BBM461 SECURE PROGRAMMING Research Report – Spring 2021

Name : Muhammed Said Kaya
Student ID : 21627428
Report Subject: Ethereum Smart Contract Vulnerabilities

List of the reported papers:

1. Johannes Krupp and Christian Rossow, CISPA, Saarland University, Saarland Informatics Campus (2018). teether: Gnawing at Ethereum to Automatically Exploit Smart Contracts In 27th {USENIX} Security Symposium ({USENIX} Security 18).

2. Christof Ferreira Torres, Mathis Steichen, and Radu State, University of Luxembourg (2019). The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts In 28th {USENIX} Security Symposium ({USENIX} Security 19).

3. Daniel Perez and Ben Livshits, Imperial College London (2021). Smart Contract Vulnerabilities: Vulnerable Does not Imply Exploited In 30th {USENIX} Security Symposium ({USENIX} Security 21).

4. Michael Rodler, University of Duisburg-Essen; Wenting Li and Ghassan O. Karame, NEC Laboratories Europe; Lucas Davi, University of Duisburg-Essen (2021). EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts In 30th {USENIX} Security Symposium ({USENIX} Security 21).

5. Mengya Zhang, Xiaokuan Zhang, Yinqian Zhang, and Zhiqiang Lin, The Ohio State University (2020). TxSpector: Uncovering Attacks in Ethereum from Transactions In 29th {USENIX} Security Symposium ({USENIX} Security 20).

1. teether: Gnawing at Ethereum to Automatically Exploit Smart Contracts

Contrasted to Bitcoin, which provides Peer Peer cash transmission generated by Satoshi Nakamoto, the first example in this area, the Ethereum chain created by Vitalik Buterin, which is preceded by a large-scale Ethereum chain, facilitates the creation of decentralized apps in the field of Web 3.0 enabled by blockchain technology. These programs are written in programming languages like Solidity and Vyper and then compiled into bytecode. On the Ethereum chain, these byte codes are executed on ethereum virtual machines. Smart contracts are the name for these services. Assailants can see the potential for security flaws and financial attacks as a result of this situation.

In this paper, we will examine the weaknesses of smart contracts and how these vulnerabilities can be used by attackers and the automation application teEther, which allows any written smart contract to be tested for vulnerability.

Any transaction done through blockchain, a technology that relies on cryptography and distributed computing, is kept in a peer-to-peer network, linking the hash values of the blocks to ensure they cannot be altered or modified by attackers. A structure that provides this is the consensus protocols used on the entire network. Through these protocols, the reliability and, once again, the irreversibility of the transactions made on the network, provided that they do not dominate 50 + 1 percent of the network, are ensured by rules, which are the expanded version of the consensus protocol for the Ethereum network, which is the second chain after Bitcoin. These rules are defined in the Turing-complete language. This means a kind of smart contract. However, the fact that smart contracts deployed on the Ethereum chain cannot be changed and updated causes financial losses and attacks. Now let's examine the working structure of smart contracts, EVM, and Ethereum in the Ethereum chain that causes these vulnerabilities.

The cryptocurrency of the Ethereum chain is Ether. The transaction made by users in this network to call an Ether transfer or a function of any smart contract is called a transaction. Those who approve these transactions by following the consensus protocol are called miners. This verification process has a certain fee, it is called a fee or gas. Smart contracts, in which users call their functions, are somehow accounts in the network. These accounts also have public addresses and balance values. If we need to get down to the structure of the smart contract, there is a private area called storage in this smart contract. What we mean by private is that the variables stored in this storage by smart contracts in the other network can neither be read nor changed. However, the fact that everything provided by blockchain technology is public indicates that the values stored in the storage can be obtained in a cryptographically

encrypted form. The codes of the smart contract are executed through a stack called EVM. This virtual machine includes more than 70 arithmetic, control, and some low-level instructions specific to the blockchain. Some of these instructions allow changing the contents of private storage. Apart from the permanent structure of the storage, there is another area called memory where data is always volatile. This memory area is reset as soon as the function of any smart contract is called. Let's come to the weaknesses of Smart Contract.

