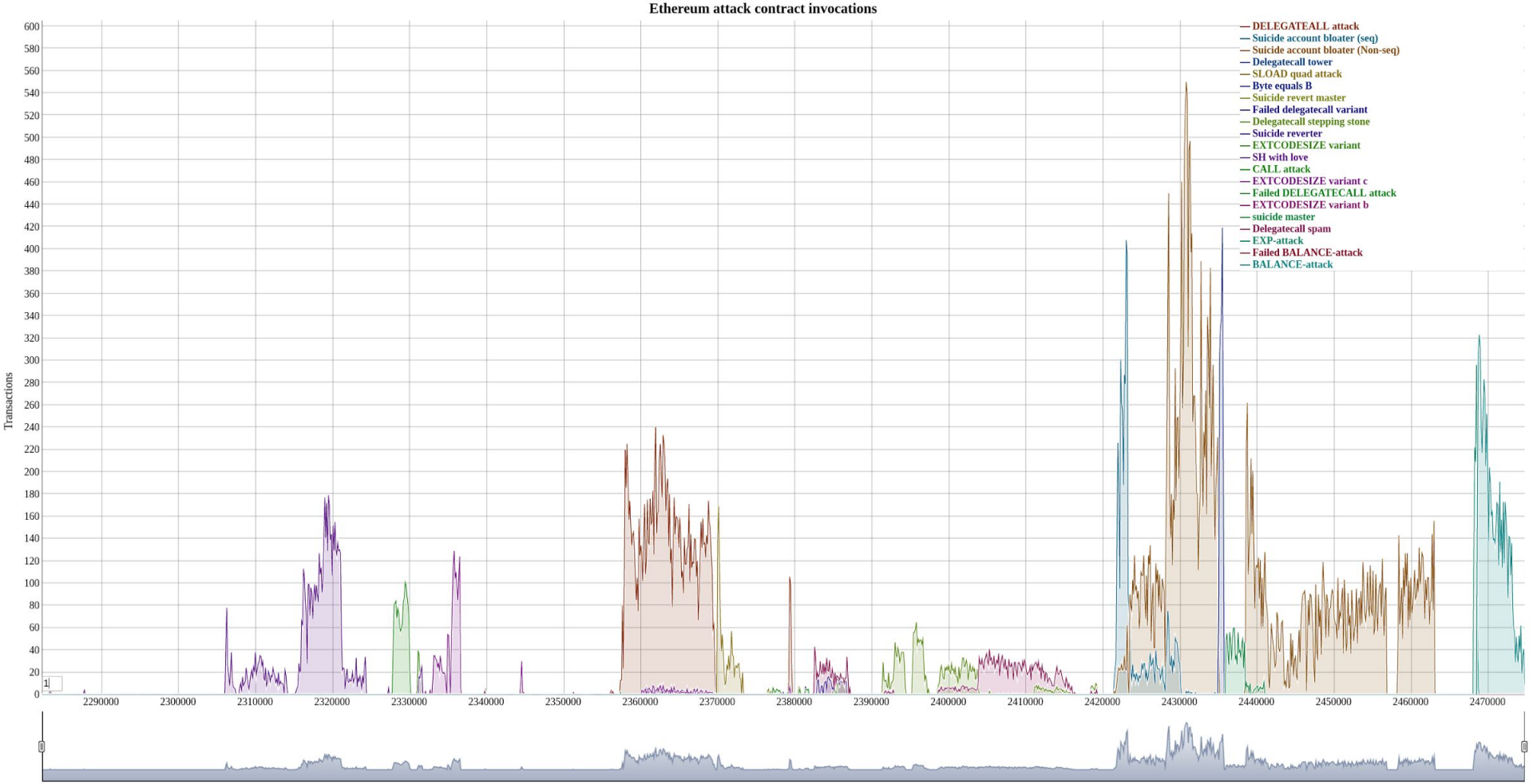
