1-1-2020

Verification of Cryptocurrency Mining Using Ethereum

Dong Her Shih
National Yunlin University of Science and Technology

Ting Wei Wu
National Yunlin University of Science and Technology

Tzu Hsin Hsu
National Yunlin University of Science and Technology

Po Yuan Shih
National Yunlin University of Science and Technology

David C. Yen
Texas Southern University

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 30, 2020, accepted June 18, 2020, date of publication June 29, 2020, date of current version July 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005523

Verification of Cryptocurrency Mining Using Ethereum

DONG-HER SHIH[1], TING-WEI WU[1], TZU-HSIN HSU[1], PO-YUAN SHIH[2], AND DAVID C. YEN[3]

[1]Department of Information Management, National Yunlin University of Science and Technology, Douliu 640, Taiwan
[2]Department of Finance, National Yunlin University of Science and Technology, Douliu 640, Taiwan
[3]Jesse H. Jones School of Business, Texas Southern University, Houston, TX 77004, USA

Corresponding author: Dong-Her Shih (shihdh@yuntech.edu.tw)

This work was supported in part by the Taiwan Ministry of Science and Technology under Grant MOST 108-2410-H-224-040.

ABSTRACT With advancement in computer technology, financial technology has also evolved. Blockchain technology has evolved over the past decade; however, this has also resulted in some malicious attacks. To earn mining rewards of a blockchain, a new type of attack, called cryptojacking, has emerged in the online world. This attack uses the computer resources of a victim to obtain mining rewards without user confirmation. However, this monetization model was originally intended to replace advertising as a source of revenue for a website. To resolve such problems, a storage verification architecture based on smart contracts was proposed in the study. The decentralized system of blockchain enables users to identify and participate in verification of mining sites through Ethereum smart contracts.

INDEX TERMS Blockchains, cryptojacking, malicious detection, smart contracts.

I. INTRODUCTION

With the rapid development of information technology and the Internet, various industries, governments, and institutions are frequently using information and communication technology to improve their processes. However, various attacks can occur in unexpected places. A paper titled ''Bitcoin: A Peer-to-Peer Electronic Cash System'' (published in November 2008) describes methods to use a peer-to-peer (P2P) network to create an electronic trading system that does not require user dependencies and trust [1]. This network combines each transactional information with proof of work (PoW) to form a decentralized database.

Eskandari *et al.* [2] refer to a new form of attack, namely cryptojacking. Such an attack uses the computing resources required by cryptocurrency to verify the transaction, and an executable mining script is embedded in the webpage to occupy the computer resources of the user, which enables the malicious attacker to obtain the mining reward. In one case of abuse, Pirate Bay, a well-known BitTorrent file-sharing website, had an abnormally high CPU usage (80%–85%) when browsing in September 2017 because it was using Coinhive's Javascript mining script.

In addition, in recent years, many media reports have emerged on cryptocurrency mining. In the first half of 2018, Trend Micro (www.trendmicro.com) indicated that in the first half of 2018, 47 cryptocurrency mining malicious software series were detected, including various forms of attacks, indicating cryptocurrency mining. The number has more than doubled compared with the second half of 2017, proving that cybercriminals are interested in cryptocurrency mining. Malicious mining programs are a topic that has been gradually valued, but the current research in academia is relatively limited compared with other common malicious attacks. In addition, some studies have revealed that the detected malicious mining cases of webpages are rapidly increasing, and the damage caused by them has been estimated. The extra power consumed per day exceeds 278 000 kW, and all attackers earn a daily income of at least US$59 000 [3]. Malware has features that capture common computer resources, such as CPU, GPU, memory, and storage space, without the knowledge of the user [4].

Eskandari *et al.* [2] revealed that benign web mining must be performed to inform web users of their responsibilities. However, no clear definition of the extent to which webpages maliciously exploit system resources has been defined. Therefore, the goal of this study is to store the data in an Ethereum smart contract to achieve a publicly verifiable effect. This can provide the basis for other sites to distinguish mining behavior when a user is browsing the website. Therefore, we attempted to resolve the following concern in this study: Whether the blockchain smart contract

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

situation can be used as the basis for webpage mining certification.

The second section introduces the preknowledge and literature review, from the introduction of Bitcoin to the later extended Ethereum, Monroe, and others, and finally, describes the web mining framework to be certified by the institute. The third chapter defines the experimental setup, proposed system flow, and the designed contract SiteManagement and VerifiedSite. The fourth chapter analyzes the common vulnerabilities of smart contracts.

II. LITERATURE REVIEW AND PRELIMINARY KNOWLEDGE

Before Bitcoin was released, several virtual currency technologies and products were already in existence in the online world. Although blockchain technology has evolved over the years, Bitcoin still is prominent in the field of cryptocurrency. This article sorts out the origins, techniques, and mechanisms of blockchain technology, and then delves deep into the web mining in the field of cryptocurrency.

A. BLOCKCHAIN TECHNOLOGY

Blockchain was developed as the core technology of Bitcoin. After approximately a decade of development, Blockchain has gradually become one of the most groundbreaking technologies today, covering a wide range of industries including finance, manufacturing, and educational institutions [5]. In the field of IoT, blockchain technology can allocate addresses of large number of IoT devices, and transform the ownership of the long-life cycle. [6] With the generation of 5G technology, blockchain can also solve problems such as 5G infrastructure sharing, Network Slicing and international roaming [7]. In addition to Bitcoin, the currency generated by the same or similar technical concepts is Altcoin (competition currency) and is present in the trading market with Bitcoin [8].