The codes of the smart contract are executed through a stack called EVM. This virtual machine includes more than 70 arithmetic, control, and some low-level instructions specific to the blockchain. Some of these instructions allow changing the contents of private storage. Apart from the permanent structure of the storage, there is another area called memory where data is always volatile. This memory area is reset as soon as the function of any smart contract is called. Let's come to the weaknesses of Smart Contract.

In smart contracts, there are structures called modifiers that provide who can call any function. In other words, if this authorization mechanism cannot be set up correctly, attackers can steal the Ether of the smart contract. In order to understand this structure better, it is necessary to examine the EVM bytecode instructions.

Two of the EVM instructions allow the transfer of Ether from a specific address to a different address. These are the CALL and SELF-DESTRUCT instructions. To start with CALL; The stack structure of this function includes fields such as gas, to, value, and offset. Here, the attacker's job may be to change the "to" value and put his own account. The SELF-DESTRUCT function enables the termination of a smart contract. In other words, the functions of this smart contract can no longer be called in any way. Here, the weak part of this function is that it transfers the token or Ether values of the smart contract to an address via the parameter it has received. If the attacker can call this function, he will have all the assets of that smart contract with his own address.

There are 2 more functions from the EVM instructions that enable Code Injection rather than direct Ether transfer. These are CALLCODE and DELEGATECALL. To start with CALLCODE, functions of a different smart contract in the network can be called through this function. If there is a "to" value here, this situation of the function may lead to vulnerabilities. The DELEGATECALL function, on the other hand, transfers the values of the first created transaction such as sender and value directly to the second function, rather than creating a new transaction. If the attacker can manage the "to" value, he can inject any code he wants into the contract. To find out the weaknesses of your smart contract, you can control these vulnerabilities with the automation application named teEther.

2. The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts

Ethereum, one of the advanced chains of the Blockchain, can run any code through smart contracts. These smart contracts can cause the blockchain to be analyzed and attacked by attackers due to the fact that the blockchain cannot be changed and updated. As an example of these, the smart contract named DAO, which took place in 2016 was subjected to a recursion attack by an attacker and caused a large-scale cryptocurrency to be seized in the system. However, with the emergence of these attacks and the fact that the developers in the ecosystem write more reliable code, and there are millions of smart contracts in the ethereum network, the attackers are now waiting for their victims by uploading the smart contracts that are written as if they have weaknesses, rather than analyzing smart contracts and carrying out attacks. These smart contracts are also called honeypots.

The purpose of this paper is to analyze honeypots thoroughly and to examine the tool called HONEYBADGER, which provides the detection of honeypots with certain techniques. As a result of the analysis made through this tool, the honeypot has 690 honeypots in the system with its increasing popularity, and the attackers who make this honeypot make a profit of approximately 90 thousand dollars.

In contrast to other blockchains, such as Bitcoin, the main intent of Ethereum is not peer-to-peer money transfer. The aim is to build a decentralized computing system, which is accomplished by the use of a smart contract. Smart contracts on the blockchain allow codes to interact with the EVM, a stack-based Turing machine. The architecture of this computer contains over 100 logic, flow, and blockchain-specific instructions. The person calling this instruction receives a payment called a "gas" for each instruction that happens on this EVM. The smart contract is unchangeable in this case. As a result of this scenario, programs that have not been carefully considered from a security perspective and written without detecting their vulnerabilities may become irreversible targets of attackers once they are deployed on the chain. These smart contracts are converted to byte codes using the Solidity programming language, which is a high-level language that looks like a combination of JavaScript and C, and its parser.

Because of the growing interest in smart contracts, attackers have been searching for flaws in smart contracts in recent years, because the greater the surface area of applications, the more attacks on that area. To offer an example of previous assaults, in 2016, users were asked to deposit their money in a smart contract called Decentralized Autonomous Organization to create an application similar to a stock exchange. However, since the function called as a

consequence of the attack's feedback was exposed to recursion, it was able to move 3.6 million Ether from the smart contract's account to its own account.

