The Shanghai Attacks
From a technical perspective

Martin Holst Swende - EDCON 2017
September 18: From shanghai with love

1. Fetch data
   EXTCODECOPY

2. EXTCODESIZE

3. CALL

4. Recursive
   DELEGATECALL

D6A6
Main contract

47B3
DELEGATECALL tower

100
Helper contracts

Legit
contracts
Amplification

0x47B3 : A recursive DELEGATECALL
Sidenote: About DEALGATECALL

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

- 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

---

Diagram:
- e0ca
  - Main contract
    - CALL
    - 2495
      - DELEGATECALL tower
      - Recursive
        - DELEGATECALL
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 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

7c1c
Main contract

2495
DELEGATECALL tower

3. CALL

1. CALL

30
Helpers

2. CALL

2500
Kamikaze contracts

4. Recursive
DELEGATECALL

0xA    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

6a0a
Main contract

1. Create

Kamikaze contract

2. CALL N times

2b. Selfdestruct
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](https://github.com/ethereum/statesweep)
END

Questions?

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