(1) Bitcoin: Nakamoto [1] illustrated Bitcoin's technical logic, basic technology concepts, and how to use a P2P network to create an electronic trading system that does not require dependencies and trust. Bitcoin uses P2P architecture and cryptography to maintain the security of the entire Bitcoin network. The P2P network does not have a primary server to operate. The participants of Bitcoin are the client nodes, namely users and miners. The user can transfer and execute transactions with Bitcoin. The miner is responsible for calculating the PoW, generating the block broadcast to other nodes for verification, and then obtaining the corresponding amount of Bitcoin as compensation. The block consists of multiple transactions, the transaction is collected by the miner, and the work area certificate is used to calculate the address of the next block and verify it to generate a new block.

As depicted in Fig. 1, the blocks are linked to each other by using hash value to form a database system. If the PoW is calculated and the block is validated by other nodes, the block

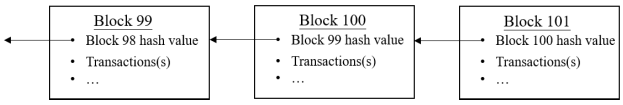


FIGURE 1. Link between the hash value and the blocks.

is written to the database, proving and recording a series of events.

(2) Ethereum: Ethereum [9] is one of the widely used blockchain networks, with a currency called ether recirculating. Smart contract is a program of Ethereum that is executed on the blockchain. Similar to general programming languages, smart contracts also have states and functions. The smart contract must be executed by the Ethereum virtual machine and running on the Ethereum node. The programming language for smart contracts includes the solidity that the Ethereum team takes over, Vyper based on Python 3 for Ethereum Smart contracts, and Serpent a high-level language designed for writing Ethereum contracts.

Unlike Bitcoin networks, accounts on Ethereum are divided into two types, namely externally owned account (EOA) and contract account (CA). The EOA is the account held by the user, including information such as address and account balance. The CA is an account attached to the contract, and the EOA also contains the address and balance, which must be established by the user. After the issuance of the smart contract, if you want to update the contract, then you can only create a new contract and cannot change the old contract. Programs that write smart contracts must consider not only the security measures of previous designs but also the security issues specific to blockchains.

In introducing Ethereum into existing systems, Wessling *et al.* [10] indicated that it is difficult to determine which attributes of the blockchain are vital to existing systems. The difficulty is deciding which elements in the architecture should use blockchain technology. Wood [9] proposed a flow for determining whether the technical level is suitable for applying blockchains.

(3) Monero: For most people, financial privacy is highly crucial. In recent years, numerous companies, banks, and government records have been maliciously damaged, leaking information about users and balance sheets. Such a result can be attributed to transactions with no transaction security [12]. One of the cryptocurrencies, called Monero, is developed on the principles of privacy, decentralization, and extensibility, and provides a platform for financial activities that value privacy.

Kumar *et al.* [13] investigated the untrackable nature of Monero and determined that even after Monero launched an update, the amount of transactions was hidden. The study revealed the convenience of using Monero in crime and malicious attacks. By default, in Monero, the receiving address and the transaction amount for each transaction remain untraceable.

The CryptoNote protocol used by Monero can build an anonymous and complete transaction, but it renders the usage

of this cryptocurrency more prominently than other currencies for crime and other purposes [2]. The famous WannaCry ransomware incorporates Monero's more transactional privacy features, converting from Bitcoin paying ransom to Gallagher [14]. Because of its privacy and difficult tracking, approximately 44% of ransomware attacks use Monero to obtain ransom, and its transaction fee is relatively low [15].

B. WEB-BASED CRYPTOMINING

The Monero development community has enabled Monero to achieve cryptocurrency decentralization, enabling the used low-end equipment to mine. Unlike Bitcoin and other currencies that use the SHA256 algorithm, Monero relies on ASIC mining to monopolize mining rewards. The cryptocurrency using the CryptoNote protocol is currently the most profitable category for personal computer mining, with Monero being the most prominent. Coinhive developers have used these features and have become the most popular web mining service provider.

If a website administrator wants to introduce Coinhive's web mining function to their website, then the mining program application programming interface of Fig. 2 can be added to the source code of the webpage to start mining. Except for SITE_KEY and username, which are set by the website administrator, the others are not required to be changed.