As a result of these threats, people who work in the ecosystem and write code in this area have written more stable software from a security perspective over the years, and millions of smart contracts have been used in the scheme. Now it seems that the attackers have no vulnerabilities, but they do, because of testing smart contracts and launching attacks. Honeypot smart contracts are used to keep an eye out for their prey. Furthermore, smart contract vulnerability analysis methods such as teEther have opened a kind of door for attackers. Which smart contract has a flaw, how it's discovered, and how it's targeted What is performed, on the other hand, is a little more complicated. But what is done in honeypots is to pretend to be vulnerable and attack the attacker by a different attacker.

Honeypot is that aspires to lose money to such an unspecified client (abuser) in exchange for extra money from the client. The money allocated by the recipient, on the other hand, will be stuck, and then the honeypot maker (attacking player) may be able to collect these.

```
1 contract MultiplicatorX3 {
2   ...
3   function multiplicate(address adr) payable {
4     if (msg.value >= this.balance)
5       adr.transfer(this.balance+msg.value);
6   }
7 }
```

Figure 1: An example of honeypot

Level	Technique
Ethereum Virtual Machine	Balance Disorder
Solidity Compiler	Inheritance Disorder
	Skip Empty String Literal
	Type Deduction Overflow
	Uninitialised Struct
Etherscan Blockchain Explorer	Hidden State Update
	Hidden Transfer
	Straw Man Contract

Table 1: A taxonomy of honeypot techniques in Ethereum smart contracts.

What the attackers do here is to load a contract that seems to be weak in terms of money transfer. Then the person trying to explode this smart contract calls the function with a certain amount of money. But there is a pitfall here. Through this trap, the code never explodes, such as if the condition that is permanently false. Due to the fact that this condition was never fulfilled, the attacker was trying to explode the smart contract, in fact, he fell into a trap and lost the money he had sent to the contract. For example, in the code in Figure 1, no matter how much money the user sends, it means that the smart contract will receive the money-back together with the money, but not knowing how the system works here causes the attacker to lose money. The balance value of the smart contract is updated with the amount of ether sent via message before any function is called. Due to this situation, the balance value

will always be greater than the message value and this scenario will remain as an infeasible path, and in this case, the person trying to attack will be attacked himself.

8 methods are used to detect honeypots. As can be seen from Table 1, these are grouped under 3 main headings. These titles are EVM, Solidity Compiler, and Etherscan Blockchain Explorer. The tool named HoneyBadger can analyze these techniques and find whether smart contracts are honeypots with an accuracy of 87 percent.

3. Smart Contract Vulnerabilities: Vulnerable Does not Imply Exploited

Security flaws in smart contracts, notably those built for the Ethereum network, have sparked a lot of educational and professional attention in recent years. While much of the research has concentrated on identifying insecure contracts, the aim of this paper is to determine that most of these agreements have been abused.

Although the large sums at stock, just 1.98 percent of the nearly 24 thousand insecure codes identified by six recently published studies have been abused as from implementation. In this network, there is 3 million Ether which equals 600 million American Dollars in that time. These findings are explained by showing that the resources are focused on a limited code that cannot be abused in operation.

Regarding the security flaw of contracts, It is very hard to guess the number of exploitable contracts in reality.

Since public blockchains have unchanging and accessible data, it allows the chance to make research for these types of studies which are done by academicians. According to this study, researchers investigated the reported smart contracts which include malicious codes at networks of Etherium. This study focused on 6 different studies which contain data about vulnerabilities and malicious codes.

In addition to the old 6 studies, this research team made contributions which are datalog formulation, evaluation of exploitation, and proposed explanation. To explain contributions of the researcher in this study in order;

Datalog formulation is an analysis method for Ethereum Virtual Machine (EVM) which is a stack-based machine that runs smart contracts.

Experimental evaluation of exploitation. Researchers realized that there is less money at the exploited contracts than they expected.

Proposed explanation: According to the hypothesis of this study, a big majority of the Ether is found in fewer amount of contracts.