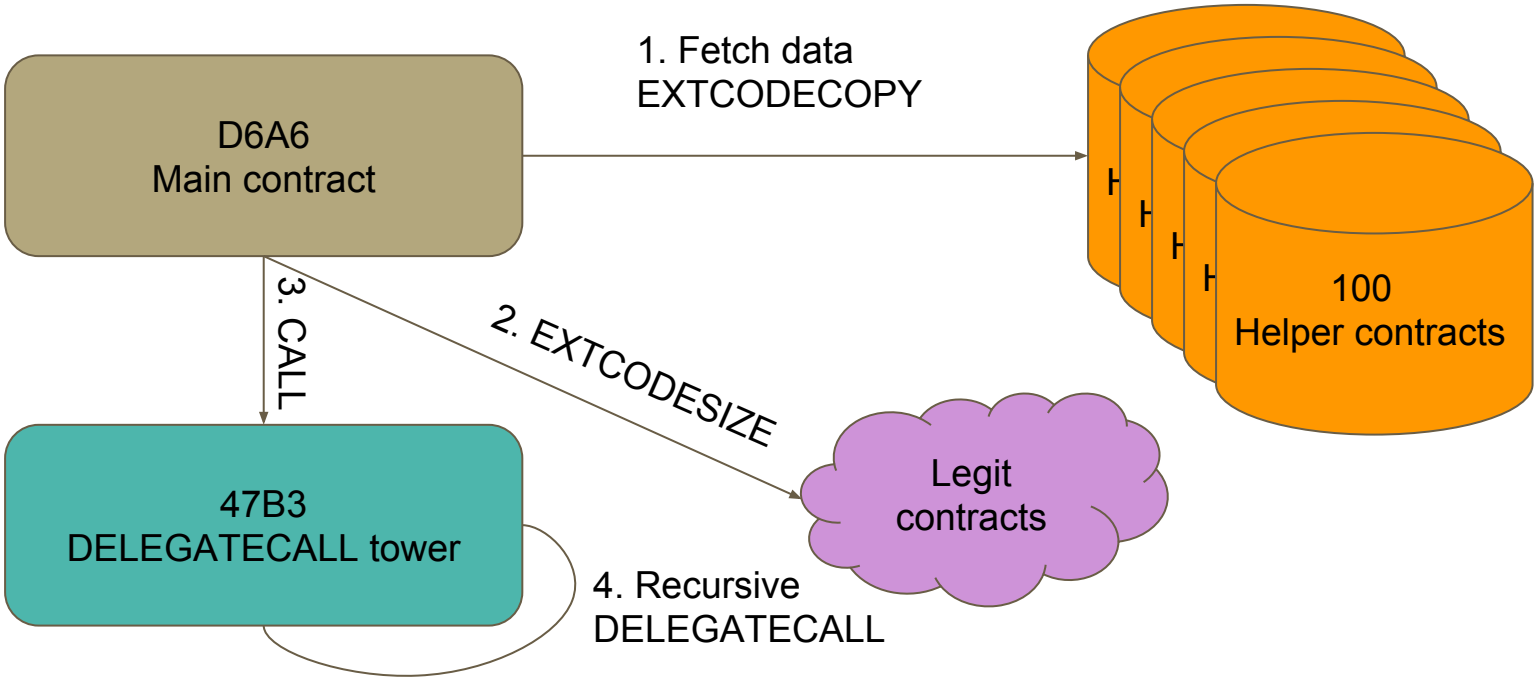
The Shanghai Attacks

