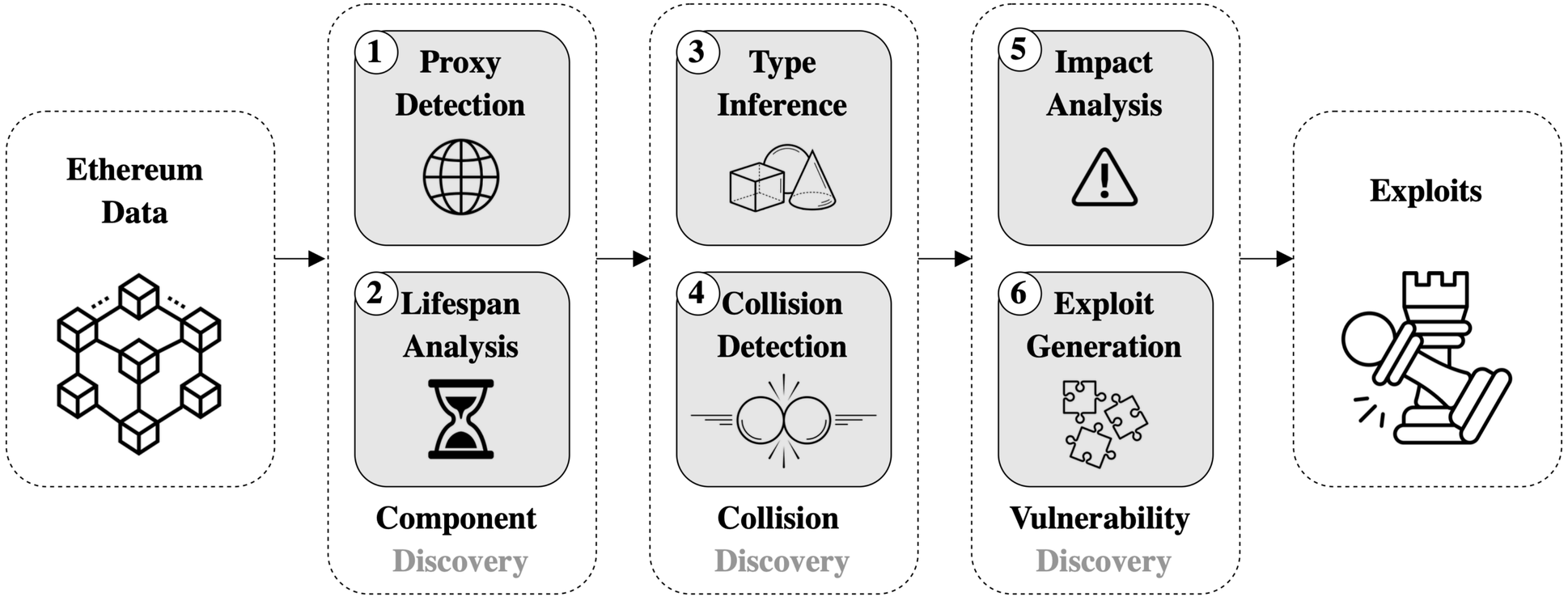


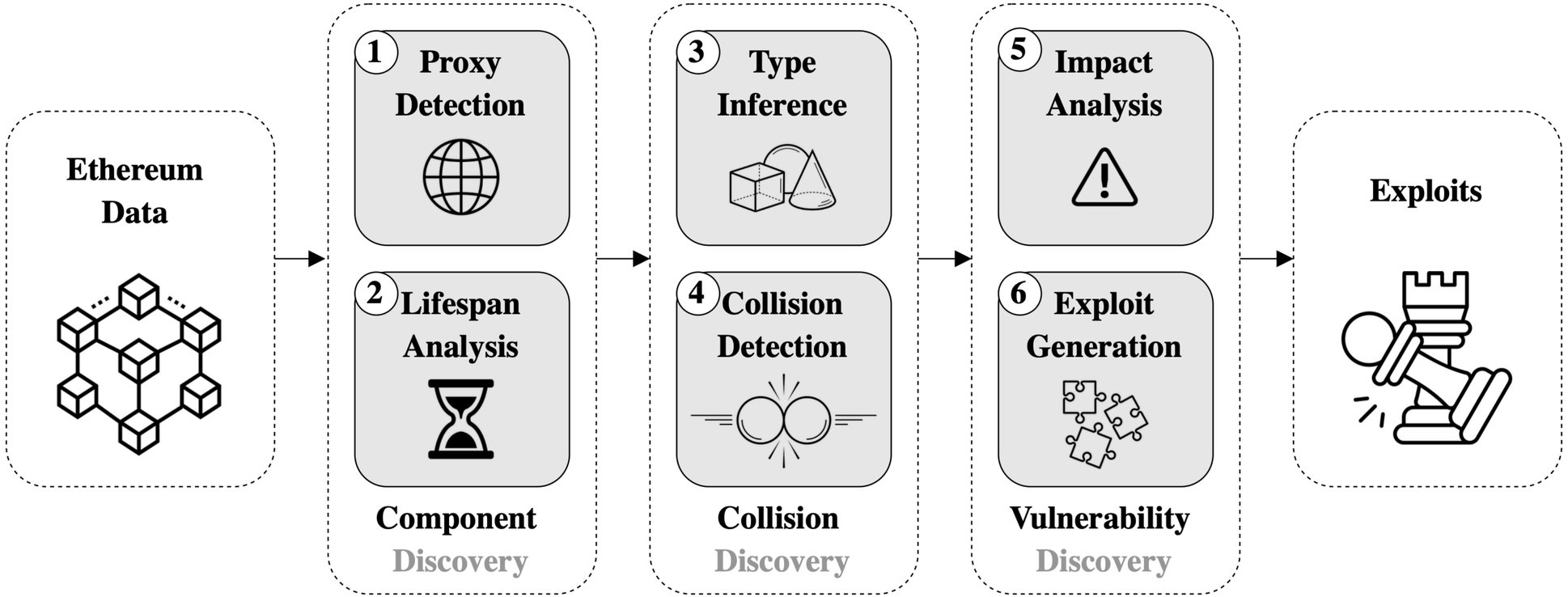
Storage Collisions

```
1  contract Proxy {
2      uint visits;          // storage slot [0x0]
3      address LOGIC;        // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract Logic {
17     bool initialized;     // storage slot [0x0]
18     address admin;        // storage slot [0x0]
19
20     function initialize() external {
21         require(!initialized);
22         initialized = 1;
23         admin = msg.sender;
24     }
25     function withdraw() external {
26         require(msg.sender == admin);
27         payable(admin).transfer(this.balance);
28     }
29 }
```



Detecting Storage Collisions: CRUSH

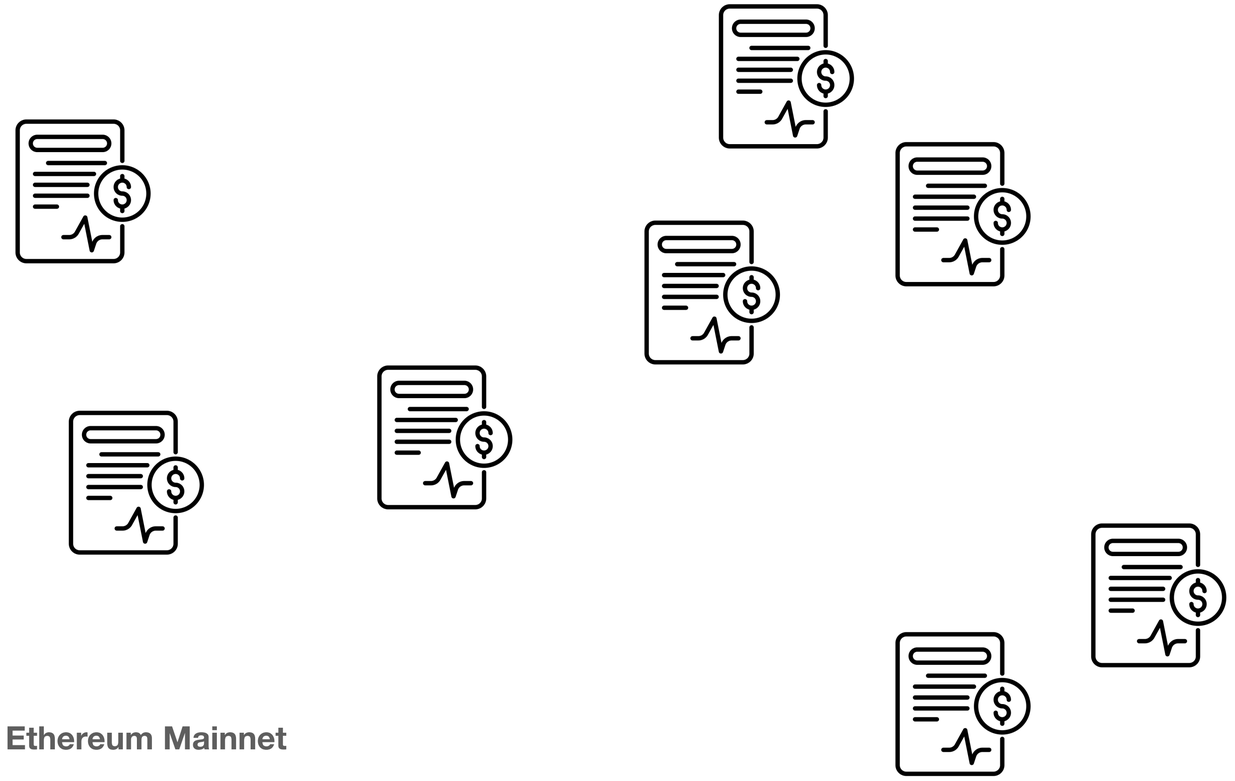

Ethereum Data
1 Proxy Detection
2 Lifespan Analysis
Component Discovery
3 Type Inference
4 Collision Detection
Collision Discovery
5 Impact Analysis
6 Exploit Generation
Vulnerability Discovery
Exploits