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<script>
    var miner = new CoinHive.User('SITE_KEY', 'username');
    miner.start();
</script>
```

FIGURE 2. Coinhive web-based mining API.

Eskandari *et al.* [2] clearly defined cryptojacking, which is unless the user's consent is obtained, it is considered a malicious abuse.

In [3], most samples were determined to use a 70% CPU usage setting. Approximately 30% of webpage mining scripts exhaust computer resources to maximize their profits, and it is difficult to judge the cryptocurrency mining of webpages through CPU usage alone. Few webpages inform about mining or even user treaty submissions. Some types of webpages were examined in the study, and some words were used to identify 35 webpages that were mining under certain types of user agreements. Approaches in which users' webpages are attacked through malicious web mining programs involves a wide range of attacks. Various attacking entities can inject mining scripts into the website's code base. In addition to the web mining services provided by Coinhive, many other service providers and even mining scripts written by attackers are available.

C. RESEARCH QUESTIONS

Most of the blockchain research is considering how to apply blockchain technology into real scenarios and reduce third-party regulatory. The research question of this study is: Can the blockchain smart contract situation be used as the basis for webpage mining certification?

The research in [16] was based on an analytical instruction set architecture that can detect up to 100% web mining. In addition, various websites were classified as malignant or benign. Therefore, from the perspective of the behavior of PoC, the frequency of hashing is a vital indicator. Webmasters or malicious script injectors can easily adjust mining programs with minor modifications. Sites that have been altered must be re-evaluated. The study conducted in [3] determined that the phenomenon of malicious mining of webpages is increasing, according to the number of their discovery on Alexa's top 100K website, which was 260% more than the number detected in previous studies. Mining behavior not only consumes the computing resources of the computers of the victims but also consumes electricity. It was determined that this malicious behavior consumes more than 278 000 kW per day, and the attacker earns at least US$59 000 a day.

Although mining program detection is considered serious and the detection aspect can be applied, there are hard to find reliable proof of the mining webpage for user's compensation. Therefore, a blockchain based application that enables users to identify and participate in verification of mining sites through Ethereum smart contracts was proposed in this study.

III. PROPOSED SYSTEM

Blockchain technology is a combination of security and transactions. From Bitcoin's example, it is easy to see the possibility of blockchain applications. Ethereum uses an easy method to implement blockchains [7]. Applications that import blockchains are known as distributed applications (dApps), which are applications on P2P networks rather than on a single device. Such application is typically designed to exist on the network in a manner that is not controlled by any single entity. To solve the problem of web mining, a solution based on the smart contract of the Ethereum platform was proposed. This program provides a publicly verifiable, scalable, and rewarding mechanism. This section describes the context of the proposed system, explains the system flow, the requirements of the system, and the pseudocode of the Ethereum smart contract.

The concept of cryptojacking is presented in Fig. 3. The web-based miner service provider provides a web-based miner service script. The webmaster of the website uses and adjusts the script of the webpage or attacker to inject or modify the webpage mining script. When the informed or uninformed victim is browsing the web, the script is performing PoC on the cryptocurrency that is being mined.

A. SYSTEM REQUIREMENT

To understand the process of the smart contract verification mining website, a proposed system flow has been presented. According to the process, when the user uses a website, the detection program automatically verifies whether the website is performing malicious mining. Next, the detection

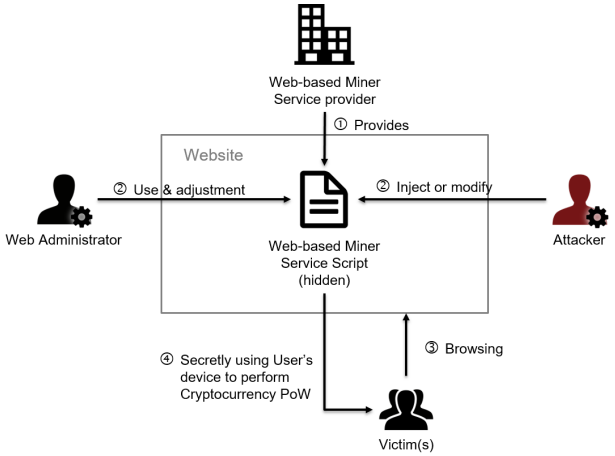


FIGURE 3. Web-based mining scenario.

program updates the smart link of the blockchain to add or update the status of the browsing site. If the website administrator determines that the website he manages has an error, then he can pay the token to request the smart contract to update the content. Before introducing the contract, it is necessary to organize and explain the key functions. The key features are listed as follows:

1. A program interface must provide different detection program entry (smart contract function).
2. After detecting the mining behavior, a feedback token mechanism must be provided to the user.
3. Website managers can pay for token request contracts to update their website.

B. SYSTEM PROPOSAL