From a technical perspective

Martin Holst Swende - EDCON 2017


Ethereum attack contract invocations
Transactions
DELEGATEALL attack
Suicide account bloater (seq)
Suicide account bloater (Non-seq)
Delegatecall tower
SLOAD quad attack
Byte equals B
Suicide revert master
Failed delegatecall variant
Delegatecall stepping stone
Suicide reverter
EXTCODESIZE variant
SH with love
CALL attack
EXTCODESIZE variant c
Failed DELEGATECALL attack
EXTCODESIZE variant b
suicide master
Delegatecall spam
EXP-attack
Failed BALANCE-attack
BALANCE-attack
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
520
540
560
580
600
2290000
2300000
2310000
2320000
2330000
2340000
2350000
2360000
2370000
2380000
2390000
2400000
2410000
2420000
2430000
2440000
2450000
2460000
2470000

September 18: From shanghai with love



Helpers

```
191c7da3941d55030ccf1738794ca36b4bf7f096 191ca3d1d47889dbc815e21509473400283cf875
592b557898080648e0ab45fbc147a789627be8a3 824804539b13b9cc3e4961ee8a45b71266aa6edd
7add626ef7c48df8debca5538d3d74a98ce354b1 51c6e9f300dfaf7ce3be314276c5f5bd6cda7452
14a6e26df1936e1166942a482138b83da74abe7f ed09d52418909d91398a809dfcf492a9915cec7f
199b55303d6bdd3e603abed3283b4c5ab01707a0 4124bdba90bc2dd4ceb4b3b620a0249a533deb77
d27f6c7fc9abe8c06d6d645957c3bc64fe5843ee 4c00c95155a1bcb15fee20b9180f8420cac4fdff
60152d8a48bfd55ee8a3c6af500c79cd2c63dd80 e2c9f65aad44367296cc3446bc64d67e276a695c
2b26d137119db89293d91d13c18ffc4aad2d9a0e e09e2af286e60766a3093d22436db48157d26e53
fddb712410853f75da945d4b60ead490988424a7 5d40fc74afdfb82e1460eb8abd25df3ecfef452c
```

XOR

```
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
fromshanghaiwithlove fromshanghaiwithlove
```

=

```
7f6e12cee775346d6ba776510e25d703279886f3 7f6eccbca710e8b5af7d837c7e2e406844538e10
3f593a15eb60672687c32492b62ed3e10e149ec6 e43a6b3ee87bd8a259210087fd2cc37a0ac518b8
1caf0d0384acec96b9d4c43afa5400c1e08c22d4 37b4869e73b7ce1284d6502b01ac81d500b50237
72d48d0082fb0f7f01fc4b215651cc55cb25c81a 8b7bba496bf8fcff5ee2e1f48b9de6c1fd339a1a
7fe93a5d4e03bc500752dfba5f523832dc7871c5 2756d2d7e3d44cbaa9dcd2df57c950f23f529d12
b40d0312bac389ae0a05053020aac80c9237358b 2a72a63c26c9dddf388641d06f66f048a6ab8b9a
066742e73bd7b4308fcba7c627650da5400cabe5 84bb9937de2c571cf1a4552fcb0da2164b051f39
4d54be5a62f5d9fcf4b17c7ab6e68822c142ec6b 86ec459ff58e6608c4615c4b3404c0e93bbd1836
9ba91e4963ed5e1bbdfc3c221783a0f8f4eb52c2 3b329319dcb7d94073088ae3ca4cab56a3803349
```

Amplification

0x47B3 : A recursive DELEGATECALL

```
# Stack: []
0x2    JUMP(:label0)

:label0
# Stack: []
0x4    PUSH(0x0)
0x6    DUP1
0x7    DUP1
0x8    DUP1
0xE    PUSH(DELEGATECALL(GAS() - 0x2B, ADDRESS(), POP(), POP(), POP(), POP()))
0xF    STOP()
```

Sidenote: About DELEGATECALL

The `DELEGATECALL` opcode can be thought of as borrowing code from another account. It means:

- I want to execute code at X, as if it was my code
  - within my own context and address

Whereas a CALL would execute within the callee-account, a DELEGATECALL executes within the caller-account.

Effect

- 100 x 512 contracts in memory
- Amplified by 1024

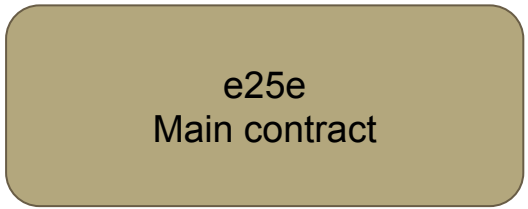
52 428 800 contracts in memory

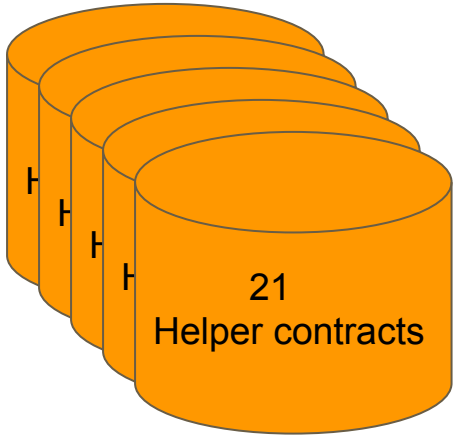
An attack against a client-specific caching mechanism