Before the reasons for vulnerabilities of contracts, we should emphasize that the infrastructure of the Ethereum network must be understood. Ethereum is a second-generation blockchain technology after bitcoin. Differences between bitcoin and ethereum, ethereum provides non-centralized codes which can be executed by peer to peer. These codes are called smart contracts. Smart contracts are written by Solidity programming languages that mix Javascript and C languages. These smart contracts are deployed to the network by fully notes.

We can list the types of vulnerabilities by inspiring from reports which were written for EVM-based smart contracts.

Re-Entrancy: When a smart contract calls up another contract or account, if the function of the contract is called up and the gas value of the contract is very high, the balance value of the smart contract can be transmitted to another smart contract. DAO's were exposed to this situation in 2016.

Unhandled Exception: There are some instructions in the solidity programming language. There are not any errors while these instructions were calling. For example, the send function is used to transfer ethereum of smart contract to an address. If the coder of the smart contract does not care about the feedback of the send function and continues to flow, there will be conflict although transactions fail.

Locked Ether: There is a balance value of Ethereum smart contracts in other words there is an address of smart contracts like a normal public address and these addresses have a certain level of Ether value. But in some circumstances by the smart contract, staking Ether can be provided to the system. Therefore, The ether currency might not be able to withdraw from the smart contract. In these circumstances, there is an instruction named SELF-DESTRUCT. By using this command, all balance values of the smart contract could be sent to the owner of the contract and be the end life of the contract. Also, there is one more vulnerable situation. If there is no fallback method which is works when exception situations and also the gas value is not clear, while calling any function, ether value could not be refunded.

Transaction Order Dependency: Transactions, which are done by users, are stored on block structure in blockchain technology. After mind operation and consensus mechanism, the decision block is added to the chain. If there are two or more transaction which calls same smart contract function on the block, the order of transactions is really important for data consistency. This might cause some security situations.

Integer Overflow: The programmer which writes the smart contract should know the difference between signed and unsigned words and their meanings. If the programmer does not look from this perspective, the Integer Overflow attack could happen. When a smart function is called, if the input is given signed, but the parameter is unsigned value, that only accepts positive numbers, in for loop or any other operations this situation causes infinite loop or the function that should not work may work. Also, this situation gives the attackers the opportunity for Denial Of Service attacks.

Unrestricted Action: There is a structure named Modifier for used restrictions on smart contracts. If the programmer that writes the smart contract forgets the authorization mechanisms and not using modifiers again this might cause problems and anybody call any

function. For example, all balance value sending functions should be called by the only owner so a modifier must be used.

Name	Vulnerabilities						Report month	Citation
	RE	UE	LE	TO	IO	UA		
Oyente	✓	✓		✓	✓		2016-10	[35]
ZEUS	✓	✓	✓	✓	✓		2018-02	[31]
Maian			✓			✓	2018-03	[39]
SmartCheck	✓	✓	✓		✓		2018-05	[48]
Securify	✓	✓	✓	✓		✓	2018-06	[51]
ContractFuzzer	✓	✓					2018-09	[30]
teEther						✓	2018-08	[32]
Vandal	✓	✓					2018-09	[15]
MadMax			✓		✓		2018-10	[24]

Figure 1: A summary of smart contract analysis tools presented in prior work.

Name	Contracts analyzed	Vulnerabilities found	Ether at stake at time of report
Oyente	19,366	7,527	1,287,032
Zeus	1,120	861	671,188
Maian	NA	2,691	15.59
Securify	29,694	9,185	724,306
MadMax	91,800	6,039	1,114,958
teEther	784,344	1,532	1.55

Figure 2: Summary of the contracts in our dataset.

In Figure 1, there are some tools that are used for smart contract analysis and the accuracy of them while detecting vulnerability types which are defined above. if in figure 2, while testing these tools the number of datasets are given.

In conclusion, by this paper, earlier detected vulnerability situations and Data Formulation analysis were done. We conclude that the amount of Ether staked in the system is included in a smart contract with fewer security weaknesses.

4. EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts

In this paper, for sudden versioning of smart contract that includes vulnerability problems the solution is named EVMPATCH. The byte codes are stored on Ethereum Virtual Machine are rewritten by EVMPATCH. The security problems were defined earlier on EVMPATCH. After analyzing the smart contract part, EVMPATCH hardens them and contributes by security perspective. The functions are smart contract works again same way so the original path is rewritten in another way by using the Solidity programming language.

End to end attacks that uses the vulnerabilities of smart contract had devastating consequences and because of this problem, the benefits of blockchain technology and smart contract technology are questioned. Blockchain technology provides the immutability of blocks by using SHA-256 Hash digests. So fixing the vulnerability problems on the smart contract is not easy. In this part suddenly fixing problems are really important. So versioning could fix the problems.

Blockchain Technology 2.0 provides the smart contracts that are working on EVM, Turing-complete stack-based machine. With smart contracts the third party situations like public key infrastructure used for authenticity between two parts. So anymore there is no need for a centralized third party. With smart contract codes, the rules can be written and the two parts can communicate easily. Insurance, Bank, or any other system like Swift can change by using the smart contract. But there is a peer-to-peer network and some smart contracts stores more Ether which is a cryptocurrency and might be the target for attackers because all pieces of information on the stored blockchain are public. (for public blockchains, not ERP situations). Programming mistakes or any low-level instructions vulnerabilities give opportunities to attackers.

With the Re entrancy attack in 2016, that is, as a result of calling another smart contract from a smart contract, Ether values in a smart contract named DAO were stolen. Such situations provoked people to write code from a security perspective. This situation has also caused the community to diverge on one issue. Such as whether the transactions of the Ethereum network that contain this attack should be rolled back or not. As a result, 2 chains were formed as a classic Ethereum and Ethereum as a result of a hard fork. In Ethereum Classic, transactions involving this attack still exist.

As for how the EVMPATCH application rewrites weak smart contracts, each smart contract is converted into byte codes with Solidity Compiler and stored in machines called EVMs. Here, every code, every instruction, has a 1-byte opcode. Arguments are stored in a structure called the stack. In other words, arguments and code structures are located in different address

blocks and are extensible. Here, the field called memory is used for temporary data. For the non-volatile, the field called storage is used.

Rewriting codes consisting of 1s and 0s was said to be a solution. It can also be used here to dynamically implement the event of complicated from a security perspective without changing the flow diagram of the program. There are 2 different approaches here. The first is dynamic, but this approach is not suitable for the Ethereum network. Because something that is deployed to Ethereum, that is, something that works, cannot be written again instantly. In addition, the machine where the codes are running is not suitable for this situation. Static rewriting can be used here. Performing a static analysis will be sufficient to correct the program in terms of information security.

Due to the immutable structure in which the blocks create hash values, there are problems in versioning the smart contracts loaded into the system. The most common solution to this is to upload a new version to a new address. However, in this case, the data and balance value kept in the previous smart contract will cause trouble. In this case, the suggestion of the programmers is that the data is kept in an external smart contract and each versioned smart contract calls the functions of that smart contract. But in this case, it will cause the gas value to increase due to calling functions between too many layers. So there is nothing common about solving this situation.

Aware of this situation, attackers consider it appropriate to attack smart contracts that contain high amounts of Ether value, ie cryptocurrencies. In this case, with EVMPATCH, which provides instant analysis and rewriting of smart contracts, vulnerabilities can be detected automatically and measures can be taken against attackers.

5. TxSpector: Uncovering Attacks in Ethereum from Transactions

In this paper, a tool named TXSPECTOR is used for detecting vulnerability on a transaction level. There are also many tools but they are analyzing the smart contract on bytecodes. Firstly need to explain the background of blockchain 2.0, Vitalik Buterin who is the creator of Ethereum brought new features according to Bitcoin that is the blockchain 1.0. One of the topics is decentralized computing means that codes named smart contracts work on every full node on the network and store the same storage. Despite the big magic, this technology gives new vulnerable circumstances for software. Because the data or smart contract deployed on the network is unchangeable and can not be updated, they are open for attacks. Then they cause lots of loss of cryptocurrencies to attackers. Also, the general rule as the surface of any software grows up, the attackers are more interested in that.

