Evil Chaincode: APT Attacks Based on Smart Contract

Zhenyu Li[1], Yujue Wang[1], Sheng Wen[2], and Yong Ding[1,2(✉)]

[1] School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
stone_dingy@126.com

[2] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China

Abstract. In this paper, we discuss methods of stealing data via advanced persistent threat (APT) attacks on blockchains. Blockchain technology is generally used for storing data and digital coins and counts more than 562 organizations among its users. Smart contracts, as a key part of blockchain technology, are used for blockchain programmability. APT attacks are usually launched by government-backed hackers to steal data. APT attacks build hidden Command and Control (C&C) channels to steal resources remotely. Smart contracts represent a vulnerability of blockchain technology to APT attacks because of their sandbox-style open execution environment. Therefore, we performed several attack experiments to test methods of abusing smart contracts, including the remote execution of commands, and the stealing of large amounts of data. These experiments demonstrated that APT attacks could be successfully executed on a blockchain platform. In the large-scale data-stealing experiments, we found that the transmission rate for a maximum target data size of 100 MB can reach 27.771 MB/s, faster than the average rate of approximately 100 kB/s of a three-layer network proxy. We also investigated APT attacks based on public APT events, which use hidden techniques to steal data as critical APT attack actions. We propose several attack algorithms that can be applied for APT attacks.

Keywords: Blockchain · Hyperledger Fabric · Smart contracts · APT

1 Introduction

A blockchain is essentially a distributed database; each participant keeps a complete copy of the database, or at least a record of a large number of recent

This document is supported in part by the National Natural Science Foundation of China under projects 61772150, 61862012, and 61962012, the Guangxi Key R&D Program under project AB17195025, the Guangxi Natural Science Foundation under grants 2018GXNSFDA281054, 2018GXNSFAA281232, 2019GXNSFFA245015, 2019GXNSFGA245004 and AD19245048, the National Cryptography Development Fund of China under project MMJJ20170217, and the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004.

G. Xu et al. (Eds.): FCS 2020, CCIS 1286, pp. 178–196, 2020.
https://doi.org/10.1007/978-981-15-9739-8_15

transactions [1]. With the development of blockchain, smart contract technology has emerged. Many large banks, internationally notable companies, and governments use blockchains to store data and use blockchain smart contracts to process business data [2].

Advanced persistent threat (APT) attacks are characterized by slow and small movements of a group of attackers to accomplish a particular goal, which is usually to steal data from a target without being caught. APT attackers might use familiar methods to break into their target entity's network, but the tools they utilize for penetration may not be familiar [3]. Large banks, internationally notable companies, and governments are often attacked by APT attackers because of the high value of their data [4].

Blockchain technology is still under development, and security researchers who study blockchain typically focus only on transaction security; however, this type of security is less relevant to APT attacks.

In this paper, we attempt to address the above problems by proposing three algorithms for APT attacks to provide a concrete illustration of the security threats faced by blockchain users, including the malicious chaincode installation, the remote execution of commands, and the stealing of large amounts of data.

In our related work, we have found that the smart contracts used in blockchain frameworks can run any code in the corresponding sandbox environment. The ability to run arbitrary code in the sandbox is extremely useful for facilitating APT attacks. We have found that APT attacks can take advantage of the unique features of smart contracts, including their native encryption for communication and their unrestricted sandbox environment. This makes smart-contract-based APT attacks on blockchain technology difficult to distinguish from legitimate activity. These novel features of smart-contract-based attacks, which can be used to hide such attacks more effectively, are therefore worthy of further study.

We design algorithms to demonstrate APT attacks based on smart contracts. Through these algorithms, we can abuse blockchain technology to quickly break the security-related restrictions of internal network area boundaries, rapidly expand the influence of an attack, and construct long-term communication channels to steal valuable data.

All experiments have succeeded in demonstrating our proposed attack techniques. In large-scale data transfer experiments, we have shown that the transmission rate for a maximum target data size of 100 MB can reach 27.771 MB/s, faster than the average rate of approximately 100 kB/s for an APT attack via a three-layer network proxy.

The contributions of this paper are summarized as follows: 1) We propose a means of launching hidden APT attacks against blockchains. 2) We demonstrate that the unrestricted smart contract environment of blockchain platforms is extremely dangerous. 3) We illustrate the security threats faced by blockchain users through experiments.

2 Related Works

2.1 Vulnerabilities and Attack Detection in Blockchain Systems

In a review of the related research, we have found that blockchain platforms have many vulnerabilities.

Yamashita et al. [5] reviewed the potential security risks posed by the smart contracts used in Hyperledger Fabric, a particular blockchain framework. These security risks can be divided into four types and 16 subtypes. There are four types of security risks caused by access from outside of the blockchain: web services, system command execution, external file access, and external library calls. Furthermore, that paper proposed a method of detecting malicious code in a smart contract based on an abstract syntax tree detection tool. However, this detection tool cannot detect smart contracts that are used only to establish Command and Control (C&C) communication channels. Nicola et al. [6] reviewed the methods of attack related to the smart contracts in Ethereum, another blockchain framework, and discussed external calls for blockchain transaction-related attacks. Alexander et al. [7] proposed a detection tool for the security vulnerabilities of Ethereum smart contracts. These authors designed a detection tool that analyzes smart contracts for malicious code. However, this detection tool is still in beta, and there are many ways it can be bypassed. Liu et al. [8] proposed the ReGuard tool to detect bugs in smart contracts. However, only smart contracts established in the C++ programming environment were discussed. There are also many ways to bypass detection by this tool. For example, we can execute the attack code after it has split into multiple smart contracts.