- Fix in 1.4.12 “From Shanghai, with love”
  - The new version only copied 'dirty' objects in the state cache

September 26: Variant #1

s/EXTCODESIZE/CALL





- Fetches data from helpers (21), XOR:s out addresses (21 * 256 = 5376)
- Performs a 0-value `CALL` to each one

Sidenote: About CALL

The `CALL` opcode is the mechanism used to transfer value in Ethereum, AND to invoke contract execution.

```
contract y{
    function bazonk(){ }
}
contract x{
    function baz(address bar){
        bar.send(1);      // Uses the CALL opcode
        y(bar).bazonk(); // Also uses the CALL opcode
    }
}
```

Effect

- CALL flagged an object as 'dirty'
- When it neared the end of the run, the `CALL` would necessitate a copying of 5000 objects into the new `state` cache.
- Since the dirtyness of the state is increased (linearly), the state copying becomes worse(linearly).

- Fix in 1.4.13 "Into the Woods"
  - Various fixes to state handling, as well as shortcutting transfers of `0`-value, to prevent setting ‘dirty’ flag on those objects.

September 27: Hitting the IO

- Very simple construct
- Fetch code size of ‘random’ addresses
- Causes heavy IO
- On 1.45MGas
  - POP: 2
  - GAS: 2
  - EXTCODESIZE: 20
- 60K lookups

7a30
Main contract

```
:label0
0x3     POP(EXTCODESIZE(GAS()))
0x6     POP(EXTCODESIZE(GAS()))
...
0x13EF     JUMP(:label0)
```

- Fix in prerelease 1.4.14 "What else should we rewrite?”
  - Among other things, this contained a codesize cache.

October 3: SLOAD quad attack

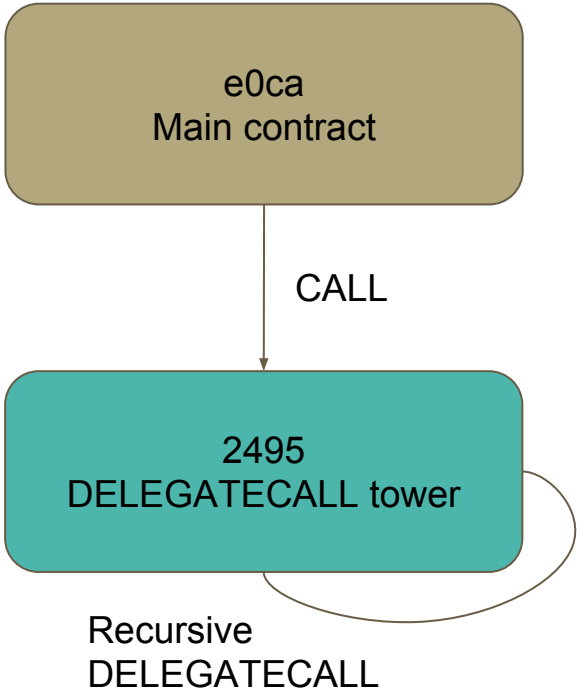
1. Setup section:

- While gas left, write '1' to next storage slot
- Update slot '0' with last slot
- Return

Called 457 times, filling 6754 slots

2. Execution section:

- Do 'SLOAD' on all storage slots 6,7K
- Call DELEGATECALL tower



Sidenote - What sources of data are there?

Storage: The persistent data-storage area where a contract can read/write data for later use. (**per-account**).

Memory: A temporary data area where, during execution, data can be placed. Memory is (**per-context**).