Fig. 4 illustrates the system architecture. The web-based miner service provider provides miner scripts, or attacker performs injection or modification of web mining scripts. When the users browse the webpage, they can use the detection program that interacts with the dApp. The detection program verifies the website and then sends the message back to the dApp to establish a public verification environment by establishing a contract or certification on Ethereum. If a web administrator has doubts regarding the content of the verification, he provides a token (ether in this case) using a function that requests reverification for the user to reverify. Then users can perform 2–4 steps on the same website and obtain a token reward. SiteManagement Owner (Contract Owner) plays the role of maintaining some of the features and providing a distributed application. Through the characteristics of the blockchain, Smart contracts can record whether a website is injected into a miner script and give users feedback, hope this will encourage users to detect if a website has been scripted. Fig.5 is a sequence diagram of the system.

C. IMPLEMENTED CONTRACTS

Table 1 lists the equipment and environment used in this study. Several tools were used in this study. Geth [17] is a command line interface of the complete Ethereum node

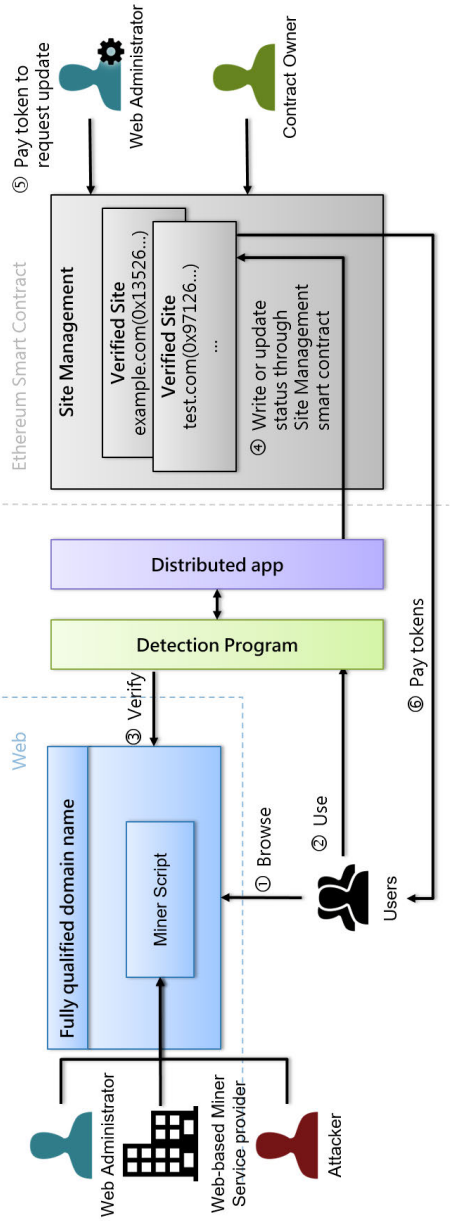


FIGURE 4. System flow.

implemented in the language Go. In addition to excavating real ether, it can not only transfer funds between accounts, establish smart contracts and send transactions but also explore features such as block content. The integrated development environment (IDE) Remix [18] is a browser-based compiler and IDE that builds Ethereum contracts and debugs using the Solidity language.

For this study, the smart contracts designed and used were SiteManagement and VerifiedSite. Fig. 6 illustrates the architecture of the contract for this study on the IDE Remix. SiteManagement is designed to manage contracted sites that have been certified. VerifiedSite is used to store managed website certifications. The following describes the algorithm virtual code for each function.

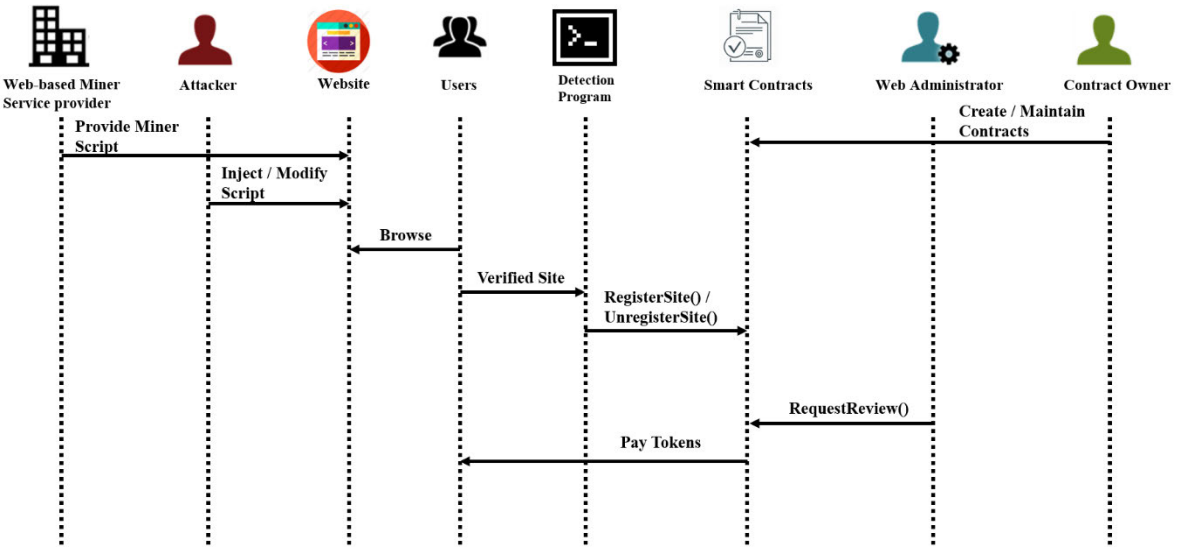


FIGURE 5. System sequence diagram.

TABLE 1. Development and test environment.

Parameter	Value
Machine Spec	Windows 10 Enterprise Intel Core i5-7200U 2.7GHz 8GB RAM
Ethereum Client	Geth v1.8.23 [15]
IDE	Remix IDE [16]
Test Environment	Ethereum Private chain Remix IDE

FIGURE 6. Deployed contracts.

1) VERIFIEDSITE

VerifiedSite contract stores certified website information and processes review functions. To reduce the wastage of gas, the design is focused on the process of establishing a contract and the result of the first detection is written into the contract rather than using a function to update after creating a new contract.

TABLE 2. Detection structure used by VerifiedSite.

Type	Variable Name
Address	*owner*
Uint	*expireTime*
bytes32	*detectionTech*
bytes32	*detectionTechVersion*
Bool	*malicious*
bytes32	*description*

As presented in Table 2, detection is the structure used to store the individual detection verification in VerifiedSite contract. The variables in the structure were as follows: the *address* of the detector, expiration time of the detection *expireTime*, detection method, and version *detectionTech*. Finally, it is judged as *malicious* and its *description*. The constructor of VerifiedSite contract writes the necessary information when the contract is first created to avoid the waste of gas caused by the two transactions that create the contract and then add the new data. VerifiedSite contract also uses several global variables to store the status, the full qualify domain name (*FQDN*) is used to identify the name of the website, the *reviewable* is used to identify the current review status, TRUE is the executable review, FALSE is the opposite, and the *bounty* is used to set the reward when reverify is performed. *Detections* are used to store all detected dictionaries.

requestReview(): When a particular web administrator has doubts about the content of the website's certification, he can request a reconsideration, and he can also attach a token to increase the motivation for user verification. This study set the amount of ether that web administrator can provide between 0.01 and 1 ETH. If the *reviewable* is already true, using this function returns all ethers that were transmitted at the time.

Review(*_detectionTech*, *_detectionTechVersion*, *_malicious*, *_description*): After the web administrator requestReview(), users can use review function to append the record

Algorithm 1 Pseudocode of RequestReview()

Inputs: amount of ether: *ether,*
if *reviewable* is false and msg.value limited from 0.01 to 1 ether **then**
  bounty ← *ether*
  reviewable ← true
else
  return fund to requester
end if

TABLE 3. Site structure used by SiteManagement.

Type	Variable Name
Address	*Owner*
Address	*Addr*
Uint	*Timestamp*
bytes32	*Description*

of the website. Resolving reentrancy problems is necessary because of the transfer of tokens. The design problem was solved by using the checks–effects–interactions pattern. The details are mentioned in the next section.

Algorithm 2 Pseudocode of Review()

Inputs: transaction sender: *reviewer*
  new approach that detection using: *_detectionTech*,
  new approach version that detection using: *_detectionTechVersion*,
  new detection status: *_malicious*,
  new detection description: *_description*
if *reviewable* is true and blocktime > *Detections*[length(*Detections*)].*expireTime* **then**
  Specify the New detection *det*
  Specify Detection structure variable *det* and assign the *reviewer*, *_detectionTech*, *_detectionTechVersion*, *_description*.
  Append *det* to *Detections*
  Specify *share* by value of *bounty*
  bounty ← 0
  reviewable = false
  Send *share* amount ether to *reviewer*
else
  Do nothing
end if

2) SITEMANAGEMENT