~~SOURCE CODE~~ → **EVM BYTECODE**

CRUSH

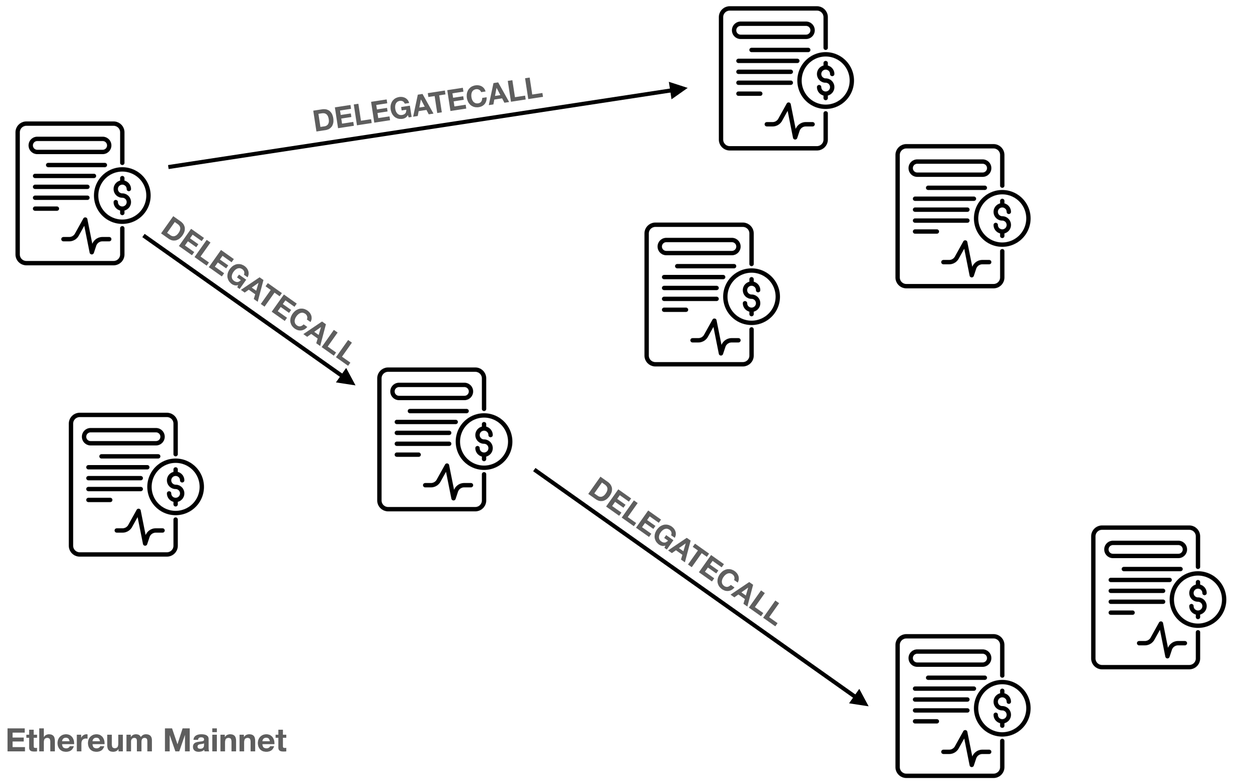
1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation