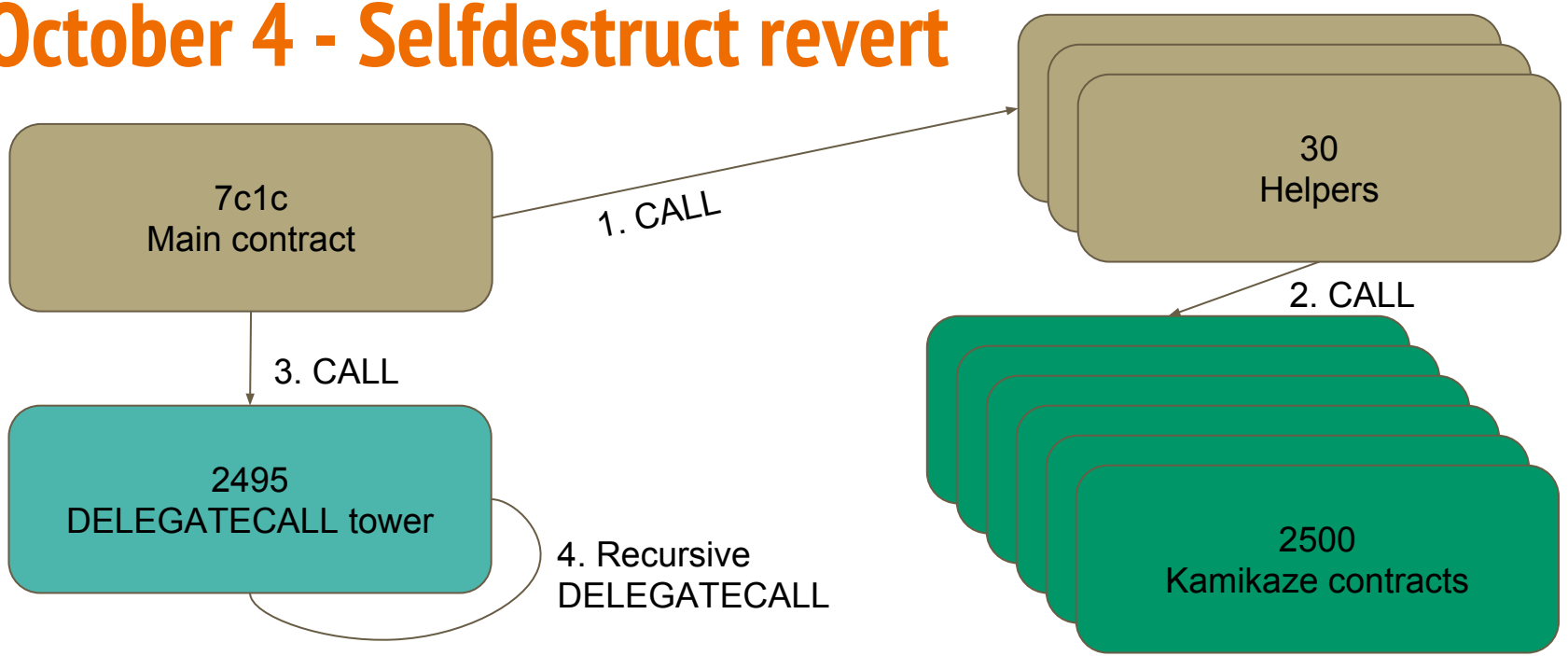
Other data-sources include:

- Calldata - data coming from the transaction (from the caller)
- Code - either own code (CODECOPY) or external (EXTCODECOPY)

Effect

- Very similar to “Shanghai with love” original attack
- Account storage was treated “as a whole”, causing a similar quadratic blowup of state cache during DELEGATECALL recursion
    - Caveat: 6K storage slots are takes less memory than 52M contracts. But quadratic effects are still bad
- Nodes at 100% CPU and 4G memory consumption
- Fix in 1.4.15 "Come at me Bro"
    - Track dirty state entries for each account object.

October 4 - Selfdestruct revert




```
0x1A    JUMPI(ADDRESS, !(0xD3E325.. == ORIGIN()))
0x30    SELFDESTRUCT(0x764D7849..)
```