SiteManagement contract is used to manage the storage of verified website contracts. The registerSite() and unregisterSite() functions are contracts for adding and deleting management, in addition to setters and getters. This section describes the data structure used and the two main features.

As presented in Table 3, the data structure used to store individual websites is *Site*, which contains the *owner* address and contract address (*addr*) of 20 bytes in length, the UNIX format time of the block when the website is registered (*timestamp*), and the *description* field. In addition to the use of Site, several global variables were designed to provide the function usage, namely the address *creator* of the contractor, the number of registered websites *numSites* and the dictionary *regSites* indexed using *FQDN*. Store the previously unregistered dictionary *unregSites*.

registerSite(_fqdn, _addr, _description): This function writes VerifiedSite contract to SiteManagement. The contract stores *_fqdn* as the site identification, the corresponding VerifiedSite contract address *_addr*, additional information *_description*, and the time block.timestamp to write to the block.

Algorithm 3 Pseudocode of RegisterSite()

Inputs: Site full qualify domain name: *_fqdn*, address of VerifiedSite contract: *_addr*, extra information: *_description*
Output: A boolean (*True* or *False*) represent success or fail
if *_fqdn* not isn't saved by SiteManagement contract **then**
  Specify Site structure variable *site* and assign the *_fqdn*, *_addr*, *_description,* block.timestamp to it.
  Append *site* to *regSites*.
  Return true
Else
  Do nothing
  Return false
end if

unregisterSite(_fqdn): When an error occurs in a registered VerifiedSite, an unregistered function is designed to remove registered with VerifiedSite. To leave a record, the cancelled website is stored in *unregSites*. If _fqdn has been stored in VerifiedSite and the user who executes this function is the user who originally registered this website, VerifiedSite can be executed smoothly. Otherwise it will return FALSE directly and lose gas.

Algorithm 4 Pseudocode of UnregisterSite ()

Inputs: Site full qualify domain name: *_fqdn*
Output: A boolean (*True* or *False*) represent success or fail
if fqdn is saved by SiteManagement contract and address of owner equals function performer **then**
  Make a copy of *regSites*[*_fqdn*] to *unregSites*.
  Wipe *regSites*[*_fqdn*] record.
  Return true
else
  Do nothing
  Return false
end if

TABLE 4. Roles and their addresses.

Alias	Address
User A	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
User B	0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db
Web Administrator	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

D. ROLE-BASED FUNCTION

Next, we explain the functions that can be used according to each role, as depicted in Fig. 6. Users can create a specific website VerifiedSite contract and then register to SiteManagement contract using the registerSite() function. In case the user finds that the website is set to be wrong, he can use unregisterSite() to log out of the website. The web administrator can use requestReview() to request to update the status of VerifiedSite and set the reward token to increase the motivation for the user to update. After the website manager uses requestReview() and the *expireTime* expiration time in VerifiedSite contract is also reached, the user can review and recycle the reward token set by the web administrator.

E. CONTRACT TESTING

The test results of the contract are introduced in the section. The test environment is Remix IDE. Table 4 lists the roles and addresses that are used in the next two sections to compare the detailed results of the transaction.

1) VerifiedSite

When this contract is established, the first data are written into the data structure site. The data written by user A were as follows: the detected website FQDN (www.test.com; 0 × 7777772.746573742e636f6d), detection method (approach_a; 0 × 617070726.6163685f61), detection mode (version 1.0; 0 × 312.30), detect malicious results (true and description field: information; 0 × 696.666f726d6174696f6e). Fig. 7 demonstrates the content of the information required to establish the contract and the detailed result of the contract. Decoded input illustrates aforementioned information.

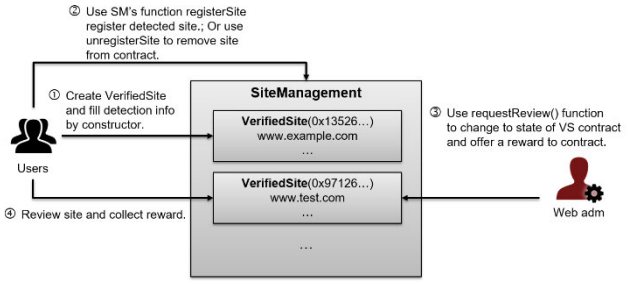


FIGURE 7. Role-based function.

Fig. 8 displays the web administrator requesting a website update, using requestReview() to send 0.1 to the contract. The balance change of the contract can be seen from the value field, and the reward is changed from the previous place to the detection method changed to approach_b (0 × 617070726.6163685f62) and changed to the nonmalicious website.

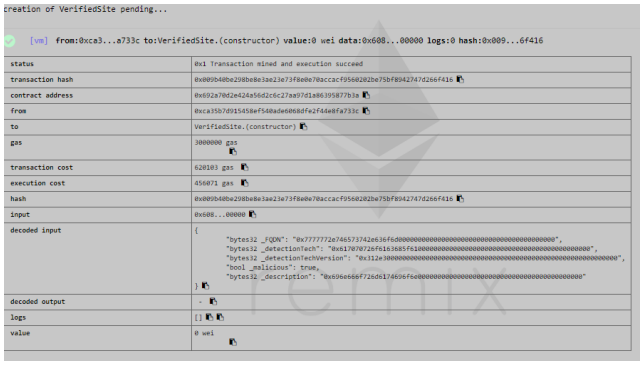


FIGURE 8. VS contract constructor and successful output.

2) SITEMANAGEMENT

After SiteManagement is completed, the contract must execute the registerSite function to write the certified website into the contract's data structure. The information required by registerSite is: full domain name *fqdn*: www.test.com, previously established contract address and description provided from VerifiedSite.

Execute unregisterSite to erase the already registered contract. As long as you enter *fqdn*, the contract creator can log out the website. The Truffle framework is employed as a development platform for smart contracts in this work, its an Ethereum like blockchain simulation platform, we can compile and deploy the smart contracts in this platform, with the Ganache GUI client is used to monitor the running status of the blockchain for testing the smart contracts [19]. Fig. 9 (a) and (b) are testing and monitoring smart contracts through the Ganache GUI.