2.2 APT Attacks on Blockchains

In a review of APT attacks, we did not find any discussion of APT attacks against blockchains.

The construction of a C&C communication channel is a crucial research problem related to the lateral movement stage of the APT attack cycle. An attacker can use C&C tools to obtain persistent and covert access to valuable data. Furthermore, attackers can abuse various protocols for C&C communications, such as the Hypertext Transport Protocol (HTTP), HTTP Secure (HTTPS), Internet Relay Chat (IRC), and peer-to-peer (P2P). Alshamrani et al. summarized the C&C tools that can be used to abuse various protocols by generating malicious traffic that will be confused with general traffic to bypass security infrastructures [3]. However, these authors did not discuss the APT-attack-based abuse of blockchain technology. Marchetti et al. [9] proposed a detection framework that can detect a large number of suspicious host activities at high speed. This detection framework can identify abnormal encrypted communications without decrypting them. However, this framework cannot separate abnormal traffic from encrypted communications involving smart contracts, which are widely used in typical production environments. McCusker et al. [10] proposed a group-based

method of detecting abnormal traffic. However, an APT attacker typically constructs a C&C communication channel through an ordinary communication traffic device using a smart contract. Consequently, the group-based method of detecting abnormal traffic is not an effective solution for detecting abnormal C&C communication.

3 Background Knowledge

This section describes some background knowledge relevant to our research, including an introduction of the Hyperledger Fabric platform, network management techniques, network protection techniques, and APT attack techniques.

3.1 Hyperledger Fabric

Hyperledger Fabric is an open-source project launched by the Linux Foundation in 2015 to promote digital technology and transaction validation for blockchains. Hyperledger Fabric uses Docker technology to host smart contracts. The smart contract applications used in this blockchain system are also known as chaincodes. In a blockchain, each peer node stores snapshots of the most recent system state in the form of key-value pairs [11].

Smart Contract: A smart contract is a computer protocol designed to disseminate, validate, or execute a contract. Smart contracts allow the execution of trusted transactions without supervision by a third party. In general, a smart contract is a segment of executable code that can deploy unalterable code without any third-party guarantee of trust. The use of smart contracts can significantly reduce manual participation, guarantee system security and efficiency, and significantly reduce transaction costs [12].

Chaincode: Chaincodes are the smart contracts of Hyperledger Fabric, which are instantiated in Docker containers [13]. However, because chaincodes run on Docker, they provide a highly convenient vector for APT attacks. APT attackers can enhance the effectiveness of their attacks by embedding malicious code in smart contracts. This malicious code can be used to execute any arbitrary operation.

Security Policy: A security policy is an essential function for smart contracts in a consortium chain. Through a suitably configured security policy, it is possible to mitigate attacks and avoid the direct execution of malicious code in smart contracts. Administrators must configure their security policies in accordance with the principle of permission minimization. If a security policy is misconfigured, a user can execute code with administrator permission, which can lead to serious security issues. We have studied the access control list (ACL) method used to divide user permissions in Hyperledger Fabric. The architecture of the Hyperledger Fabric security policy is shown in Fig. 1.

MSP: An MSP is a Membership Service Provider. The MSPs in Hyperledger Fabric provide the definitions for member operation permissions. An MSP

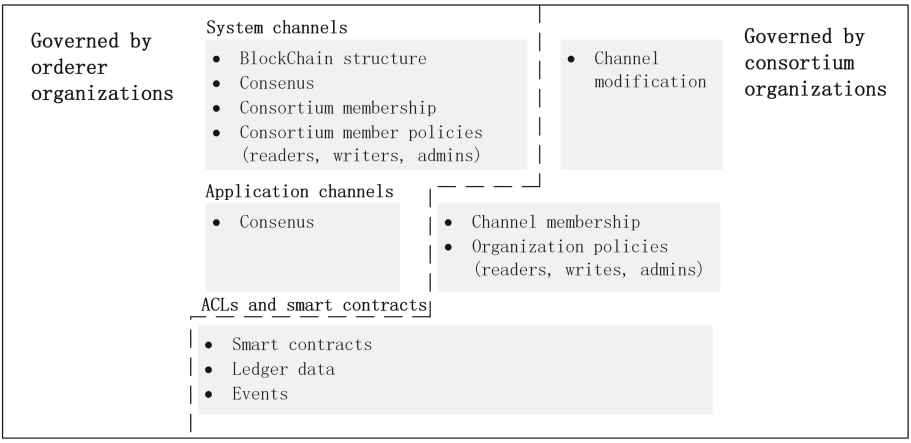

Fig. 1. Hyperledger fabric policy hierarchy [14]

abstracts the encryption mechanisms and protocols used to issue certificates, authenticate certificates, and authenticate users. An MSP manages the identities of members, the rules that govern those identities (including authentication), and authentication functionalities (including signature generation and authentication). One or more MSPs can simultaneously manage a Hyperledger Fabric network. The MSPs provide modularity of member operations and interoperability across different members [15]. Each MSP maintains cacert, keystore, signcert, tlscacert, and organization profiles.

3.2 Network Management and Protection Technology

Network management is a necessary responsibility of large enterprises. The security of an enterprise network can be enhanced by appropriate network management technology and security settings. At the same time, suitable network management technologies can make the network environment itself more hostile to attacks.

ACL: An ACL is an access control list. An ACL can be used to provide essential security control for a network. A device queries the ACL to match a received Internet Protocol (IP) packet in accordance with specified rules. If a match is found, the device will either allow or block packet transmission in a rule-based manner [16].

VLAN: A virtual local area network (VLAN) is formed by logically combining several physical subnets in different locations to function as a local area network. VLAN technology can be used to subdivide a network to limit the potential range of an attack [17].