Sidenote: SELFDESTRUCT

The SELFDESTRUCT opcode is a special snowflake

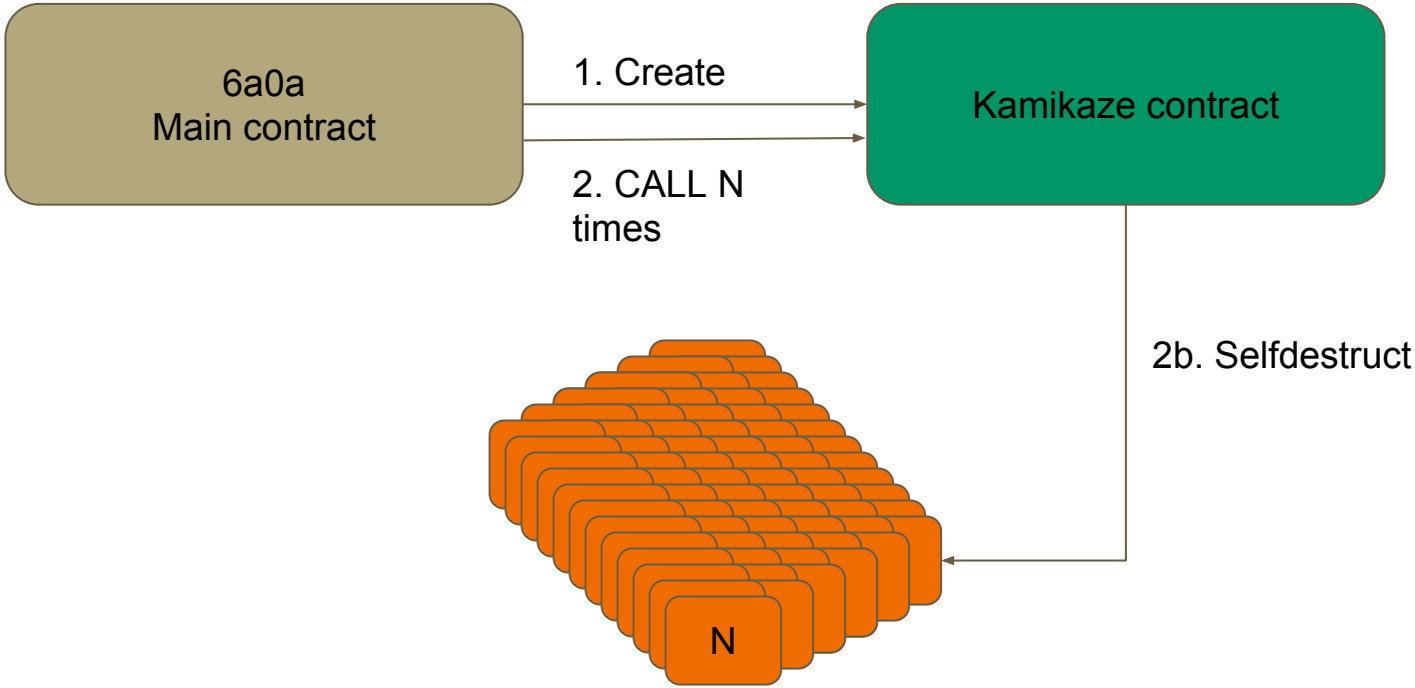
- An account is to be terminated, removing all state (code, storage) associated with the account.
- Very cheap, to incentivise clean-up of data
- Sends remaining funds to a beneficiary
- Terminates the current call
- Quirk: Can be called multiple times, even after selfdestruct has occurred
- … And all of this work is wasted/reverted in the case of OOG…

October 4 - Selfdestruct Revert (with a twist)

- Same as before, but also endowing each selfdestructor with 1 wei
  - (1 wei = smallest unit of ether)
- The attack(s) require quite a lot of set-up, in order to create the kamikaze contracts