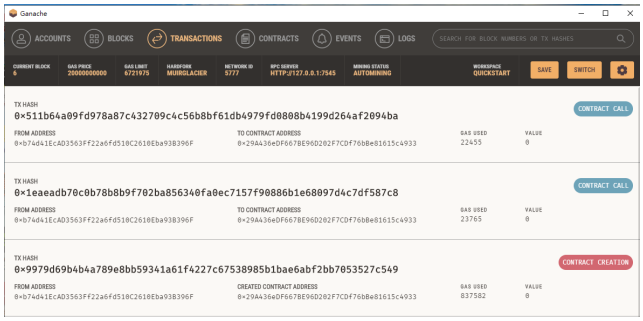
IV. SECURITY ANALYSIS AND DISCUSSION

SiteManagement and VerifiedSite presented in the previous chapter represent the contract for managing the identified sites and the site-specific contracts addressed by FQDN. Although the smart contract provides a certain degree of transparency, some weaknesses in the design persists, which are vulnerable to attack.

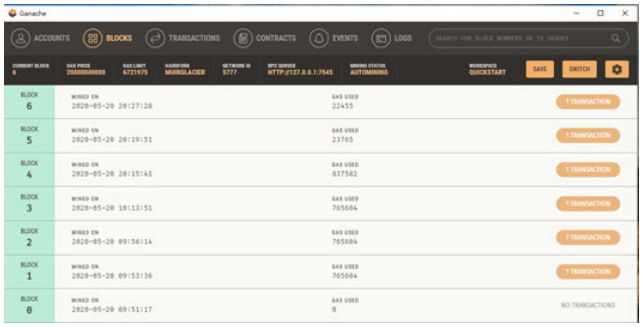
In [19], the security issues of smart contracts were introduced and some improvements were suggested. Common vulnerabilities include reentrancy vulnerability, transaction–ordering dependence, mishandled exceptions, and timestamp dependence. We analyze these issues one-by-one and compare them with other smart contract related research.

A. REENTRANCY VULNERABILITY

When a smart contract uses a remittance-related function, it is possible that reentrancy vulnerability is caused by a problem in the design processing order. Thus, a remittance before the state of storage changes can cause a malicious attacker to create a new contract through the vulnerability



(a) The information of the deployed contracts.



(b) The information of the submited transactions.

FIGURE 9. Testing and monitoring smart contracts with Ganache GUI.

to steal the Ethereum in the victim contract. In June 2016, a German startup, the DAO, was stolen from the then-current US$50 million Ethereum. Given that the remittance process is involved, reentrancy should be monitored.

A design pattern called checks–effects–interactions pattern solves this problem. First, in the checks phase, the designer must first determine whether prerequisite conditions are satisfied, such as the use of the require() function. The second phase of effects updates the status in the contract. Finally, the interactions stage uses remittance instructions or interacts with other contracts or accounts. In this research context, only VerifiedSite review() function uses the remittance-related function. This mode was to solve the reentrancy problem. Referring to Algorithm 2, the statement on line 2 first checks if the situation can be executed, the third to eighth line changes the state in the contract, and the line 9 performs the remittance action.

B. TRANSACTION–ORDERING DEPENDENCE

The transaction–ordering–dependence (TOD) problem was considered. When executing a contract function, the usage of the gas can affect the order in which the transaction is written. In the context of this study, using a control from the application side appears to be a more practical solution. Because users were not allowed upload verification results in VerifiedSite contract, it was required to be reverified to be able to do it again. The verification can be performed again if more than one person submits the verified transaction at the same time. Only one person can write and receive feedback. No settings are available for role access in the current contract. The current implementation is conducted on a test environment on the private chain of the Ethereum platform. The access order must be controlled by the distributed app of Fig. 4.

C. OVERFLOW

In addition, when using the smart contract function, if excessive information is transmitted, then it can cause the gas to exceed the gas limit and the function cannot be successfully executed. Such disadvantages are common in using dynamic length variables such as string or array operations. In the context of this study, the string parameter of all functions was changed to a finite-length bytes32 store to avoid such problems. Limiting strings to 32 bits limits the length of the full domain name that can be stored.

D. MISHANDLED EXCEPTIONS

If an exception occurs when calling a function in Ethereum, then the contract must be aborted and the recovery state returns FALSE. However, an abnormal behavior in the design can occur that is not directly transmitted back to the user. This situation is common in calling functions between contracts, but the contract function in this study is not linked, so the effect is nonsignificant.

E. TIMESTAMP DEPENDENCY

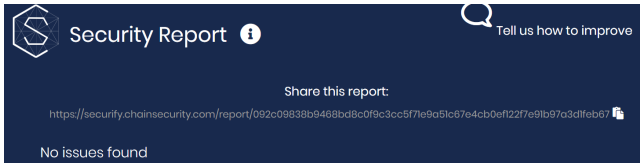
In smart contract design, block.timestamp or now is often used to obtain the timestamp of the block. When using these numbers for calculations, the miner has a certain level of ability to master the write time. For instance, to calculate a random draw through block.timestamp given that the miner has the ability to dig out the block at a specific time, it can be rewarded by participating in the draw. In the context of this study, although the block.timestamp was used, only the time for the user to verify the webpage was stored, and it was not used in the calculation.

F. CRYPTOLOJACKING LIFECYCLE

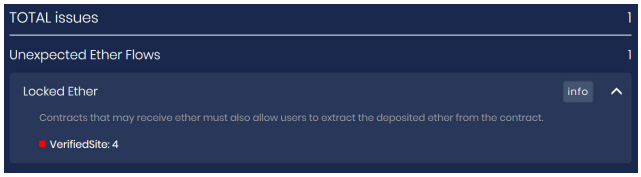
According to [3], one-third of cryptojacking samples disappeared within 15 days, and updates were common. Considering such sample activity, this study sets the certification website VerifiedSite's certification period to 1 day, unless the website manager uses requestReview, other users must wait for 1 day before submitting other updates.

G. DISCUSSION

