What Caused the Accidental Killing of the Parity Multisig Wallet & How to Detect Similar Bugs

Originally published by Bernhard Mueller on November 8th 2017

5,354 reads

4

@muellerberndt
Bernhard Mueller

Security Engineer

Bernhard Mueller, the creator of MythX, shows how to detect vulnerabilities in Ethereum smart contracts.

On November 6th, a user playing with the Parity multisig wallet library contract “accidentally” triggered its kill() function, effectively freezing the funds on all Parity multisig wallets linked to the library’s code. According to early estimates this might have made more than $100 million worth of Ether inaccessible (update: in the meantime, that number has gone up to $280 million).

As it happens, just a couple of weeks ago I wrote about using symbolic analysis to detect unprotected SUICIDE instructions. The newest Parity accident was caused by an instance of this vulnerability. In this blog article, I’ll examine what happened in the latest attack and introduce a new Mythril analysis module that can be used detect similar issues in Solidity code and on-chain contracts.

Ethereum rendition by Shed Designs for EventChain.io

First, let’s have a brief look at what happened. User devops199 (for the sake of simplicity, I’ll just call him “the attacker”) managed to cause havoc by sending only two transactions. The first transaction called the `initWallet` function to set the `owner` state variable to devops199’s address.

```
Function: initWallet(address[] _owners, uint256 _required, uint256
_daylimit)
MethodID: 0xe46dcfeb
[0]:0000000000000000000000000000000000000000000000000000000000000060
[1]:0000000000000000000000000000000000000000000000000000000000000000
[2]:0000000000000000000000000000000000000000000000000000000000000000
[3]:0000000000000000000000000000000000000000000000000000000000000001
[4]:000000000000000000000000ae7168deb525862f4fee37d987a971b385b96952
```

You might be wondering why this worked so easily. Weren't there checks in place to prevent random senders from calling `initWallet`? As you can see in the source code, access to this function is restricted with the `only_uninitialized` modifier:

```
/ throw unless the contract is not yet initialized.
 modifier only_uninitialized { if (m_numOwners > 0) throw; _; }
// constructor - just pass on the owner array to the multiowned and
 // the limit to daylimit
```

```
 function initWallet(address[] _owners, uint _required, uint _daylimit)
only_uninitialized
```

This modifier allows the wallet to be initialized once. Normally, when a new wallet is set up, this library function is called immediately during creation.

However, `WalletLibrary` itself was never to be supposed to be used as a *wallet*. It's only purpose was to provide code to other contracts calling it via DELEGATECALL. Therefore, and very unfortunately, the library contract instance itself had never been initialized. This was an oversight by the developers, and can be partly attributed to Ethereum's confusing library/CALL semantics. This topic would warrant its own blog post at some point, but for now let's focus on the particular attack.

Being the contract owner, the attacker now called the `kill` function in the second transaction:

```
Function: kill(address _to)
MethodID: 0xcbf0b0c0
[0]:0000000000000000000000000ae7168deb525862f4fee37d987a971b385b96952
```

This terminated the library contract, causing all wallets that depend on the library code to be unusable.

Detecting the Bug

Mythril's "unchecked suicide" module attempts to detect exposed kill functions like the one in `WalletLibrary`. The module is executed automatically when running mythril with the `-x` option.

To try it out, you can run the analysis against the Parity wallet. Unfortunately, Mythril's on-chain scanning feature can't be used on the dead contract instance, and the `WalletLibrary` code only compiles with solc 4.10. The best way to get it to work is to first compile the code in remix, then pass the runtime bytecode to Mythril with the `-c` option. You should get the following output:

```
./myth -x -c "60606040(…)0029"
=== Unchecked SUICIDE ===
```

```
Type: Warning
```

```
The function kill(address) executes the SUICIDE instruction.
```

```
The remaining Ether is sent to an address provided as a function
argument.
```

```
There is a check on storage index
keccac_14615016373309029182036848327162830196559325429 75_&_caller. This
storage index can be written to by calling the function
'initMultiowned(address[],uint256)'.
```

```
There is a check on storage index 0. This storage index can be written to
by calling the function 'initMultiowned(address[],uint256)'.
```

```
There is a check on storage index 1. This storage index can be written to
by calling the function 'initMultiowned(address[],uint256)'.
```

The algorithm works as follows:

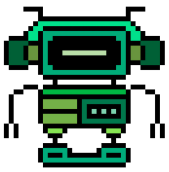
- Make an inventory of all SSTORE intructions that are not constrained by msg.sender (i.e. callable by anyone);
- Look for SUICIDE instructions and check the constraints on each.
- If a ‘tainted’ SSTORE for every relevant storage index found, attempt to solve the constraints on the basic block containing theSUICIDE block to make sure it is reachable.

It’s a pretty simple check, but sufficient to detect trivial flaws like the Parity one (although it isn’t false-positive-proof).

About Mythril and MythX

Mythril is a free and open-source smart contract security analyzer. It uses symbolic execution to detect a variety of security vulnerabilities.

MythX is a cloud-based smart contract security service that seamlessly integrates into smart contract development environments and build pipelines. It bundles multiple bleeding-edge security analysis processes into an easy-to-use API that allows anyone to create purpose-built smart contract security tools. MythX is compatible with Ethereum, Tron, Vechain, Quorum, Roostock and other EVM-based platforms.



by Bernhard Mueller @muellerberndt.
Security Engineer

Read my stories

BUILD WEB3 DAPP ON OASIS NETWORK

TAGS

#ethereum #security #hacking #reverse-engineering #ethereum-bug

RELATED STORIES

EVM-Puzzles: Learn Ethereum By Solving Interactive Puzzles

Published at Mar 28, 2022 by 0xkitsune #ethereum

Perp v2 Goes Open Source: Including Codebase, frontend SDK, and Subgraph

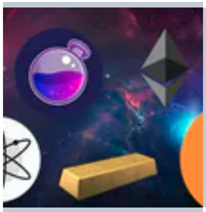
Published at Mar 28, 2022 by perpetualprotocol #defi

"My mission has always been about freedom," says Anthony Di Iorio, Ethereum Co-founder

Published at Mar 28, 2022 by scott-d.-clary #business

Stop Worrying About Uncertainty in Crypto Markets and Start Loving the Chaos with Coinwink

Published at Mar 28, 2022 by coinwink #cryptocurrency-investment



3 Passive Income Strategies For Earning Crypto on Cosmos and Terra Blockchains

Published at Mar 25, 2022 by cryptospace #defi-guide

THIS ARTICLE WAS FEATURED IN...

Ycombinator | Cryptobriefing | Samsclass | Steemit | Globalapptesting | Samsclass | Lab10 | Sc2tv