Ethereum Mainnet

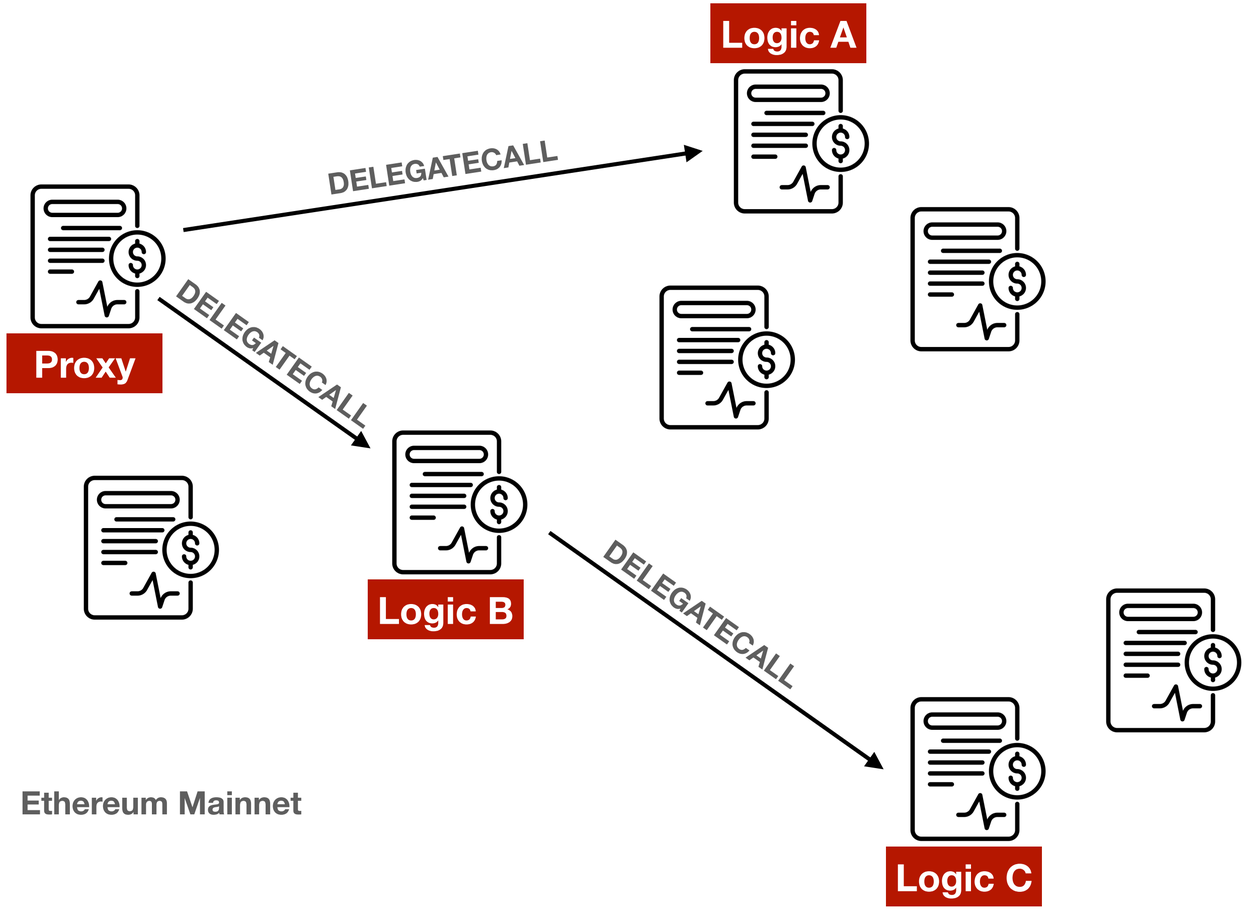
CRUSH

1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation



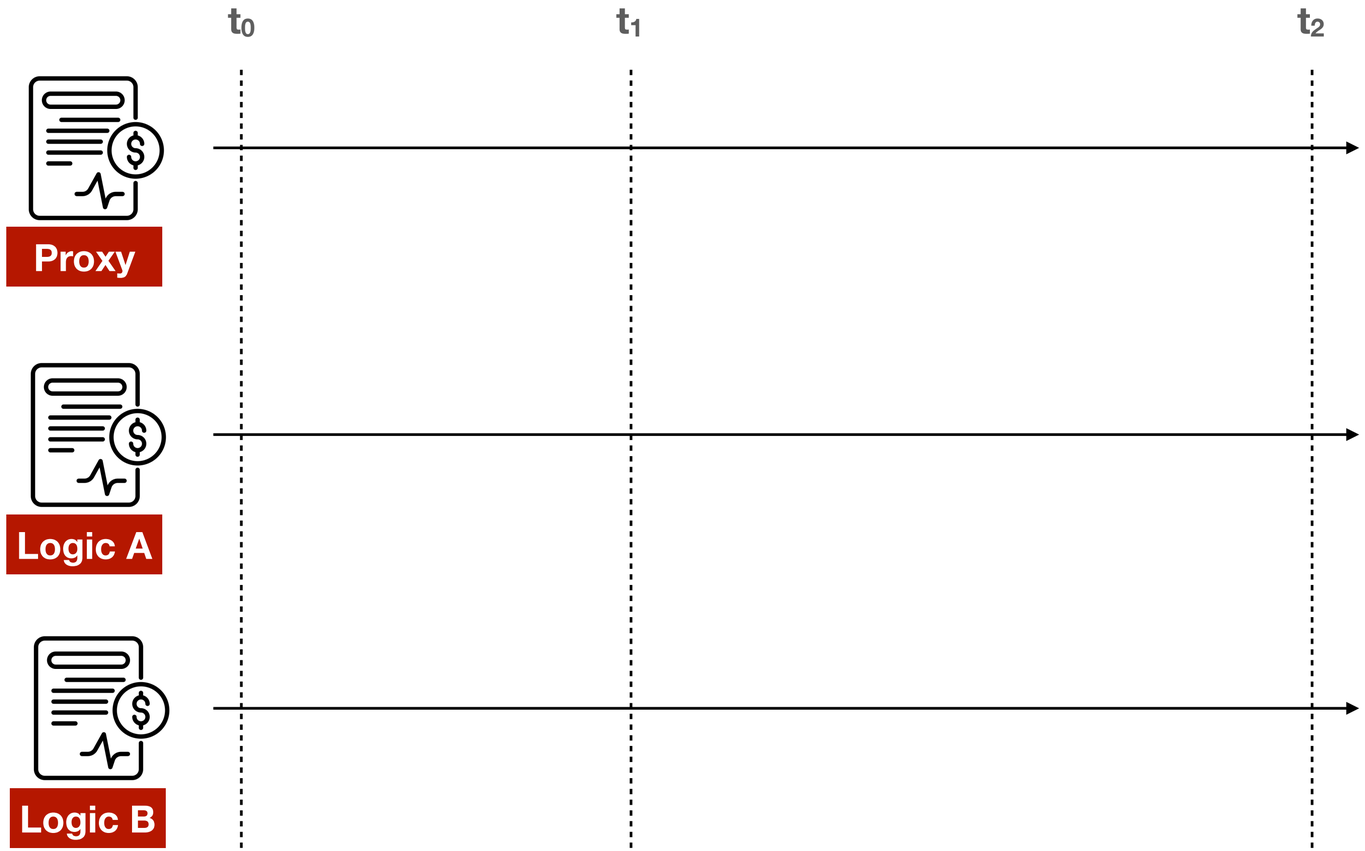
CRUSH

1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation

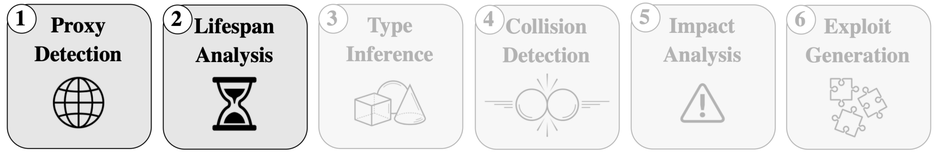


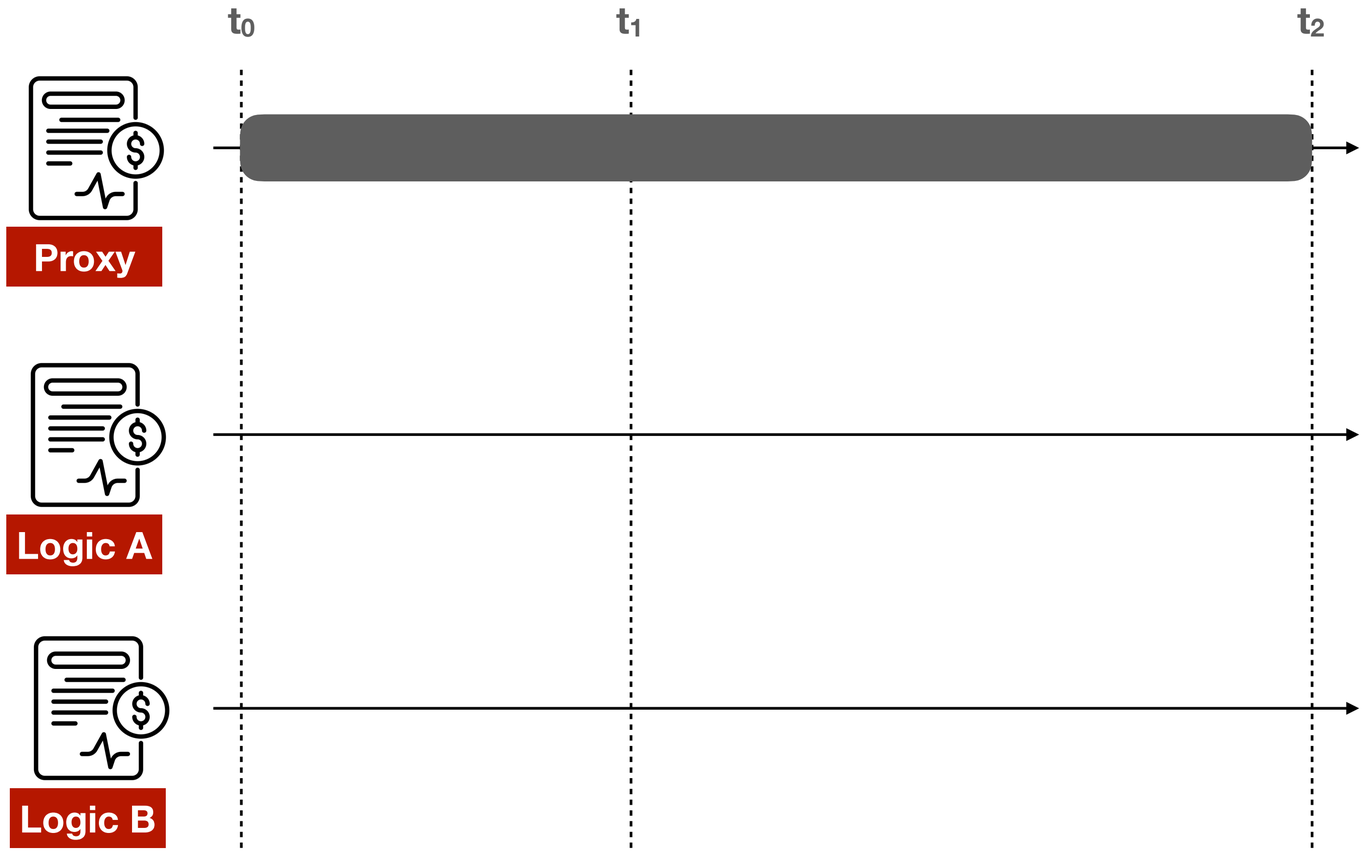
CRUSH

1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation



CRUSH

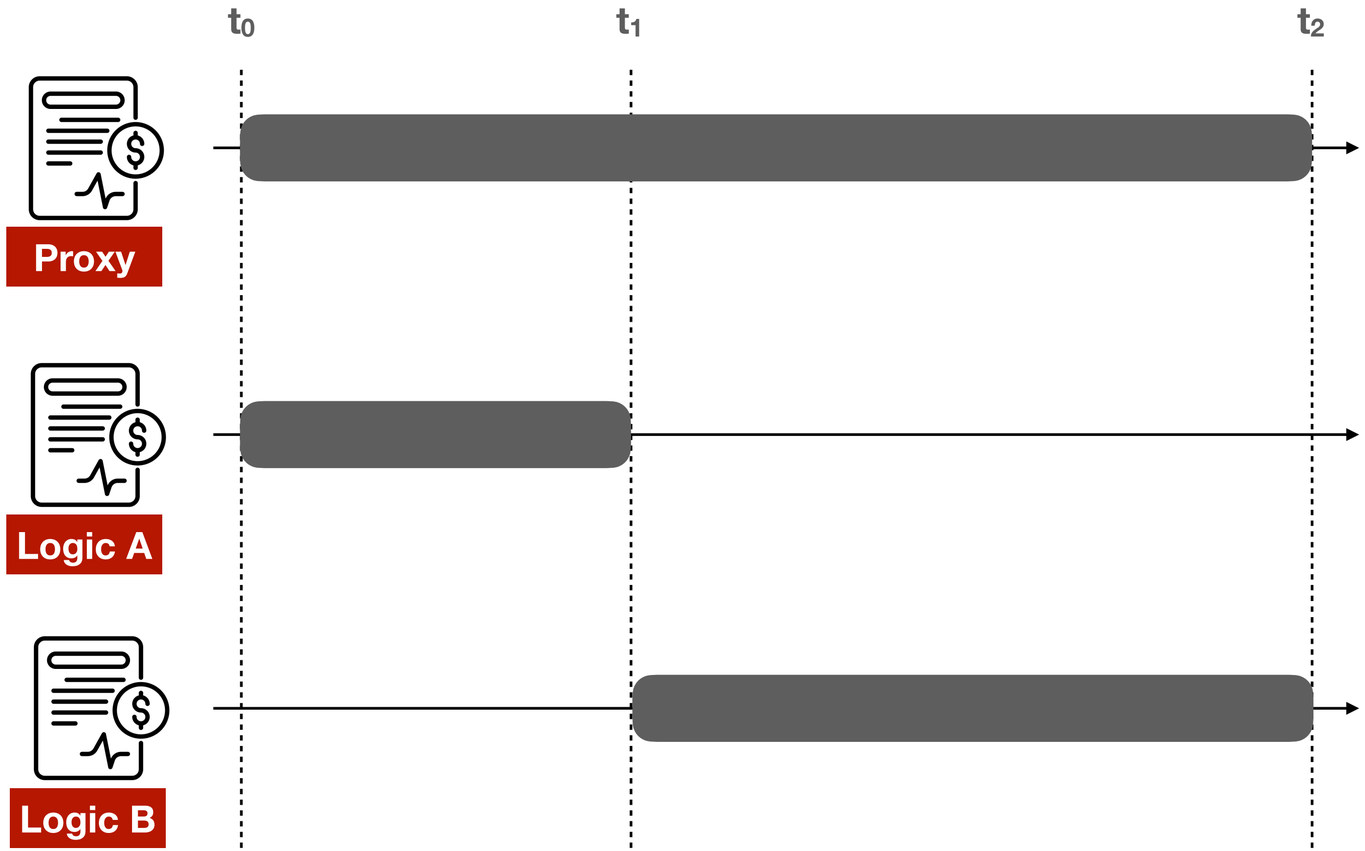




- **Proxy** - always active

CRUSH

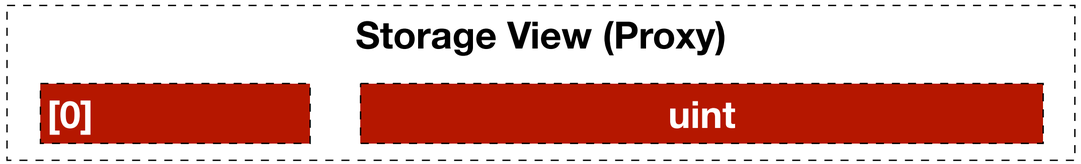
1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation

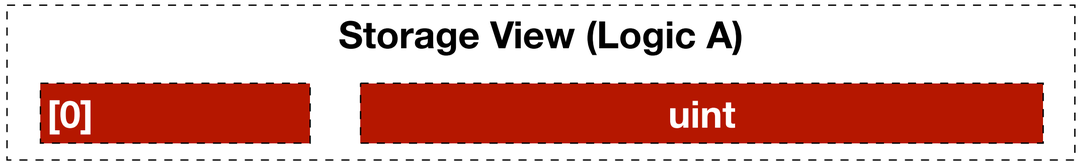


- **Proxy** - always active
- **Logic** - historical values

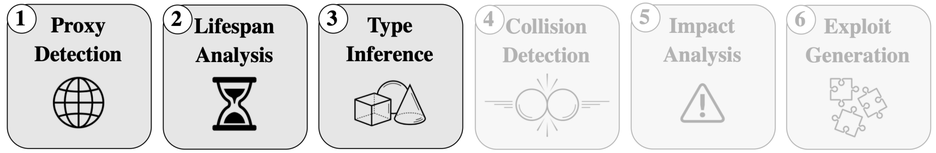
CRUSH

1 Proxy Detection | 2 Lifespan Analysis | 3 Type Inference | 4 Collision Detection | 5 Impact Analysis | 6 Exploit Generation



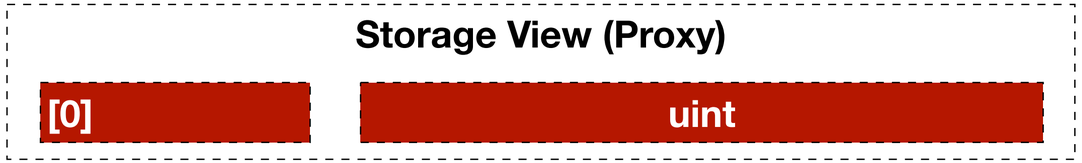


CRUSH



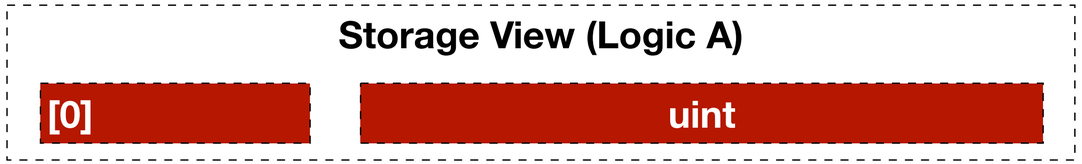

```
v1 = SLOAD 0x0
```

```
SSTORE v2, 0x0
```




```
v1 = SLOAD 0x0
```

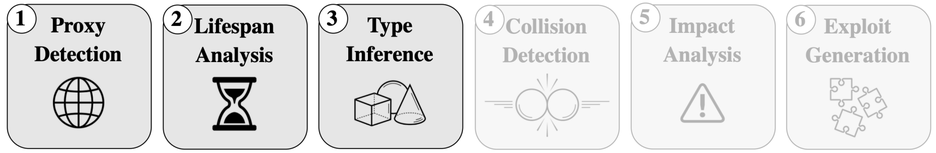
```
SSTORE v2, 0x0
```




```
v1 = SLOAD 0x0
```