Firewall/IDS: A firewall or intrusion detection system (IDS) is a device that APT attackers need to bypass to execute an APT attack successfully. How-

ever, the communication of smart contracts relies on Transport Layer Security (TLS), which means that firewall or IDS devices cannot be used to check such communication. Therefore, there is no inherent requirement to encrypt the contents of smart contract communication for APT attacks. However, when the victim device has a built-in IDS installed, encryption and code confusion are required [18].

3.3 APT Attack Technology

Lateral movement is a critical stage of an APT attack. Figure 2 shows the various stages of the APT attack life cycle as analyzed by FireEye, the world's leading anti-APT company. FireEye divides an APT attack into five main stages: initial reconnaissance, initial compromise, establishment of a foothold, escalation of privileges, and internal reconnaissance. Lateral movement and maintenance of presence are necessary when the initial internal reconnaissance does not acquire the necessary data.

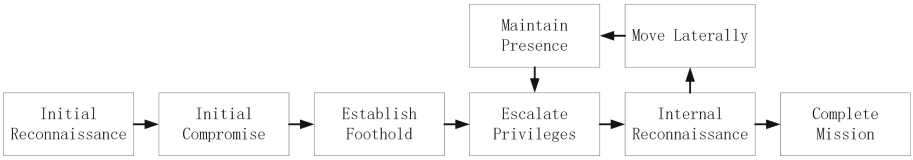

Fig. 2. APT attack life cycle [19]

Lateral Movement: Lateral movement is defined as logging onto other servers using existing credentials. Lateral movement provides communication channel support for subsequent attacks by means of remote services on a remote server [20].

Remote Command Execution: Carefully constructed code can force a victim node to execute system commands. The Docker container of the victim node can be made to execute these commands by means of code for remote command execution, which allows the attacker to control the victim Docker container [21].

4 Theory of APT Attacks Using Smart Contracts

This section describes the theory of how to abuse the smart contracts of Hyperledger Fabric to execute APT attacks. Hyperledger Fabric is a typical blockchain framework, which means that this attack theory for Hyperledger Fabric can also be extended to other blockchain systems with similar structures.

4.1 Preparation for Smart Contracts

Various programming languages can be used for coding in smart contracts. Hyperledger Fabric supports Go, Java, Node.js, and Python. For other blockchain frameworks, it would be necessary to use the corresponding software development kits (SDKs) for smart contract coding.

The coding of a smart contract includes the Init and Invoke functions. The Init function is used to initialize the smart contract, and the Invoke function is used for subsequent calls to the smart contract. The Init function is less convenient than the Invoke function for repeated calls. Therefore, we suggest that the attack code should be introduced into the Invoke method for subsequent query calls. Moreover, proper debugging and exception handling should be performed to avoid the remote smart contract container exiting due to abnormal input.

An MSP is required to certify and authenticate an installation in Hyperledger Fabric. We can use a smart contract management application to install a smart contract with the MSP. In Hyperledger Fabric, smart contracts are installed via the "peer chaincode" command. Next, "peer instantiate" or "peer upgrade" is used to start the container of a remote smart contract.

A smart contract can be called by means of a query command in the blockchain system. The blockchain system distributes smart contract operations via the communication system.

Hyperledger Fabric's official development documentation shows that the command "peer chaincode query -n mycc -c'{"Args":["query", "a"]}' -C myc" can be used to execute a smart contract. After the smart contract runs, we can see that the result is 20 [22].

4.2 Using Smart Contracts for APT Attacks

The official query example clearly illustrates the input and output structure of a smart contract, which can be widely abused in APT attacks. An APT attacker will attempt to exploit a target system, which has a controlled input and output. After the target system is exploited, the attacker will attempt to escalate his or her privileges to the highest level, which is usually the kernel privilege level for the operating system. If this privilege escalation fails, the attacker will use the victim node as a network proxy, meaning that no code is required to be written to the drive and the attack code can be executed dynamically in the memory only.

To construct a network proxy, it is necessary to place repeated, high-frequency calls to the Invoke function. At the same time, another common APT attack function can be introduced into the Invoke function.

Depending on the attack requirements, malicious code can be embedded into a smart contract to avoid disruption by the security infrastructure to the greatest possible extent.

5 Using Smart Contracts for APT Attacks

In this section, we propose three attack algorithms that abuse smart contracts to support the postpenetration stage of an APT attack. Specifically, the algorithm for malicious chaincode installation is a prerequisite for all subsequent attacks, and the remaining two algorithms will be installed on-demand as modules in the victim nodes. These two algorithms enable lateral movement or communication channel construction for subsequent attacks.

5.1 Attack Topology

The attack topology is similar to the typical topology for APT attacks [23–26]. APT attackers commonly select valuable targets, which are usually large and contain multiple network areas. Regarding the topology design, each network area should allow the passage of traffic based on certain business requirements. Additionally, each network area should include one or more network devices and be related to a defined network security boundary area. Either direct or indirect communication between the attacker and the victim should be possible through the network; they should not be entirely physically isolated. Moreover, the network should contain valuable targets or data. The attacker must be able to steal either control permission for the victim system or valuable data through appropriate attack methods.

Crucially, for the scenario of interest in this paper, the network must contain a blockchain system that can use smart contracts. The blockchain system should be spread across multiple networks and indirectly allow an attacker to access multiple networks through the blockchain system.