To do this, it transforms transactions into abstractions over time without sacrificing valuable details from the initial transactions. Second, by modifying the Selected Features, it could become appliable to evaluate transactions with different options, together with no malicious situations. The tool is also intended to be fast. While it is hard to detect the same attacks in real-time using the logic-driven paradigm, efforts were made to greatly minimize both storage and output overheads by using the tool to fund studies.

The tool's main purpose is to find damages from Blockchain technology transactions using the provided rules. The tool will expose smart contract bugs as a natural consequence of performing transactions, which is essentially running machine language fragments. As a result, Tool can intrusion detection upon this network over interactions, and also the insecure contracts associated with any of those operations.

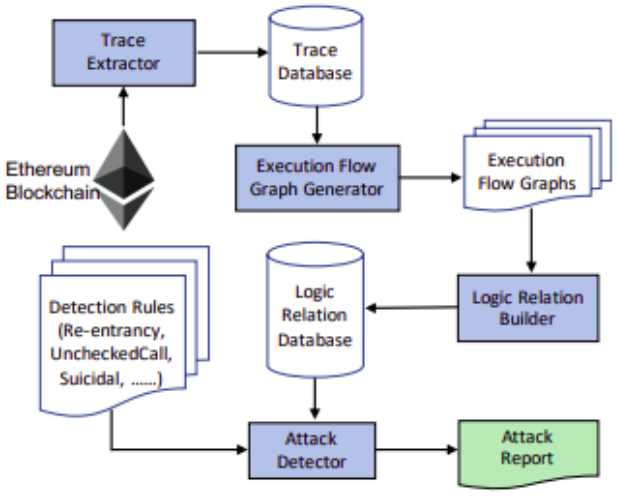

Figure 1: Components and the workflow of TXSPECTOR.

There are four parts of the tool.

The Execution Flow Graph Generator parses the bytecodes and generates Execution Flow Graphs. These are traversed by Logic Relation Builder, which extracts control and data relationships before expressing those as rationale relationships in the database. Lastly, Attack Detector makes queries the Logic Relation DB using customer rules then generates a full action record.

The Attack Detector is a critical part of the tool that receives customer inquiries as triggers and research activities upon that database created by the Logic Relation Builder to justify such a given asset's protection value. The database can be used for diverse forms of assessment when it is created; without the need to recreate a new record for each rule. The results aren't all yes or no responses to a single query; in place of, comprehensive information about the actions are given if they're found, allowing for more research.

Re-entrancy Attacks: When calling a function of a smart contract, another function can be called within the transaction. In this case, the Ether values in the smart contract can be stolen by calling the function of the first called smart contract within the second called function. It is the DAO attack that has happened over the years and has led to huge ethical debates. It happened in 2016 and caused a hard fork that would cause transactions to be rolled back, which would contradict the fixed structure of the Blockchain.

UncheckedCall Attacks: Instruction can work on EVM with CALL opcode in flowing contracts. There are some transfer functions. (like send). When these functions are called, if they do not encounter any error throwing situation, the developer should wait for the result of this function and continue the workflow according to the answer. Scenarios and situations that are forgotten here cause security vulnerabilities and financial losses.

Suicidal Attacks: The SELF-DESTRUCT function enables the termination of a smart contract. In other words, the functions of this smart contract can no longer be called in any way. Here, the weak part of this function is that it transfers the token or Ether values of the smart contract to an address via the parameter it has received. If the attacker can call this function, he will have all the assets of that smart contract with his own address. The tool named TXSPECTOR cares the transactions instead of contracts. Then unauthorized function calls as SELFDESCTRUCT can easily detectable.

In conclusion, The decentralized computing Blockchain 2.0 Technology While bringing back anonymity and libertarian approaches, the fact that the codes cannot be instantly patched causes the attackers to detect the error of the contracts and attack. After all, we have to see that anything with connectivity can be hacked.