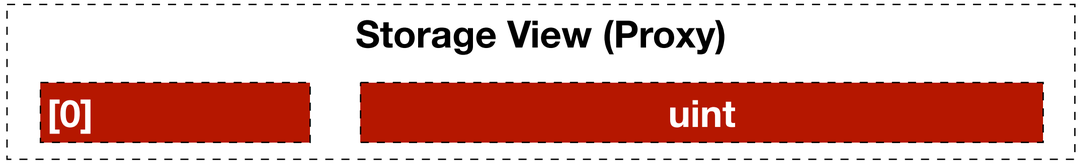
```
SSTORE v2, 0x0
```

CRUSH




```
v1 = SLOAD 0x0
```

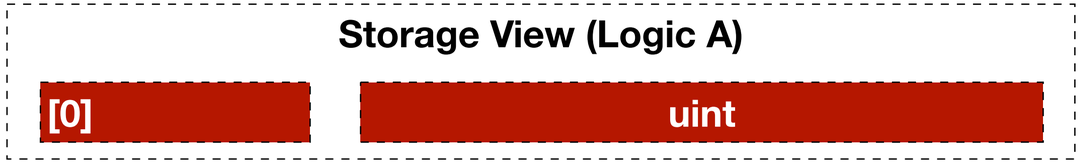
```
SSTORE v2, 0x0
```




```
MASK (uint): ff ff ff ff
```


```
v1 = SLOAD 0x0
```

```
SSTORE v2, 0x0
```



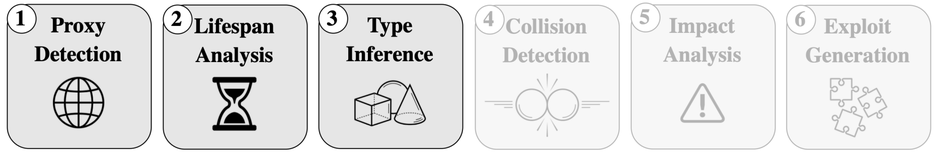

```
MASK (uint): ff ff ff ff
```


```
v1 = SLOAD 0x0
```

```
SSTORE v2, 0x0
```

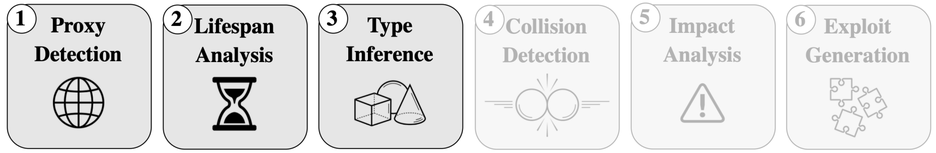

```
MASK (addr): ff ff ff 00
MASK (bool): 00 00 00 ff
```

CRUSH



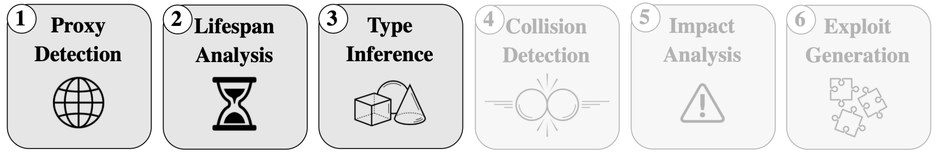
1. Find storage operations (SLOAD, SSTORE)

CRUSH



1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE

CRUSH




```
v1 = SLOAD 0x0
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

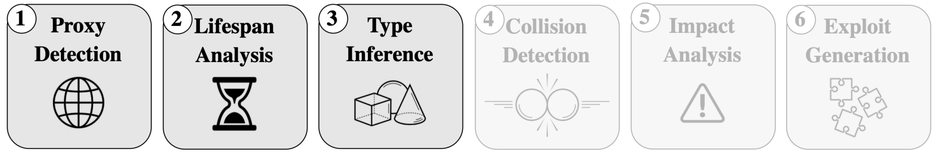
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

```
V2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE

CRUSH




```
v1 = SLOAD 0x0
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

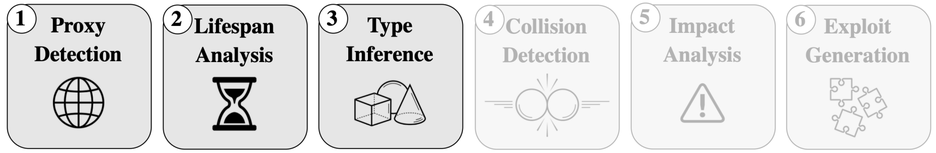
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

```
v1 = CONST 0x0
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE

CRUSH




```
v1 = SLOAD 0x0
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

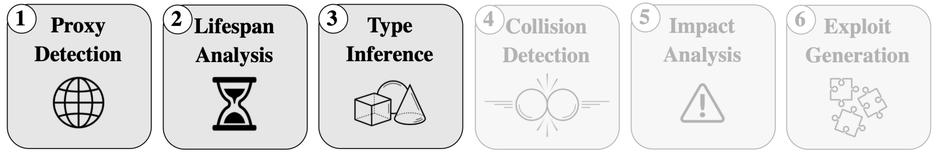
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
```

```
v1 = CONST 0x0
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD

CRUSH




```
v1 = SLOAD 0x0
CALL v1 ...
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
CALL v1 ...
```

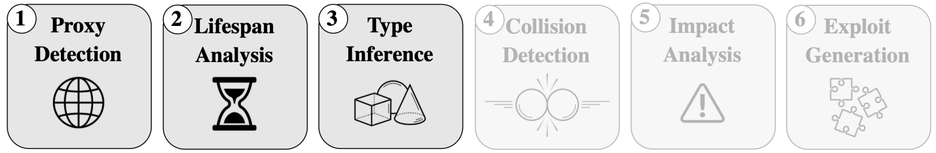
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
```

```
v1 = CONST 0x0
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD

CRUSH




```
v1 = SLOAD 0x0
CALL v1 ...
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
CALL v1 ...
```

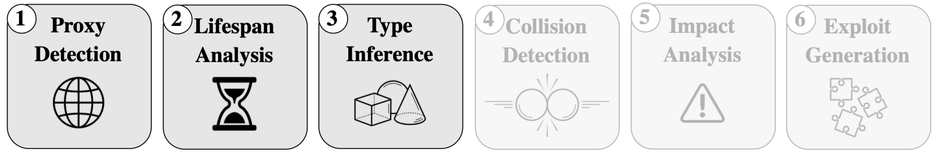
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
CALL v2 ...
```

```
v1 = CONST 0x0
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD

CRUSH




```
v1 = SLOAD 0x0
CALL v1 ...
```

```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
CALL v1 ...
```

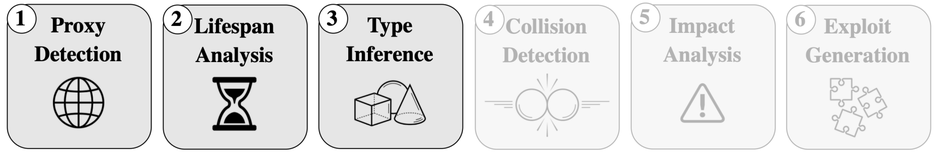
```
v2 = CONST 0x0
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
CALL v2 ...
```

```
v1 = CONST 0x0
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD
4. Symbolic Execution to track **access mask**

CRUSH




```
v1 = SLOAD 0x0
<sink>
```

```
v2 = <source>
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
<sink>
```

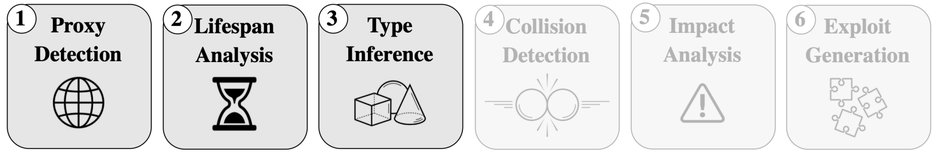
```
v2 = <source>
SSTORE v2, 0x0
```


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD
4. Symbolic Execution to track **access mask**

CRUSH




```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

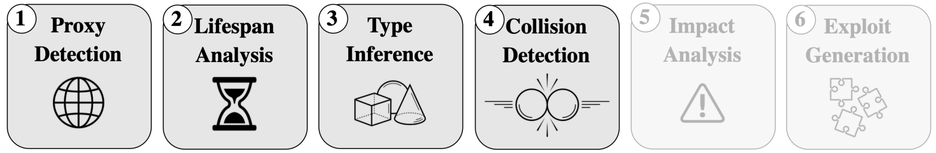
MASK: **ff ff ff 00**

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

MASK: **00 00 00 ff**

1. Find storage operations (SLOAD, SSTORE)
2. Backward slice on SSTORE
3. Forward slice on SLOAD
4. Symbolic Execution to track **access mask**

CRUSH



Proxy

```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**

Logic A

```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**

Logic B

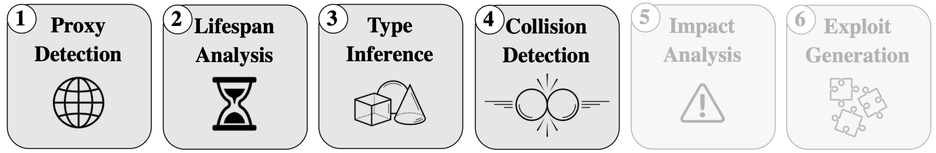
```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

MASK: **ff ff ff 00**

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

MASK: **00 00 00 ff**

CRUSH




```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

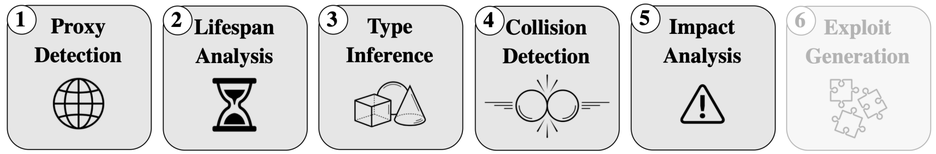
MASK: **ff ff ff 00**

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

MASK: **00 00 00 ff**

- Detected collision at slot 0x0

CRUSH




```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

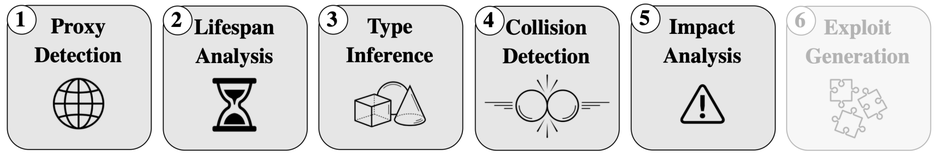
MASK: **ff ff ff 00**

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

MASK: **00 00 00 ff**

- Detected collision at slot 0x0

CRUSH




```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
<sink>
```

MASK: **ff ff ff ff**

```
v2 = <source>
SSTORE v2, 0x0
```

MASK: **ff ff ff ff**


```
v1 = SLOAD 0x0
v2 = SHR v1, 0x8
<sink>
```

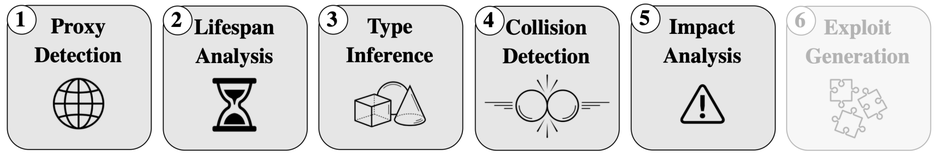
MASK: **ff ff ff 00**

```
v1 = <source>
v2 = AND v1, 0xff
SSTORE v2, 0x0
```

MASK: **00 00 00 ff**

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?

