Empirical Study of Security Patches in Solidity: Known Smart Contract Weaknesses

Sungjae Hwang
sjhwang87@kaist.ac.kr
KAIST
South Korea

Sukyoung Ryu
sryu.cs@kaist.ac.kr
KAIST
South Korea

1 Solidity Smart Contract Weaknesses

In this section, we summarize the reported weaknesses of Solidity smart contracts.

1.1 A List of the Weaknesses

We identified the weaknesses from the following sources:

1. The Official Solidity Documentation: `https://solidity.readthedocs.io/en/v0.5.7/`
2. Smart Contract Weakness Classification (SWC): `https://smartcontractsecurity.github.io/SWC-registry/`
3. The Consensys Group: `https://consensys.github.io/smart-contract-best-practices/known_attacks/`
4. The NCC Group: `https://dasp.co`
5. 185 Academic Papers ("smart contract" in their titles and abstracts)

Table 1 shows the list. Each weakness is labeled with either `O` or `X`. `O` indicates that the Solidity compiler has either patched the weakness or announced to patch it in an upcoming release.
We further categorized the root cause of each weaknesses. The weaknesses fall into the "Blockchain" are due to the characteristics of Ethereum platform. The weaknesses fall into the "Solidity" are due to the vulnerable semantics of Solidity programming language itself. The weaknesses fall into the the "Developer" are due to the mistakes from developers.

Type	**Detail**	**Root Cause**	**Patch**
Denial of Service	DoS due to Gas Limit Excess	Blockchain	X
	DoS due to External Call Failures	Developer	X
	DoS due to Call Stack Overflow	Blockchain	X
Untrusted Interaction	Delegatecall to Unknown Code	Developer	X
	Reentrancy	Solidity	X
	Call to Unknown Fallback	Developer	X
Miner Dependency	Transaction Order Dependency	Blockchain	X
	Timestamp Dependency	Blockchain	X
	Random Value Dependency	Blockchain	X
Unintended Visibility	State Variable Without Visibility	Solidity	X
	Function Without Visibility	Solidity	O
Storage Corruption	Arbitrary Writes to Storage	Developer	X
	Uninitialized Storage Pointer	Solidity	O
Improper Signature Verificaiton	No Signature Verification	Developer	X
	No Protection Against Replay Attack	Developer	X
	Unauthorized Signature Modification	Developer	X
Typo Weakness	Typo of "=+" Operator	Solidity	O
	Typo in Constructor	Solidity	O
Insufficient Authentication	Careless Handling of `tx.origin`	Developer	X
	Unauthorized `SELFDESTRUCT`	Developer	X
	Unauthorized Ether Withdrawal	Developer	X
	Unauthorized Initialization Function	Developer	X
	Unauthorized Delegatecall	Developer	X
Inheritance Weakness	State Variable Shadowing Confusion	Solidity	O
	Inheritance Order Confusion	Solidity	O
Compiler Misusage	Floating Pragma Version	Developer	X
	Usage of Outdated Compiler Version	Developer	X
Property Violation	`require` Statement Violation	Developer	X
	`assert` Statement Violation	Developer	X
Platform Weakness	Private Information Leakage	Blockchain	X
	Contract Immutability	Blockchain	X
Careless Type Operations	Type Casting to Arbitrary Contract	Solidity	O
	Jump to Arbitrary Function Code	Solidity	X
Ether Loss	Inability to Transfer Ether	Developer	X
	Transfer to Orphan Address	Developer	X
Exception Weakness	Improper Exception Invocation	Solidity	X
	`call` Return Value Bypassing	Developer	X
Others	Usage of Deprecated Function	Solidity	O
	Arithmetic Overflow/Underflow	Developer	X
	Incorrect Contract Logic	Developer	X
	Failure to use Cryptography	Developer	X

Table 1: Weaknesses of Solidity Smart Contracts.

1.2 A List of the Solidity Weakness

The weaknesses fall into the "Blockchain" category cannot be handled by the Solidity team because they are not the problems of programming language. However, we believe that the following 11 weaknesses can be addressed by the Solidity team to improve the security of smart contracts.

ID	Name	Patch
ID-01	Function Without Visibility	Patched in 0.5.0
ID-02	Uninitialized Storage Pointer	Patched in 0.5.0
ID-03	Typo of "+=" Operator	Patched in 0.5.0
ID-04	Typo in Constructor	Patched in 0.5.0
ID-05	State Variable Shadowing Confusion	Planned to be Patched
ID-06	Inheritance Order Confusion	Patched in 0.4.24 and 0.5.0
ID-07	Type Casting to Arbitrary Contract	Patched in 0.4.0 and 0.5.0
ID-08	Usage of Deprecated Function	Patched in 0.5.0
ID-09	Reentrancy	Not Patched
ID-10	State Variable Without Visibility	Not Patched
ID-11	Improper Exception Invocation	Not Patched
ID-12	Jump to Arbitrary Function Code	Not Patched

Table 2: Solidity Weaknesses of Smart Contract.

As explained in our ICSE paper, the patches for ID-04, and ID-07 are not sufficient because the patches were developed without considering the mistakes patterns of developers. Moreover, ID-02 can be abused by attackers by not adopting security patches intentionally to develop benign-looking malware. Finally, the patches are required for ID-05, ID-09 to ID-12.

1.3 Brief Description of Weaknesses

a) **Function Without Visibility:** Functions unintentionally open to the external attacker due to the default function visibility that is "public"

b) **Uninitialized Storage Pointer:** Storage variables may be corrupted as uninitialized storage pointer alters the 0-index of storage variable

c) **Typo of "+=" Operator:** Unexpected behavior because "=+" operator is used for arithmetic addition

d) **Typo in Constructor:** The constructor is open to the external attacker as its name differs to contract name

e) **State Variable Shadowing Confusion:** Unexpected behavior because the same name of storage variables exist in both parent and child contract.

f) **Inheritance Order Confusion:** Misleading because of errors in Solidity specification regarding the concept of multiple inheritances

g) **Type Casting to Arbitrary Contract:** Unexpected behavior because the down-casting or unrelated-casting is possible.

h) **Usage of Deprecated Function:** Vulnerable because of Using deprecated functions

i) **Reentrancy:** Unexpected behavior because of unintuitive semantics of reentrance to the functions.

j) **State Variable Without Visibility:** Misleading because of missing visibility of storage variables

k) **Improper Exception Invocation:** Misleading because the same exception is handled inconsistently depending on the instructions

l) **Jump to Arbitrary Function Code:** Unsafe because of no verification for in-line assembly code

Please contact the authors for more information about each of the smart contract weaknesses discussed in this article.