According to [20], SECURIFY is a security scanner of Ethereum smart contracts. The contract bytecode is first converted into their own custom language, and then compared with a validation module to verify whether its semantics are satisfied, with the security report is generated. Fig. 10(a) and Fig. 10(b) are security analysis reports for our proposed SiteManagement and VerifiedSite contracts. There is nothing wrong in Fig. 10(a). However, the security report in Fig. 10(b) showed that VerifiedSite contract only has a minor security concern. This concern describes that our



(a) Safety analysis report for the SiteManagement contract



(b) Safety analysis report for the VerifiedSite contract

FIGURE 10. Contracts safety analysis reports.

TABLE 5. Studies of smart contract security analysis comparison.

Authors	Reentrancy	TOD	Mishandled	Timestamp	Overflow
[18]	✓	✓	✓	✓	
[19]					
[20]					
This study	✓	✓	✓	✓	✓

VerifiedSite contract may not contain deposit function to let user call deposit function and withdraw the deposited ether. However, we had the withdraw function in source code to let user calling already. Therefore, we believe that this minor concern in this report may not cause further serious problems in our environment.

Table 5 summarizes the security analysis comparisons of research related to smart contracts. Hasan and Salah [22] studied the use of smart contracts to establish proof of delivery of digital assets, and the content of the smart contract security analysis contained the security issues of multiple smart contracts. However, they did not analyze overflow. Hasan and Salah [22] used the smart contract to certify the authenticity of the original and secondary author's films against the fraudulent deep fake films. They analyzed the security of blockchains but did not analyze the security issues that smart contracts are likely to encounter. Cruz *et al.* [24] studied a mechanism for role access control. That is, services are provided between the organizations according to the assigned roles. Smart contracts and blockchain technology are used as common infrastructure to represent the trust and recognition relationships that are essential in RBAC.

In the transaction gas of the issuance contract, the information of the CoinGecko website on May 14, 2019, one-unit ETH denomination is calculated at US$207.98. The user of Table 6 consumed the most when the first VS contract was established, whereas the website administrator requestReview consumed the least.

TABLE 6. Function transaction fee comparison.

Contract	Function	Gas Consumption	To USD
VS	Constructor	1076174	$0.67153
VS	requestReview	103354	$0.0645
VS	review	242924	$0.15159
SM	registerSite	227776	$0.14213
SM	unregisterSite	173414	$0.1082

However, limited to the scale of the experiment, the smart contract of the research department did not undergo extensive stress testing. When too much data is accessed by the contract, the result is unknown. This program can be added to other web authentication services if it is effective. However, the large-scale testing of smart contracts has not yet been achieved in the context of this study.

A solution for borrowing user's resources by anonymous in terms of web mining programs was presented. The solution involved adjusting the webpage mining service provider, so that when the user's resources are intentional used by others for mining, the mining reward feedback are obtained. A web browsing environment with no ads, reasonable use of system resources, and user feedback was created. However, there is a concern about malicious users may continue to use detection programs to send messages to Ethereum to establish more contracts or authentication, and try to obtain more token rewards. A possible solution maybe inherited from [25]. A status of revalidation maybe set only after the website has been detected for an estimated period of time, or classify users according to its credit state such as Credible, Normal, Excepted and Invalid level [25] to prevent intentional user. This idea may leave to researcher for further investigation in the future.

V. CONCLUSION

A new type of attack, called cryptojacking, has emerged in the online world. This attack uses the computer resources of a victim to obtain mining rewards without user confirmation. To resolve this problem, a storage verification architecture based on smart contracts was proposed in the study. The decentralized system of blockchain enables users to identify and participate in verification of mining sites through Ethereum smart contracts. Furthermore, we designed and proposed two smart contracts, namely SiteManagement and VerifiedSite, to solve the certification problem of web mining, and analyzed some security issues of the contract such as Reentrancy and TOD. Currently, this research is focused on building decentralized applications through smart contracts in the Ethereum blockchain, but in the future, it can be extended and integrated with Hyperledger or ENS (Ethereum Name Service) system, or establish a viable business model or system to replace web advertising revenue and so on. In addition, the code is also public on github.com: {https://github.com/bar1eycool/webminingver}.

REFERENCES

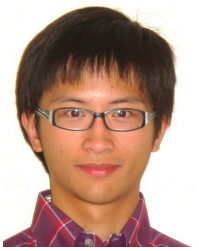
[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: bitcoin.org/bitcoin.pdf

[2] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based Cryptojacking," in *Proc. EEE Eur. Symposium Secur. Privacy Workshops (EuroS&PW)*, London, U.K., Apr. 2018, pp. 58–66. [Online]. Available: https://arxiv.org/abs/1803.02887v1