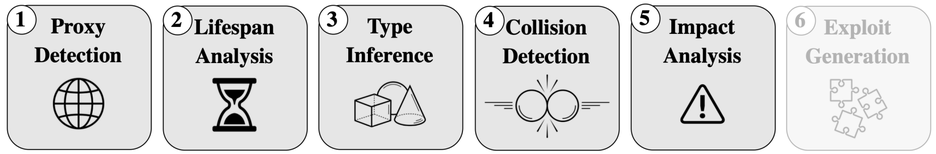
CRUSH




```
1  contract Proxy {
2      uint visits;        // storage slot [0x0]
3      address LOGIC;      // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract LogicA {
17     uint visits;        // storage slot [0x0]
18     [...]
19 }
20
21 contract LogicB {
22     bool initialized;   // storage slot [0x0]
23     address admin;      // storage slot [0x0]
24
25     function initialize() external {
26         require(!initialized);
27         initialized = 1;
28         admin = msg.sender;
29     }
30     function withdraw() external {
31         require(msg.sender == admin);
32         payable(admin).transfer(this.balance);
33     }
34 }
```

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?

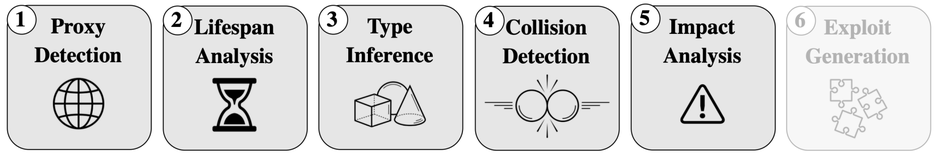
CRUSH




```
1  contract Proxy {
2      uint visits;        // storage slot [0x0]
3      address LOGIC;      // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract LogicA {
17     uint visits;        // storage slot [0x0]
18     [...]
19 }
20
21 contract LogicB {
22     bool initialized;   // storage slot [0x0]
23     address admin;      // storage slot [0x0]
24
25     function initialize() external {
26         require(!initialized);
27         initialized = 1;
28         admin = msg.sender;
29     }
30     function withdraw() external {
31         require(msg.sender == admin);
32         payable(admin).transfer(this.balance);
33     }
34 }
```

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB

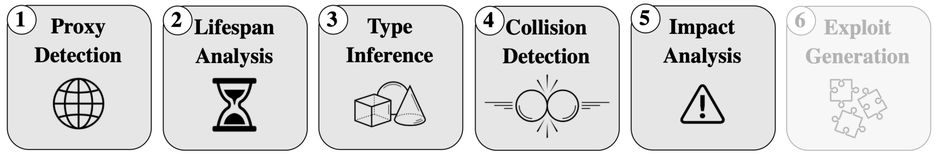
CRUSH




```
1  contract Proxy {
2      uint visits;        // storage slot [0x0]
3      address LOGIC;      // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract LogicA {
17     uint visits;        // storage slot [0x0]
18     [...]
19 }
20
21 contract LogicB {
22     bool initialized;   // storage slot [0x0]
23     address admin;      // storage slot [0x0]
24
25     function initialize() external {
26         require(!initialized);
27         initialized = 1;
28         admin = msg.sender;
29     }
30     function withdraw() external {
31         require(msg.sender == admin);
32         payable(admin).transfer(this.balance);
33     }
34 }
```

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB
- Is the slot **guarding** any sensitive operation?

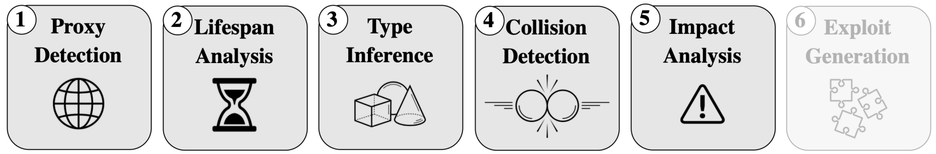
CRUSH




```
1  contract Proxy {
2      uint visits;        // storage slot [0x0]
3      address LOGIC;      // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract LogicA {
17     uint visits;        // storage slot [0x0]
18     [...]
19 }
20
21 contract LogicB {
22     bool initialized;   // storage slot [0x0]
23     address admin;      // storage slot [0x0]
24
25     function initialize() external {
26         require(!initialized);
27         initialized = 1;
28         admin = msg.sender;
29     }
30     function withdraw() external {
31         require(msg.sender == admin);
32         payable(admin).transfer(this.balance);
33     }
34 }
```

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB
- Is the slot **guarding** any sensitive operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB

CRUSH

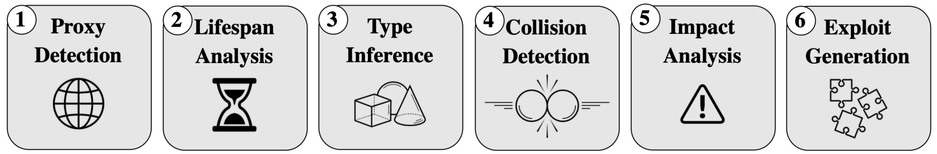


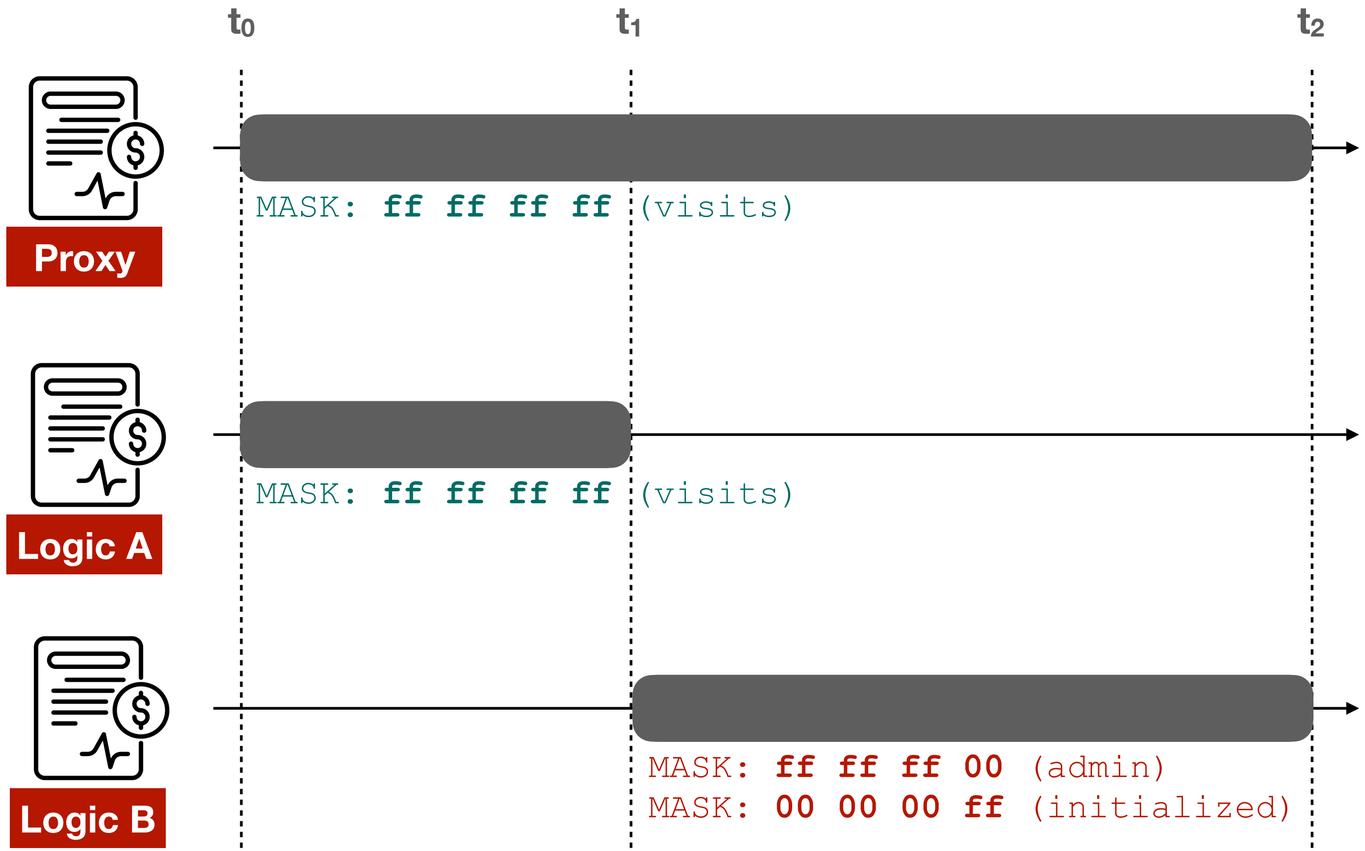

```
1  contract Proxy {
2      uint visits;        // storage slot [0x0]
3      address LOGIC;      // storage slot [...]
4
5      constructor() {
6          LOGIC = [...];
7          visits = 0;
8          LOGIC.initialize();
9      }
10     fallback() external {
11         visits += 1;
12         LOGIC.delegatecall(msg.data);
13     }
14 }
15
16 contract LogicA {
17     uint visits;        // storage slot [0x0]
18     [...]
19 }
20
21 contract LogicB {
22     bool initialized;   // storage slot [0x0]
23     address admin;      // storage slot [0x0]
24
25     function initialize() external {
26         require(!initialized);
27         initialized = 1;
28         admin = msg.sender;
29     }
30     function withdraw() external {
31         require(msg.sender == admin);
32         payable(admin).transfer(this.balance);
33     }
34 }
```

SOURCE CODE → EVM BYTECODE

- Detected collision at slot 0x0
- Is the slot used in any **sensitive** operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB
- Is the slot **guarding** any sensitive operation?
  - **NO** in Proxy/LogicA, **YES** in LogicB