5.2 Attack Framework Based on Smart Contracts

According to FireEye's 2019 APT report, the attack group known as APT37 uses lateral movement methods such as the use of compromised user credentials, among others. APT38 uses lateral movement methods such as HOTWAX, NACHOCHEESE, REDSHAWL, WORMHOLE, Remote Desktop Protocol (RDP), ReDuh, TCP Gender Change Daemon, the use of compromised user and domain credentials, and Windows Group Policy. APT39 uses lateral movement methods such as BLUETRIP/REDTRIP/PINKTRIP, various publicly available tools (PsExec, RemCom, xCmd, and others), RDP, Server Message Block (SMB), and Secure Shell (SSH). Furthermore, the report also lists a large number of other lateral movement methods for APT attacks [27]. In addition, the papers of Li et al. [27] and Lu et al. [28] address methods of lateral movement in APT attacks. Overall, lateral movement is an important stage of the APT attack cycle.

There are many specific methods of lateral movement to support subsequent APT attacks for stealing data from other networks. The attacker does not need the highest level of privilege on the victim device to steal data. Instead, the

attacker can use the victim device to access additional network resources, execute a lateral movement, find valuable resources, and steal data. In practice, if a lateral movement can provide access to more resources, then that lateral movement is meaningful.

In this section, we propose an algorithm for the remote execution of commands on a victim device that uses a remote service by means of a malicious chaincode in Hyperledger Fabric. Additionally, we review various APT events in which APT organizations such as Group 72, Unit42, APT39, APT40, and Triton have used remote service methods to attack target networks and steal data [29].

The construction of a stable communication channel is a crucial problem for lateral movement. We propose the chaincode attack execution flow chart shown in Fig. 3, which is divided into eight steps:

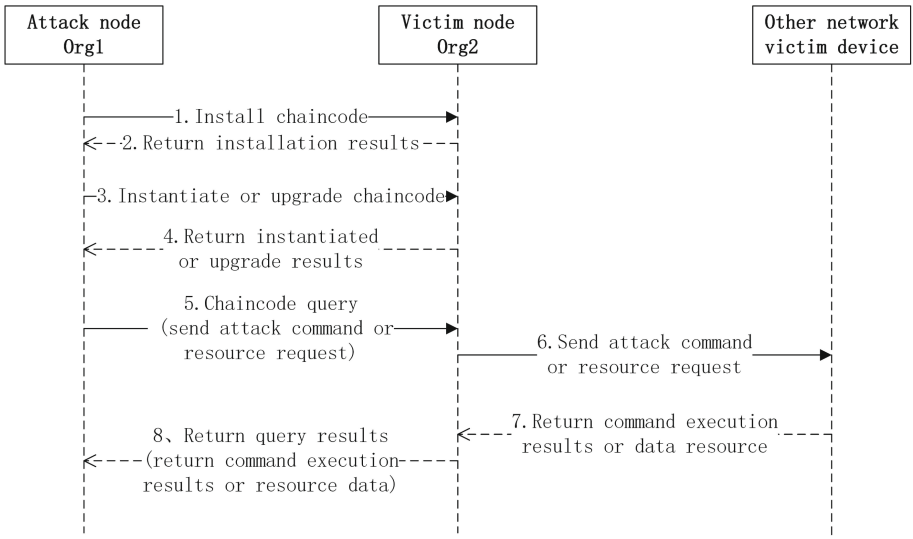

Fig. 3. Chaincode attack execution flow chart

Step 1: An attacker sends a request to install a chaincode via a controlled node. In this way, the attacker can construct an attack environment for subsequent lateral movement or other attacks.
Step 2: The results of the chaincode installation are returned.
Step 3: This step will be executed only if the returned result indicates successful chaincode installation. In this step, the attacker sends the chaincode "instantiate" or "upgrade" command to the victim node. If the chaincode is the first version, the chaincode needs to be instantiated. If the chaincode is not the first version, the chaincode needs to be upgraded. The victim node will start the Docker container of this chaincode for chaincode execution.
Step 4: The victim node returns the result of the "instantiate" or "upgrade" command.

Step 5: This step will be executed only if the returned result indicates that the chaincode on the victim node has been instantiated or upgraded successfully. In this step, the attacker executes a chaincode query command to execute system commands on the victim node through Docker remotely. Additionally, the attacker can resend attack commands or request resources from the victim node to access another victim node until the final victim node is reached. Through this step, the attacker can access multiple networks, representing lateral movement.
Step 6: This step will be executed if the attacker indirectly requests resources to access an intermediate victim node. In this step, the attacker sends a command from the current victim node to the next victim node, a process that is repeated until the final victim node is reached. Moreover, the attack payload has a multilevel embedded structure, which enables data embedding during lateral movement.
Step 7: If the final victim node is successfully accessed, the desired data will be recursively returned from the final victim node to the initial victim node. This step is complete when the initial victim node receives the data.
Step 8: The initial victim node encapsulates the data to ensure the complete transmission of the desired data to the attack node.

5.3 Attacks Based on Smart Contracts

The attacker needs to prepare a chaincode to execute remote commands or request remote resources. In this section, we propose two algorithms for

Algorithm 1. Malicious Chaincode Installation

Input: Victim Organization Administrator MSP f, Victim Peer Address a, Chaincode d, Chaincode Name n,Version v
Output: Installation Result s

```
function MaliciousChaincodeInstallation(f,a,d,n,v)
    isConnected := Connect(f,a)
    if isConnected then
        CurrentVersion:=ChaincodeVersion(f,a,n)
        if v > CurrentVersion then
            chaincodeInstallVersion:= ChaincodeInstall(f,a,d,n,v)
            if chaincodeInstallVersion == FirstVersion then
                s := ChaincodeInstantiate(f,a,d,n,v)
            else
                s := ChaincodeUpgrade(f,a,d,n,v)
            end if
        end if
    else
        s := ErrorMessage
    end if
    return s
end function
```

implementing commonly used attack types, namely, remote command execution, and large-scale resource transfer.