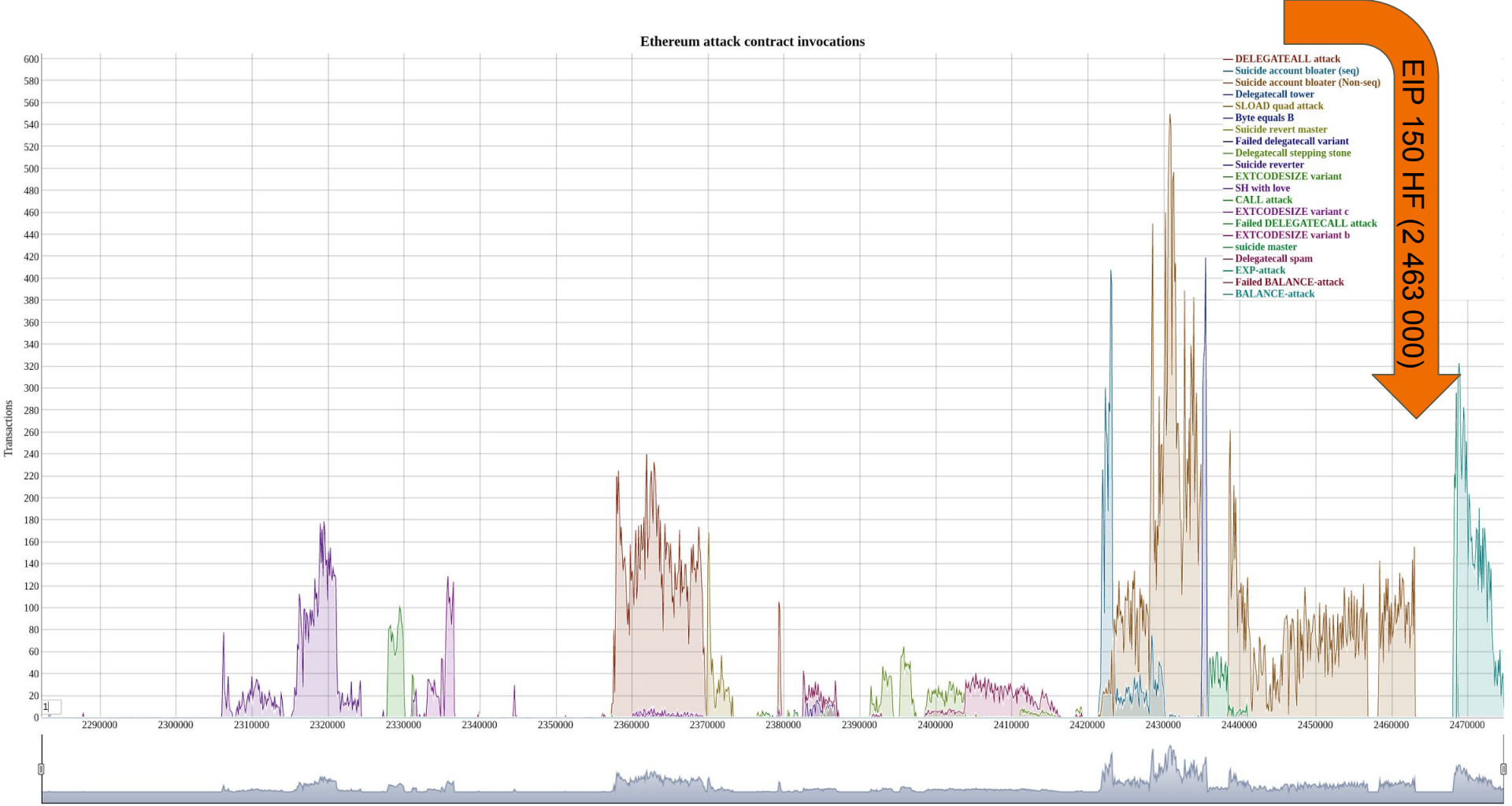
- Fix in 1.4.16 “Dear Diary” on October 6
  - Implemented state journaling, which makes state writing and reversion a linear operation.

October 11 : Kill-off and The Suicide State Bloat



Effects

- On October 13, the EIP150 HF was announced
- During the remaining time, the state bloat attack continued.
- Other attacks were also carried out:
  - EXP-attack
  - DELEGATECALL-spam
  - BALANCE-attack
- On October 18, EIP150 Rolled out at 2463000
- An estimated total of > 19M accounts were then created in the state
- Another HF (Spurious Dragon) facilitated cleanup of state
  - https://github.com/ethereum/statesweep


Ethereum attack contract invocations
Transactions
DELEGATEALL attack
Suicide account bloater (seq)
Suicide account bloater (Non-seq)
Delegatecall tower
SLOAD quad attack
Byte equals B
Suicide revert master
Failed delegatecall variant
Delegatecall stepping stone
Suicide reverter
EXTCODESIZE variant
SH with love
CALL attack
EXTCODESIZE variant c
Failed DELEGATECALL attack
EXTCODESIZE variant b
suicide master
Delegatecall spam
EXP-attack
Failed BALANCE-attack
BALANCE-attack
EIP 150 HF (2 463 000)
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400
420
440
460
480
500
520
540
560
580
600
2290000
2300000
2310000
2320000
2330000
2340000
2350000
2360000
2370000
2380000
2390000
2400000
2410000
2420000
2430000
2440000
2450000
2460000
2470000

END

Questions?

Martin Holst Swende
martin.swende@ethereum.org
Twitter: @mhswende
Github: holiman