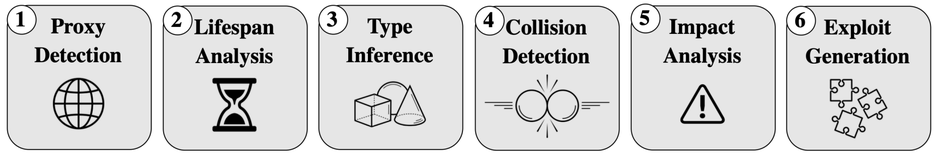
CRUSH

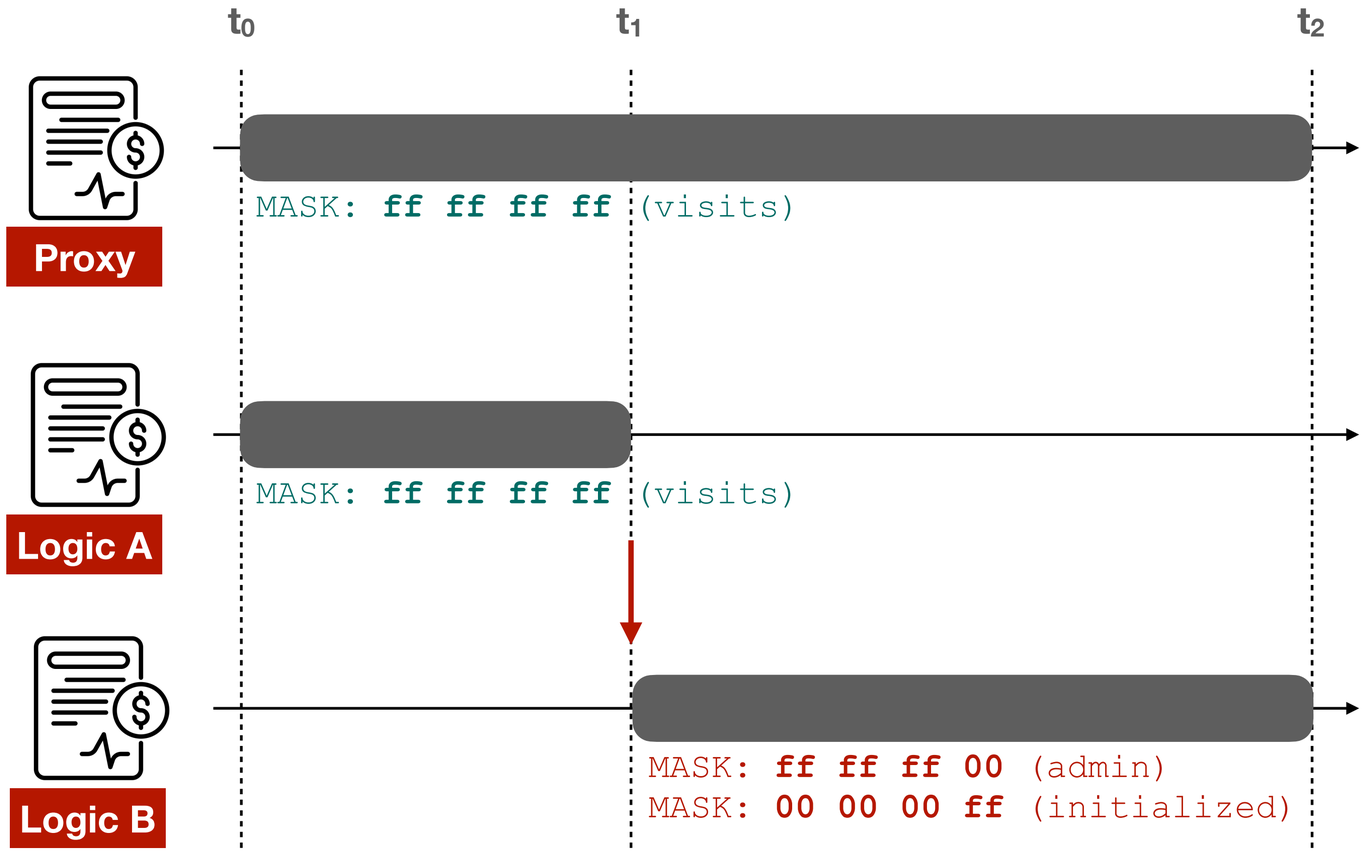




- Consider all possible attack scenarios for slot 0x0

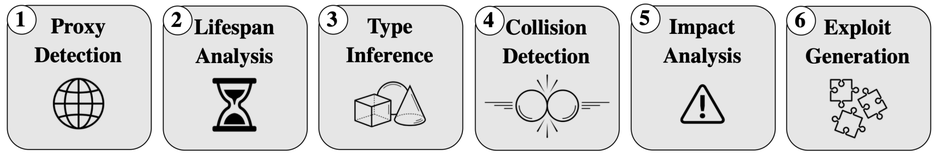
CRUSH

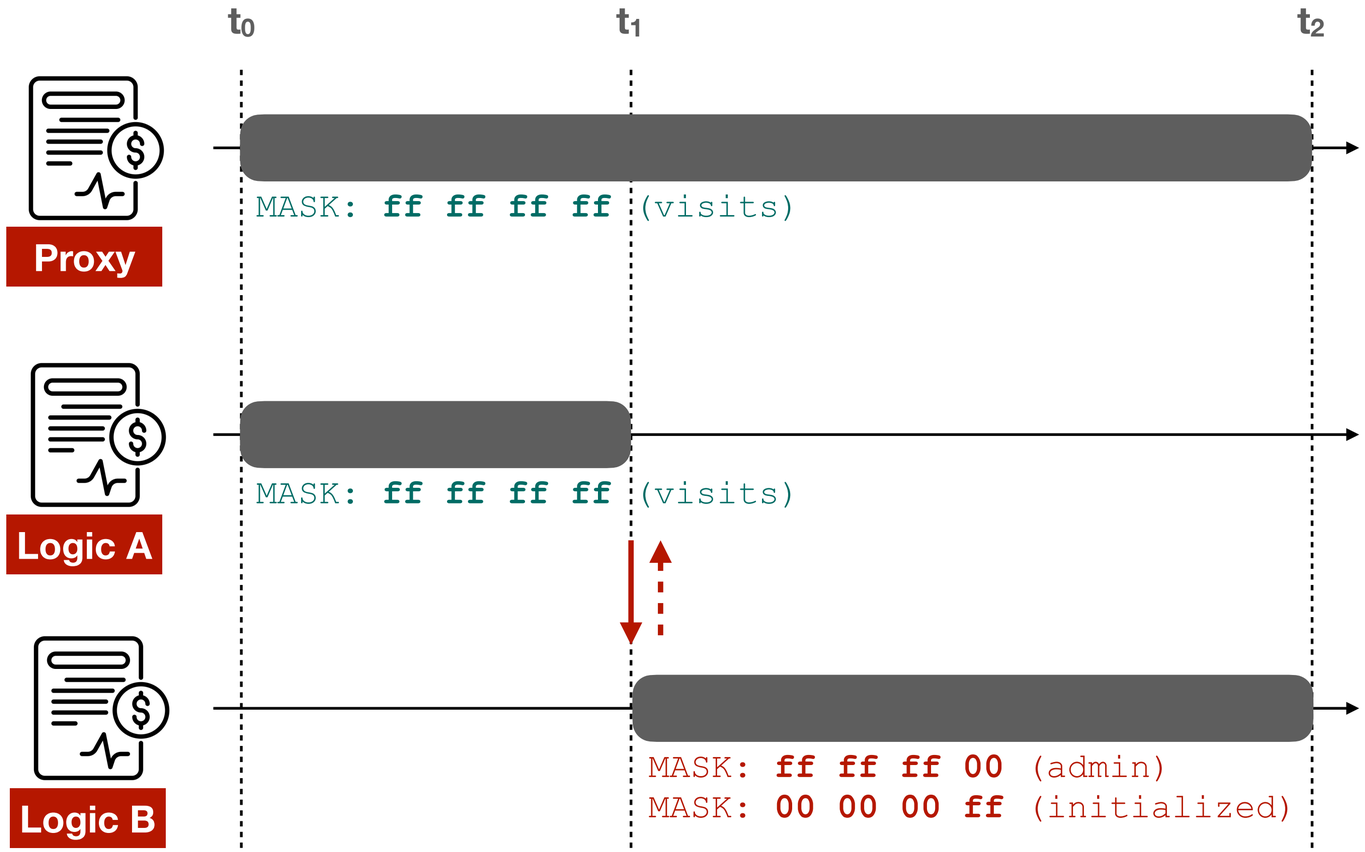




- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$

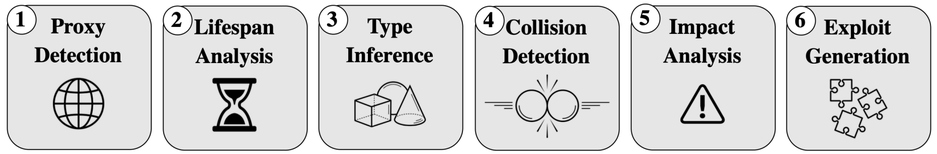
CRUSH

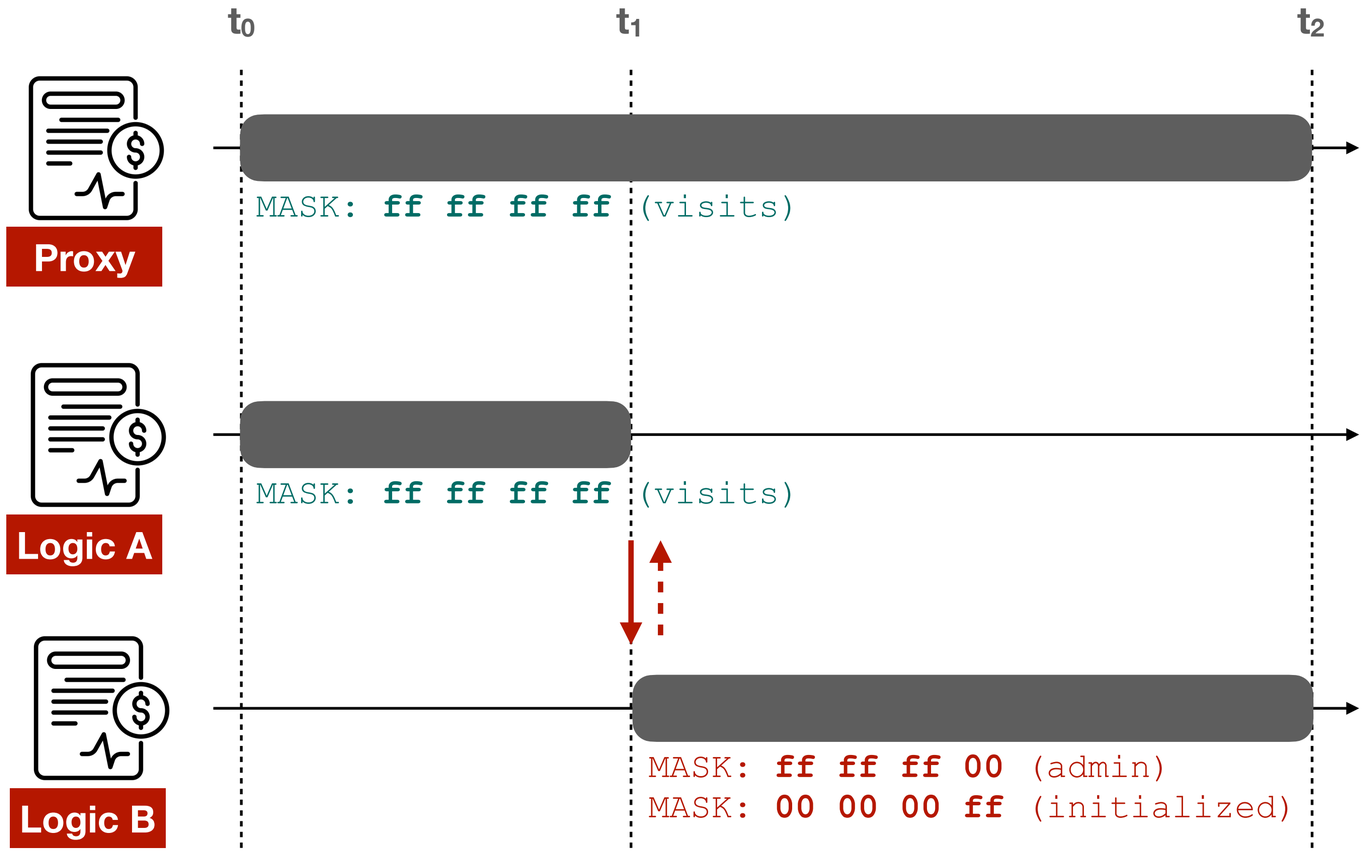




- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$

CRUSH

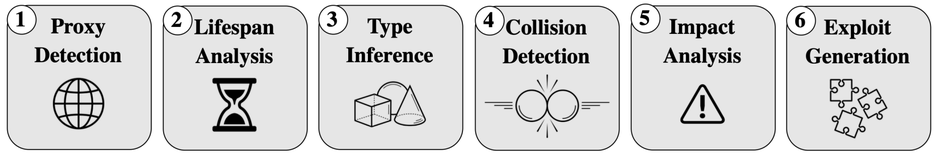