Preparation for Attack. The attacker must first prepare the attack environment for subsequent attacks, which requires several steps. The installation of malicious chaincode requires the MSP of the victim organization's administrator, the peer node address of the victim, the malicious chaincode itself, the chaincode name, and the chaincode version number. The algorithm for malicious chaincode installation is shown in Algorithm 1.

First, the attacker needs to establish a connection to the victim's peer node, for which the MSP of the victim organization's administrator and the peer node address of the victim must be provided. Second, if the connection is successfully established, the attacker should check the latest chaincode version in the blockchain system. Third, if the version number of the malicious chaincode to be installed is higher than the version number in the blockchain system, the chaincode in the blockchain system needs to be instantiated or updated. Then, once the chaincode installation is completed, the attacker will obtain the latest version of the current chaincode in the blockchain system. Fourth, if the latest version number is the first version, the chaincode should be instantiated. If the latest version number is not the first version, the chaincode should be upgraded. Finally, the result of malicious chaincode installation is returned.

Time complexity analysis of Algorithm 1: There is no loop and nested execution in the malicious code installation algorithm. Therefore, the time complexity of the malicious code installation algorithm is O(1).

Remote Command Execution. The attacker achieves Remote command execution (RCE) via abuse or exploitation of a vulnerability of the victim system. The chaincode for RCE requires the MSP of the victim organization's administrator, the peer node address of the victim, the command that needs to be

Algorithm 2. Remote Command Execution

Input: Victim Organization Administrator MSP f, Victim Peer Address a, Command to be Executed c, Chaincode Name n
Output: Execution Result s

```
1: function RunRemoteCommand(f,a,n,d)
2:     isConnected := Connect(f,a)
3:     if isConnected then
4:         Request(a,n,c)
5:         a.return :=a.RunCommand(c)
6:         s :=a.Response(a.return)
7:     else
8:         s := ErrorMessage
9:     end if
10:    return s
11: end function
```

executed remotely, and the chaincode name. The smart-contract-based RCE algorithm is shown in Algorithm 2.

Once malicious chaincode installation is complete, the RCE algorithm can be executed. For RCE, the MSP of the victim organization's administrator is needed to connect to the victim peer node. Once the connection is established, the attacker sends the chaincode name and the command to be remotely executed to the victim peer node. Once the victim peer node receives the request, it will execute the command and return the execution results to the attacker. If an error occurs during the RCE operation, the error message will be thrown back to the attacker.

Time complexity analysis of Algorithm 2: The RCE algorithm can call the RCE algorithm again for recursive execution. Therefore, the time complexity of the RCE algorithm is O(n).

Large-Scale Resource Transfer. In Hyperledger Fabric, the maximum length of the information that can be sent in response to a single request is 104857600 bytes (100 MB) by default. We propose a large-scale resource request algorithm to solve the problem of large-scale resource transfer. The data block length for a single transport is defined here as 50,000,000 by default. In practice, we have found that the data length tends to be doubled; therefore, we select 104857600/2 ≈ 50000000. The large-scale resource transfer algorithm is shown in Algorithm 3.

Algorithm 3. Large-Scale Resource Transfer

Input: Request URI u
Output: URI Data s

```
1: function REQUEST(u)
2:     dataLength := GetDataLength(u)
3:     splitBlockLength:=50000000
4:     s:=InitReturn()
5:     if dataLength >= splitBlockLength then
6:         for startIndex=0 ; startIndex<dateLength ; startIndex+ =splitBlockLength do
7:             getLength:=splitBlockLength
8:             if getLength>dateLength−splitBlockLength then
9:                 getLength=dateLength-splitBlockLength
10:            end if
11:            s+= GetDataByChunk(u,startIndex,getLength)
12:        end for
13:    else
14:        s:= GetData(u)
15:    end if
16:    return s
17: end function
```

First, the attacker needs to obtain the length of the file to be transferred. If the length of the file is less than the chunk length, the file will be returned

directly. If the length of the file exceeds the chunk length, then the chunking step will be executed. In the algorithm for data block chunking, each data read operation requires the current length of the data to be transferred and the current transfer offset. Then, the data will be transferred to the attacker, who will cyclically execute the transfer operation until the end of the transferred data block is read.

6 Experiments

APT attackers typically use a multilevel network proxy for their attacks. In practice, the data rates of multilevel network proxies are extremely low, and network performance generally does not need to be considered. Additionally, the transfer rate can be varied by modifying the code. This section presents experiments performed from two perspectives, namely, remote command execution, and large-scale data transfer, to experimentally demonstrate the requirements for attacks using smart contracts. By default, we first executed the attack preparation algorithm in all of the following experiments.

In the following experiments, we used Hyperledger Fabric version 1.4.4 for testing. To simulate real-world APT attacks, we designed an experimental environment consisting of two organizations (Org1 and Org2) and one orderer, with two peer nodes per organization and a Web server in the Org2 region to enable lateral movement experiments. The Web server can be accessed only from an Org2 device. We added settings for the router's ACL and for server access control to block the Org1 region from accessing the Org2-Server region and the Internet region. Our design goal in these experiments was to construct a stable communication channel by abusing a smart contract. We used a peer node of Org1 to initiate the attack. The graph presented in Fig. 4 shows the experimental topology diagram for an attack.