[3] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, M. Yang, Y. Zhang, Z. Qian, and H. Duan, "How you get shot in the back: A systematical study about cryptojacking in the real world," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Jan. 2018, pp. 1701–1713.
[4] L. Catuogno, C. Galdi, and N. Pasquino, "An effective methodology for measuring software resource usage," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 10, pp. 2487–2494, Oct. 2018.
[5] B. Marr. (2018). *A Very Brief History Of Blockchain Technology Everyone Should Read*. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/02/16/a-very-brief-history-of-blockchain-technology-everyone-should-read/#1ea267f97bc4
[6] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.
[7] A. Chaer, K. Salah, C. Lima, P. P. Ray, and T. Sheltami, "Blockchain for 5G: Opportunities and challenges," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.
[8] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2018, doi: 10.1109/COMST.2016.2535718.
[9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014
[10] F. Wessling, C. Ehmke, M. Hesenius, and V. Gruhn, "How much blockchain do you need?: Towards a concept for building hybrid DApp architectures," in *Proc. 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain WETSEB*, May 2018, pp. 44–47.
[11] K. Wust and A. Gervais, "Do you need a blockchain?" in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2018, pp. 45–54.
[12] Monero. (2014). *Monero—Secure, Private, Untracable*. [Online]. Available: https://getmonero.org/
[13] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of Monero's blockchain," in *LeaPS: Learning-Based Proactive Security Auditing for Clouds*, 2017, pp. 153–173.
[14] S. Gallagher. (2017). *Researchers say WannaCry operator moved bitcoins to 'untraceable' Monero | Ars Technica*. [Online]. Available: https://arstechnica.com/gadgets/2017/08/researchers-say-wannacry-operator-moved-bitcoins-to-untraceable-monero/
[15] K. Rooney. (2018). *1.1 Billion in Cryptocurrency Has Been Stolen This Year, and it Was Apparently Easy to do*. CNBC. [Online]. Available: https://www.cnbc.com/2018/06/07/1-point-1b-in-cryptocurrency-was-stolen-this-year-and-it-was-easy-to-do.html
[16] D. Carlin, P. OrKane, S. Sezer, and J. Burgess, "Detecting cryptomining using dynamic analysis," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.
[17] go-ethereum. (2019). *Ethereum/Go-Ethereum: Official Go Implementation of the Ethereum Protocol*. [Online]. Available: https://github.com/ethereum/go-ethereum
[18] remix-ide. (2019). *Ethereum/Remix-Ide: Browser-Only Solidity IDE and Runtime Environment*. [Online]. Available: https://github.com/ethereum/remix-ide
[19] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 254–269.
[20] H. Li and D. Han, "EduRSS: A blockchain-based educational records secure storage and sharing scheme," *IEEE Access*, vol. 7, pp. 179273–179289, 2019.
[21] S. Wang, Y. Wang, and Y. Zhang, "Blockchain-based fair payment protocol for deduplication cloud storage system," *IEEE Access*, vol. 7, pp. 127652–127668, 2019.
[22] H. R. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018, doi: 10.1109/ACCESS.2018.2876971.
[23] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41596–41606, 2019, doi: 10.1109/ACCESS.2019.2905689.
[24] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240–12251, 2018, doi: 10.1109/ACCESS.2018.2812844.
[25] Y. Wang, S. Cai, C. Lin, Z. Chen, T. Wang, Z. Gao, and C. Zhou, "Study of Blockchains's consensus mechanism based on credit," *IEEE Access*, vol. 7, pp. 10224–10231, 2019.

DONG-HER SHIH received the Ph.D. degree in electrical engineering from National Cheng Kung University, Taiwan, in 1986. He is currently a Senior Professor with the Department of Information Management, National Yunlin University of Science and Technology, Douliu, Yunlin, Taiwan. He has published over 70 journal articles. His current research interests include data mining, information security, block chain, and big data. He is an Associate Editor of the *International Journal of Mobile Communication*.

TING-WEI WU received the M.S. degree from the Department of Information Management, National Yunlin University of Science and Technology, Taiwan, in 2015, where he is currently pursuing the Ph.D. degree. His current research interests include data mining and computer security.

TZU-HSIN HSU received the M.S. degree from the Department of Information Management, National Yunlin University of Science and Technology, Douliu, Yunlin, Taiwan, in 2019. His major research interests include computer security and block chain management.



PO-YUAN SHIH received the M.S. degree from the Department of Finance, National Formosa University, Taiwan, in 2015. He is currently pursuing the Ph.D. degree with the Department of Finance, National Yunlin University of Science and Technology, Taiwan. His major research interests include data mining, FinTech, and marketing.

DAVID C. YEN is currently a Professor with Texas Southern University. He is active in research. He has published books and articles which have appeared in *ACM Transaction of MIS*, *Decision Support Systems*, *Information & Management*, *Decision Sciences*, the *International Journal of Electronic Commerce*, *ACM SIG Database*, *Information Sciences*, *Communications of the ACM*, *Government Information Quarterly*, the IEEE IT Professionals, *Information Society*, *Omega*, the *International Journal of Organizational Computing and Electronic Commerce*, and *Communications of AIS*. His research interests include e-government, mobile commerce, medical information systems, and IT auditing/governance.