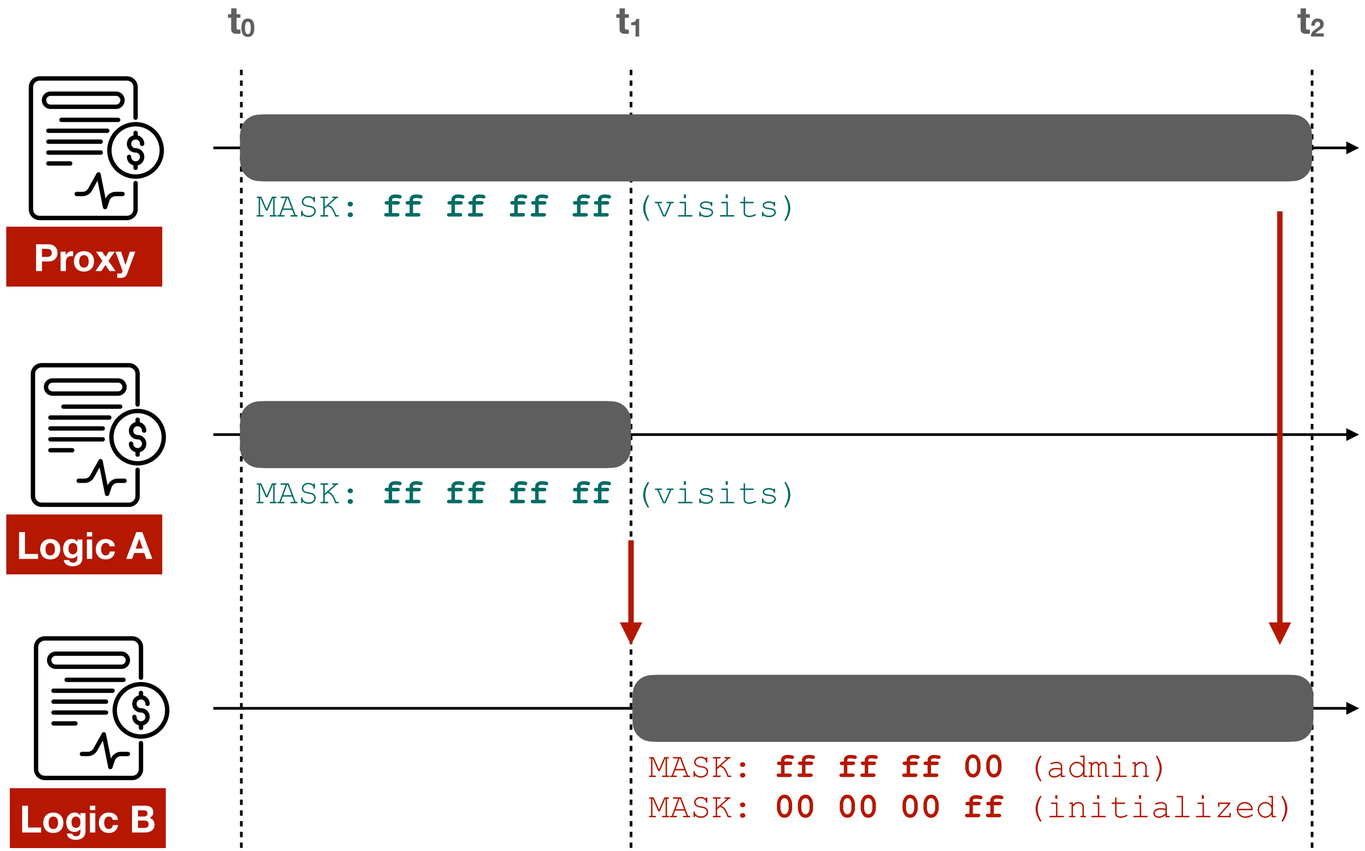


- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- × (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$

IMPOSSIBLE: LogicA is upgraded to LogicB and cannot read a value written by LogicB

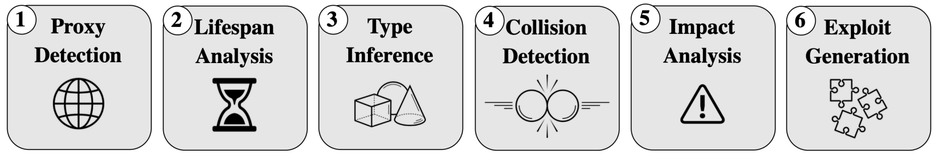
CRUSH

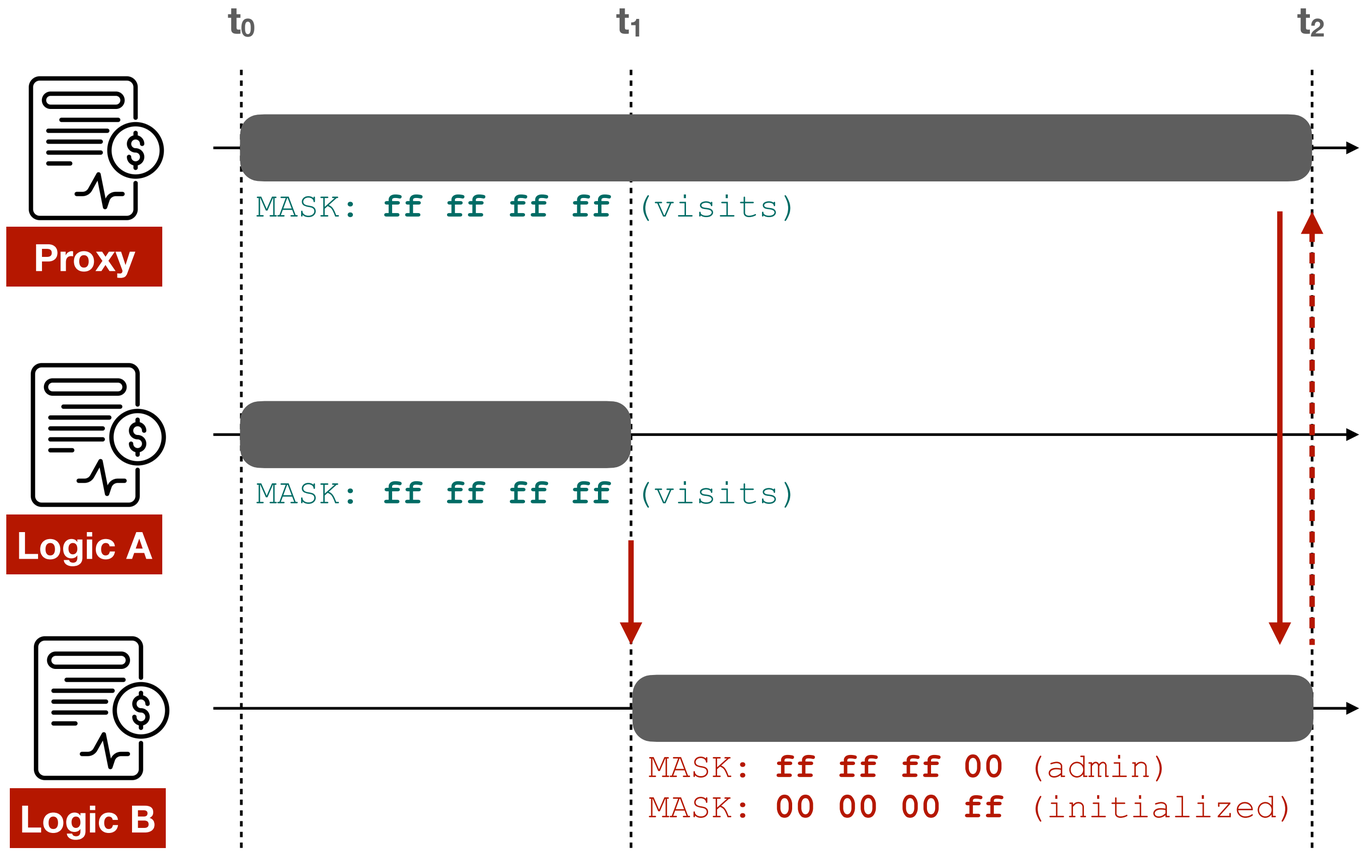




- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- × (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$
- (t_2) $Write_{Proxy} \rightarrow Read_{LogicB}$