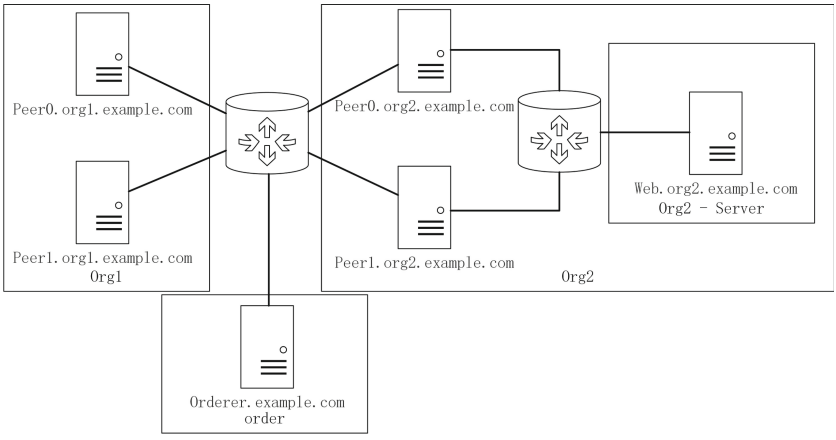

Fig. 4. Experimental topology

The configuration used for the experimental equipment is summarized in Table 1. The laboratory environment was built using the VMware ESXi virtualization platform.

Table 1. Experimental platform

Host name	CPU	RAM	NIC	Operating system
Orderer.example.com	Intel Xeon 6128 @ 3.40 GHz x4	8G	vmnic ne1000 100 Mbps	Ubuntu 18.04.3 LTS
Peer0.org1.example.com	Intel Xeon 6128 @ 3.40 GHz x4	8G	vmnic ne1000 100 Mbps	Ubuntu 18.04.3 LTS
Peer1.org1.example.com	Intel Xeon 6128 @ 3.40 GHz x4	8G	vmnic ne1000 100 Mbps	Ubuntu 18.04.3 LTS
Peer0.org2.example.com	Intel Xeon 6128 @ 3.40 GHz x4	8G	vmnic ne1000 100 Mbps	Ubuntu 18.04.3 LTS
Peer1.org2.example.com	Intel Xeon 6128 @ 3.40 GHz x4	8G	vmnic ne1000 100 Mbps	Ubuntu 18.04.3 LTS
Web.org2.example.com	Intel Xeon 6128 @ 3.40 GHz x4	4G	vmnic ne1000 100 Mbps	Windows Server 2019

6.1 Remote Command Execution (RCE)

RCE operations require the MSP of the victim organization's administrator and the peer node address of the victim.

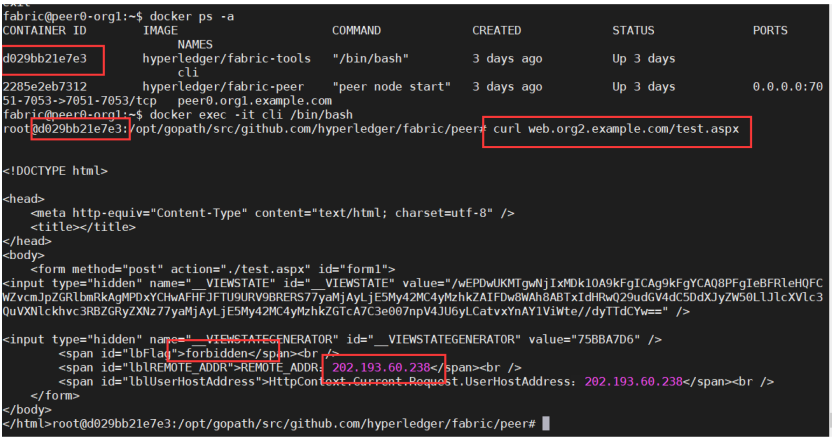

Fig. 5. Direct access is forbidden by server access control

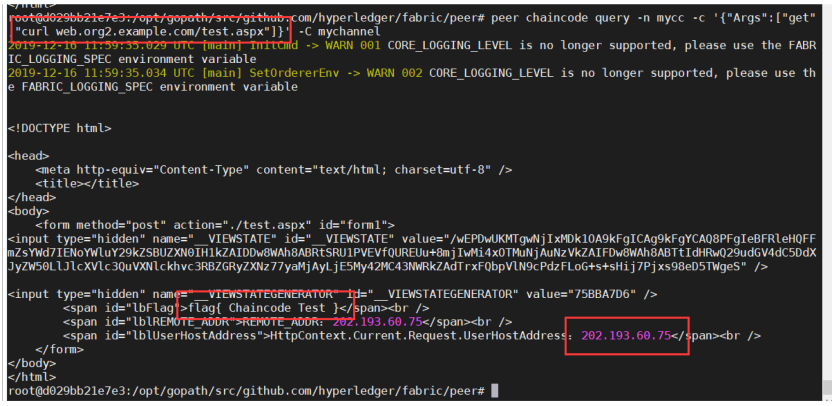

Fig. 6. Access target data via chaincode RCE

We used Hyperledger Fabric's peer application for the RCE operation. We used malicious chaincode to read out the result of a remote command executed in accordance with an input command. In this experiment, we controlled the Docker container with ID "d029bb21e7e3". Initially, we sought direct access to web.org2.example.com/test.aspx, which the flag returned by the page indicated was forbidden, as shown in Fig. 5. However, when we used RCE with the "curl" remotely command via a peer node of Org2, access to the page was granted, as shown in Fig. 6. This experiment shows that the RCE algorithm is workable. We can do the lateral movement via the RCE algorithm.

6.2 Large-Scale Data Transfer

An APT attacker usually steals data that are not fixed in size. We have proposed an algorithm for large-scale data transfer that can transfer data with no fixed size. A performance graph for the proposed large-scale data transfer algorithm is shown in Fig. 7. We used a maximum data length of 100 MB for performance testing and executed the test 1000 times with data transfer lengths varying from small to large.

We used Base64 to encode the data to ensure the correct transmission of the binary data, which increased the data length by approximately 33.3%. We defined a data limit of 50 MB for a single transfer, which should include the increase in length caused by Base64 encoding. Therefore, we chose a data length limit of 37 MB.

We found that the transmission time jitter increases as the data length increases, which indicates that larger-scale transmission has a greater effect on node performance. The proposed large-scale data transfer algorithm uses a cache to prepare the next data block for transmission, which will cost some time. We found the cache cost time to be approximately 150 ms, depending on the node performance.

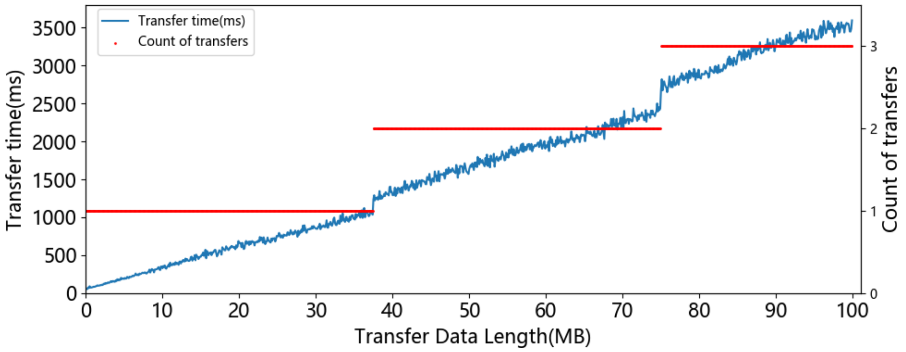

Fig. 7. Large-scale data transfer performance testing

7 Discussion

7.1 Mitigation Suggestions

Here, we propose some guidelines for attack mitigation to increase the difficulty of APT attacks and reduce the security risk posed by using blockchain smart contracts.

Keep the MSP Secure: An attacker can use the MSP of the victim organization to cause a victim node to perform any arbitrary tasks. The administrator may incorrectly certify network management actions, including copying the MSP to another computer, which would allow the administrator certificate to be stolen. An APT attacker who has stolen the MSP of a victim organization's administrator can control all nodes of the victim organization. We propose that administrators should avoid copying the MSP to any other device and should ensure that the MSP is used only on necessary nodes.

Split Large Organizations into Smaller Organizations: If the MSP of a large organization's administrator is stolen, this will place many nodes under the control of hackers. We propose splitting large organizations into smaller organizations to reduce the potential scope of the damage.

Check Smart Contract Code for Safety: Endpoint protection software can be used to monitor smart contracts in real-time, thus enabling the prompt detection of attack behavior.

Restrict the Node Communication Range: The access range of each node should be minimized, and nonessential access should be blocked.

Update the Blockchain System Version: If a new version fixes vulnerabilities, the blockchain system version should be updated.

7.2 Attack Supported Platforms

The key element of using smart contracts for APT attacks is that smart contracts can interact with outside the blockchain system for network transmission, system command execution, and other attack operation. For the programming language and remote deployment of smart contracts, we compared smart contracts in well-known blockchains, which shown in Table 2.

Table 2. Comparison of smart contracts in well-known blockchains

Platform	*Hyperledger Fabric*	*Corda*	*Ethereum*
Programming Language	Go, Java, etc. [30]	Kotlin, Java, etc. [31]	Solidity [32]
Remote Deployment	Yes [34]	Limited [35]	Yes [36]
Possibility to APT attack	High	Middle	Low

According to the survey, Hyperledger Fabric and Corda support general programming languages such as Java. Therefore, we can use general programming languages to call the operating system to implement APT attacks.

However, Ethereum uses its custom language, namely, Solidity. We can compile the Solidity program through a private compiler of Ethereum to generate Ethereum-based special machine code, which runs on the Ethereum Virtual Machine (EVM) [33]. Therefore, it is almost impossible for us to use Ethereum for APT attacks. At most, we can only use Ethereum's smart contracts to transfer data.

7.3 Next Research Plan

In the next research plan, we will try to use Corda's smart contract for APT attacks. Moreover, we will try to find an exploit to the machine code of Ethereum's smart contract, which can be further used to implement more kinds of APT attacks.

APT attacks have a variety of attack methods. We not only use smart contracts for the APT attack, but we can also insert malicious code into the source code of open-source blockchain to achieve APT attacks. According to Murphy's law, all of this is possible, and this is our motivation for further research.

8 Conclusion

APT attacks are a major concern in global cybersecurity, and corresponding advances in both attack theory and defense theory are urgently needed. Existing APT attack and defense mechanisms focus on typical protocols and services, whereas security against blockchain-related APT attacks is less often discussed. Moreover, most research on blockchain-related attacks and protection has focused on transactions.

This paper proposes ways to abuse blind spots in research on the smart contracts used in blockchains, addressing both attack and defense. We implemented remote command execution and large-scale data transfer by abusing a smart contract. We tested whether the data transmission rate for 100 MB of data can reach the average speed of three-level network proxies across multiple continents [37], which can fully support the maximum link rate required for an attack. The results demonstrate the necessity of defining a strict security policy for attack mitigation to reduce the associated losses.