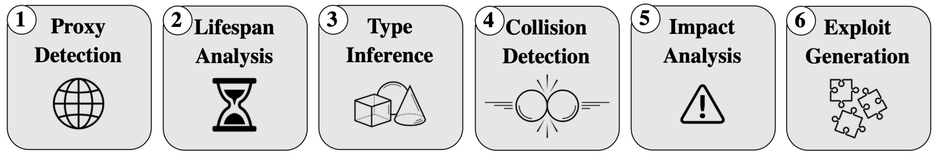
CRUSH

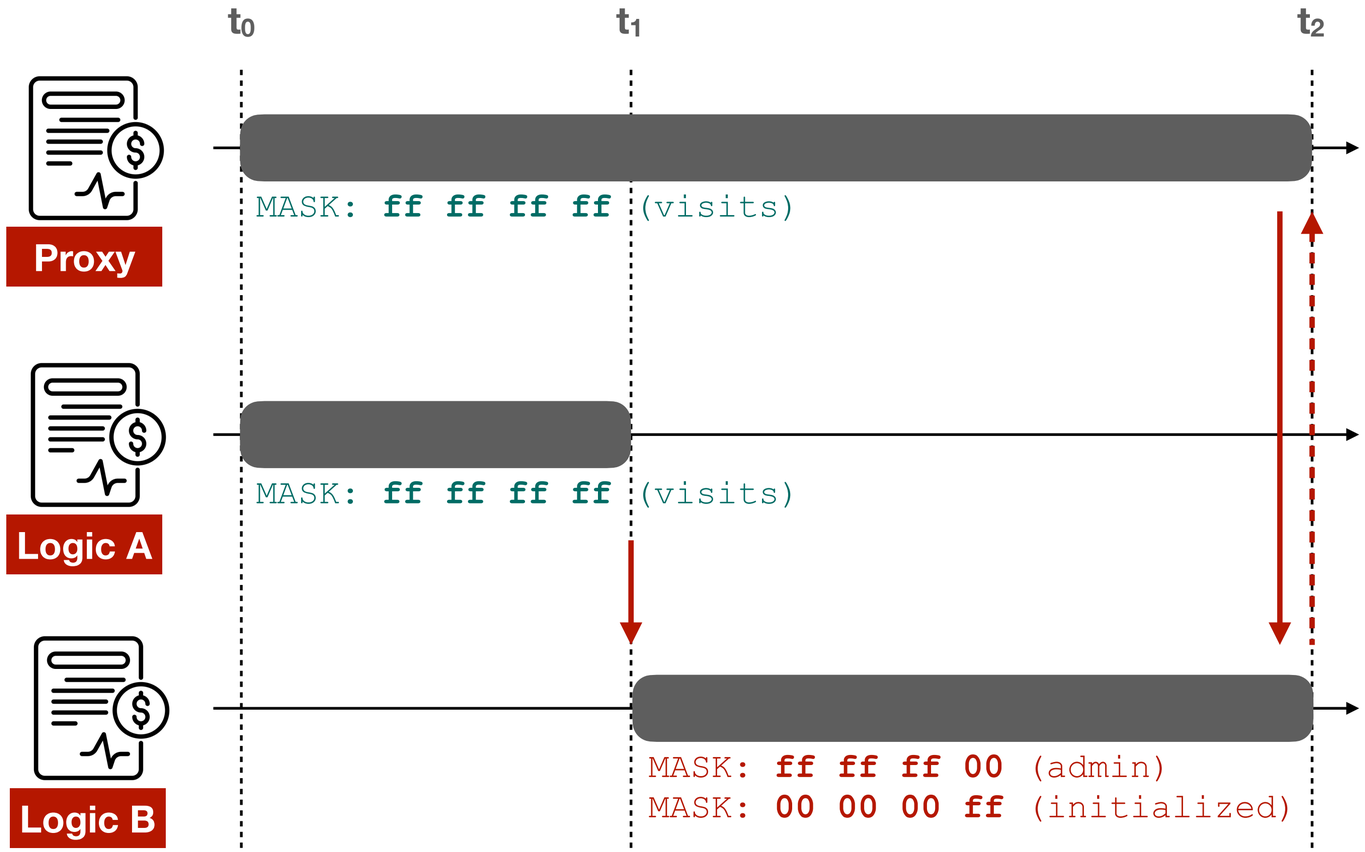




- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- × (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$
- (t_2) $Write_{Proxy} \rightarrow Read_{LogicB}$
- (t_2) $Write_{LogicB} \rightarrow Read_{Proxy}$

CRUSH

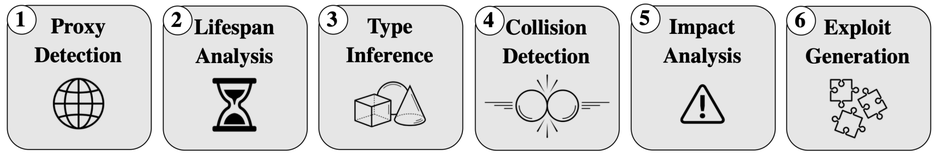


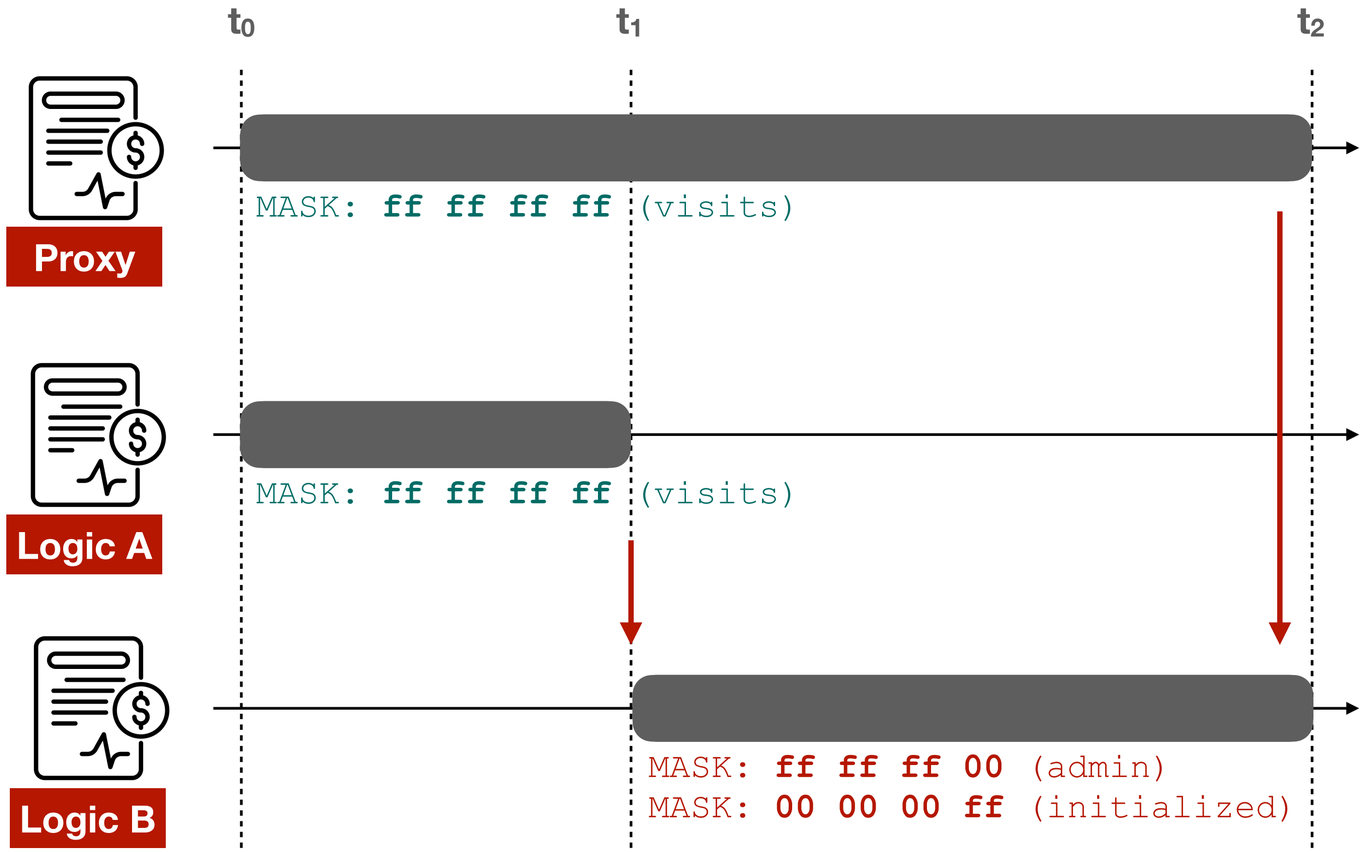


- Consider all possible attack scenarios for slot 0x0
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- × (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$
- (t_2) $Write_{Proxy} \rightarrow Read_{LogicB}$
- × (t_2) $Write_{LogicB} \rightarrow Read_{Proxy}$

NO IMPACT (uint visits)

CRUSH





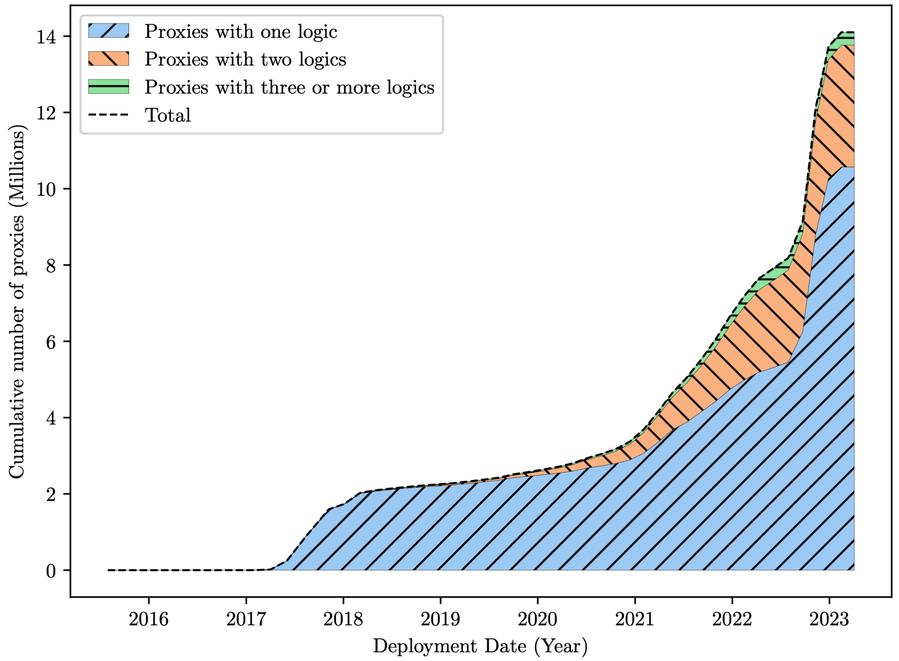
- Consider all possible attack scenarios for <u>slot 0x0</u>
- (t_1) $Write_{LogicA} \rightarrow Read_{LogicB}$
- × (t_1) $Write_{LogicB} \rightarrow Read_{LogicA}$
- (t_2) $Write_{Proxy} \rightarrow Read_{LogicB}$
- × (t_2) $Write_{LogicB} \rightarrow Read_{Proxy}$

Evaluation

- Found **14,891** (out of 14,237,696) contracts with storage collision
- Automatically synthesized an end-to-end exploit for **956** contracts
  - More than **$12M** of potential financial damage *($6M previously unreported)*
  - At least **132** still exploitable at the time of writing *($242K)*

Conclusions

- Storage collisions happen when two smart contracts have **conflicting interpretations of the same storage variables**
- Vulnerabilities in contract interactions are under-studied and **happening in the wild**
- We estimate more than **$12 millions** of possible financial damage



Thanks!

github.com/ucsb-seclab/crush

github.com/ucsb-seclab/greed

Check out our code

UC **SANTA BARBARA**

SEC LAB