References

1. Averin, A., Averina, O.: Review of blockchain technology vulnerabilities and blockchain-system attacks. In: 2019 International Multi-conference on Industrial Engineering and Modern Technologies, FarEastCon 2019, pp. 1–6 (2019)
2. Hyperledger: Hyperledger Supporting Members (2020). https://www.hyperledger.org/members
3. Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities. IEEE Commun. Surv. Tutor. **21**(2), 1851–1877 (2019)
4. Emm, D.: APT review: what the world's threat actors got up to in 2019 (2019). https://securelist.com/ksb-2019-review-of-the-year/95394/
5. Yamashita, K., Nomura, Y., Zhou, E., Pi, B., Jun, S.: Potential risks of hyperledger fabric smart contracts. In: 2019 IEEE 2nd International Workshop on Blockchain Oriented Software Engineering, IWBOSE 2019, pp. 1–10 (2019)
6. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on Ethereum smart contracts. In: International Conference on Principles of Security and Trust, pp. 1–24, no. July (2015)
7. Mense, A., Flatscher, M.: Security Vulnerabilities in Ethereum Smart Contracts (2018)
8. Liu, C., Chen, Z., Chen, B., Roscoe, B.: ReGuard: finding reentrancy bugs in smart contracts, pp. 3–6 (2018)
9. Marchetti, M., Pierazzi, F., Colajanni, M., Guido, A.: Analysis of high volumes of network traffic for advanced persistent threat detection. Comput. Netw. **109**, 127–141 (2016)
10. McCusker, O., Brunza, S., Dasgupta, D.: Deriving behavior primitives from aggregate network features using support vector machines. In: 2013 5th International Cyber Conflict (CyCon) (2013)
11. Zhou, B., Man, Y., Liu, N., Zhang, Y., Zhang, C.: Biological data security management based on Hyperledger Fabric. Cyberspace Secur. **10**(4), 55–60 (2019)
12. Pu, H., Ge, Y., Yan-feng, Z., Yu-bin, B.: Survey on blockchain technology and its application prospect. Comput. Sci. **44**(4), 1–7 (2017)
13. Hyperledger: Hyperledger Fabric Chaincode (2020). https://github.com/hyperledger/fabric/blob/master/docs/source/chaincode.rst
14. Hyperledger: Hyperledger Fabric Policy Hierarchy (2020). https://github.com/hyperledger/fabric/blob/master/docs/source/policies/Fabric_Policy_Hierarchy.pptx
15. Hyperledger: Hyperledger Fabric Membership Service Providers (MSP) (2020). https://github.com/hyperledger/fabric/blob/master/docs/source/msp.rst
16. Fang, W., Guodong, H., Xin, L.: Access control list of router and research of realization. Comput. Eng. Des. **28**(23), 5638–5640 (2007)

17. Wei, W., Jie, D., Yin, L.: Network security and LAN security solutions. Tech. Autom. Appl. **2**, 41–44 (2001)
18. Wang, X., He, J., Xie, Z., Zhao, G., Cheung, S.C.: ContractGuard: defend ethereum smart contracts with embedded intrusion detection. IEEE Trans. Serv. Comput. 1–14 (2019)
19. FireEye: Red Team Cyber Security Assessments (2020). https://www.fireeye.com/services/red-team-assessments.html
20. Bohara, A., Noureddine, M.A., Fawaz, A., Sanders, W.H.: An unsupervised multi-detector approach for identifying malicious lateral movement. In: Proceedings of the IEEE Symposium on Reliable Distributed Systems 2017-Septe(Lm), pp. 224–233 (2017)
21. MITRE: Command and control (2020). https://attack.mitre.org/tactics/TA0011/
22. Hyperledger: Chaincode for Developers (2018). https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode4ade.html
23. Mandiant: The Advanced Persistent Threat. FireEye (2010). https://content.fireeye.com/m-trends/rpt-m-trends-2010
24. MANDIANT: 2013 threat report. FireEye (2013). https://content.fireeye.com/m-trends/rpt-m-trends-2013
25. FireEye: M-Trends 2015: A View from the Front Lines Threa. FireEye (2015). https://content.fireeye.com/m-trends/rpt-m-trends-2015
26. FireEye: Mandiant consulting 2016. FireEye (2016). https://content.fireeye.com/m-trends/rpt-m-trends-2016
27. Li, M., Huang, W., Wang, Y., Fan, W.: The optimized attribute attack graph based on APT attack stage model. In: Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016, pp. 2781–2785 (2017)
28. Lu, G., Guo, R., Wang, J.: An analysis of the behavior of APT attack in the Ngay campaign. In: 2018 18th IEEE International Conference on Communication Technology 2019-Octob, pp. 1115–1122 (2019)
29. MITRE: MITRE ATT&CK™ Remote Services - Enterprise (2020). https://attack.mitre.org/techniques/T1021/
30. Hyperledger: Hyperledger Fabric application SDKs (2020). https://hyperledger-fabric.readthedocs.io/en/latest/getting_started.html#hyperledger-fabric-application-sdks
31. R3: Getting set up for CorDapp development (2020). https://docs.corda.net/docs/corda-os/4.5/getting-set-up.html
32. Ethereum: The Solidity Contract-Oriented Programming Language (2020). https://github.com/ethereum/solidity
33. Ethereum: A Python implementation of the Ethereum Virtual Machine (2020). https://github.com/ethereum/py-evm
34. Hyperledger: peer chaincode (2020). https://hyperledger-fabric.readthedocs.io/en/latest/commands/peerchaincode.html
35. R3: Building and installing a CorDapp (2020). https://docs.corda.net/docs/corda-os/4.5/cordapp-build-systems.html
36. Yohanes.gultom: Compiling, deploying and calling Ethereum smartcontract using Python (2020). https://yohanes.gultom.me/2018/11/28/compiling-deploying-and-calling-ethereum-smartcontract-using-python/
37. Yi, Y., Han, G., Yi-jun, W., Zhi, X.: Darknet resource exploring based on tor. Commun. Technol. **50**(10), 2304–2309 (2017)