Crypto-jacking Threat Detection Based on Blockchain Framework and Deception Techniques

Haissam Badih, Yasamin Alagrash
Department of Computer Science And Engineering
Oakland University
Rochester, Michigan 48309, USA.
{hbadih, yhalagrash}@oakland.edu

***Abstract*—The advances in malware attacks have taken place recently, and a robust security solution is required. Most traditional security approaches, including those proposed within the hybrid approach of malware detection, are no longer effective for detecting criminal cyber-attack strategies. In this paper, we introduced a novel approach to specifically detect malware that injected webcam protocols. The approach proposed a security model to address crypto-jacking as a threat to security in the blockchain.**

The current blockchain framework lacks the capacity for the identification of miners and nodes attacked by crypto-jacking malware. Hence, this approach is to ensure that all the nodes on the blockchain are secured, while also ensuring greater safety for the miners. The present approach requires that the identity of the miner is known, and what the miner's crypto-jacking did, which constitute major blockchain issues. The proposed novel approach involves injecting an application into each node to detect if an unusual process is taking place when the actual miner does not have access to the system.

Since the application inserted will detect the highest possible phase using the CPU will get the name of the process and give it to the cuckoo. This is the suggested solution can also extend a system to the cuckoo machine that can be used. Confusing the cryptojacker and the block can be stored in the cuckoo, but the outcome will not be returned to the miner jacking the unit. The Cuckoo is going to highlight the significant details that will shape the backbone of the blacklist, for example, the infected internet protocol and the blockchain address.

When the miner's address is on the blockchain blacklist, the miner will not allow any transaction to be received. In this way, a good miner will be protected from crypto-jacking malware or any hacker. The proposed approach was tested against the threat actor and the normal user, and it demonstrated a robust methodology capable of detecting malware in a significant way.

***Index Terms*—Crypto-jacking, cuckoo process, smart contract, webcam security.**

I. Introduction

The intellectual sphere of malware was designed to adapt to the system dynamics that they are compromising. Threat actors who developed malware variants are constantly improving their techniques and innovations to weaken the cyber defense capabilities of their targets. It is therefore important for the cyber-security community to constantly track the activities of new malware strains and find ways to mitigate their effects. Real-time anti-malware strategies must balance the introspection thoroughness with the need to avoid slowing down the performance of the endpoint. Attackers design malware to work in the blind spots that occur as a result of such compromises. However, the malware we possess causes an enormous problem in the attacked system which provides a hint to the user.

Contribution: Since we are using the blockchain framework where all the data are safe and protected, we face a problem where the node can be infected with malware. This malware attack can be a crypto-jacking and can capture the data to hash a transaction in a different user machine or node to compromise the machine's data and I/O devices. Therefore we need to protect the blockchain network nodes from any intruder who can use any network without the owner's permission.

In this paper, we make a threefold contribution:

- We used the Cuckoo sandbox tool which is emphasized the user's behaviors and redirect malware to particular operations within a consistent set of I/O cuckoo units. In the sense of protection of the cuckoo Protocol, we set up a case study of the cuckoo consumer's behavior. Along with the cuckoo Procedures, Cuckoo I/O System Party Protective Devices Approach as a whole to allow malware to hit a cuckoo sandbox machine for immediate identification.
- An expansion of our work basis on the previous work by detection. The malware and directing it to a cuckoo driver capture specific data, such as the IP address and the mining information. We used a filter driver to redirect to the cuckoo driver, designed for sensing when network traffic is a lull. When this happens, it turns any incoming traffic to the cuckoo driver and away from the network interface card [18].
- The operating system kernel the driver for sending and receiving network packets directly to the network interface board's hardware controller. There is a whole stack of drivers that perform various aspects of network traffic transmission and receipt. The filter driver is automatically placed on top of the Miniport driver automatically, therefore, below the rest of the driver stack. All traffic moving in either direction between the Miniport driver and the rest of the driver stack can be seen and handled by the filter operator [22].
- Novelty: Crypto-jacking poses a threat to blockchain security. The blockchain framework lacks the identification of the miners and nodes that crypto-jacking malware has

attacked. We need to make all the nodes in the blockchain safe and make all miners safer. Most importantly, we needed to know who the miner was and blocking the crypto-jacking caused to the miner. This is a significant blockchain problem. Our novel idea is to inject an application into each node that will detect if an odd process occurs while the actual miner has not accessed the network.

The inserted application finds the unusual behaviors which define the block header parameters that the crypto-jacking and the IP address need to produce. We can submit the method to the cuckoo device that can deceive the crypto-jacking, and the block can be stored in the cuckoo that will not return the result to the miner jacking the system. (Therefore), the cuckoo can give the information to the attacked system; the attacked system will publish it in the blacklist and list the IP in it so that when a transaction is needed to be tacked by a miner.

The blockchain will check the list to verify if the miner's IP address is on the list. When the miner's IP resides are on the blockchain blacklist, they will not allow any transaction to be picked. In this way, through crypto-jacking, the victorious miner can be defended from any hacker. The search will be open to all nodes in the peer-to-peer network in a smart contract [10].

Novelty: We use a filter driver designed for sensing to redirect to the cuckoo driver. When the traffic on the network is minimal; if this occurs, it redirects all incoming traffic to the cuckoo driver and away from the network interface card [1]. The OS kernel has a cuckoo driver to send and receive network packets directly to the network interface board's hardware controller. There is a whole stack of drivers conducting different aspects of the transmission and reception of network traffic. The filter driver is located automatically at the top of the cuckoo driver and, therefore, below the driver stack. The user of the filter can be seen and treated.

We developed an approach by adding method to cuckoo in a virtual machine (VM) on a device that can trick a crypto-jacking, and a block can be stored in a cuckoo VM that would not return the result to a miner jacking the machine. As a result, the cuckoo will send the details to the attacked system. Then the attacked one can publish it in the blacklist and list the blockchain's IP address so that every transaction needs to be tapped by the miner; the blockchain will check the list if the miner's IP and blockchain address are on the list. When the miner/node IP is on the blockchain blacklist, the node or miner will not allow any transaction to be selected. This way, by crypto-jacking, the victorious miner defends against any hacker. The search will be made open to all the nodes on the peer-to-peer network under a smart contract [10].

Organization. The remaining components of this paper are organized as follows: A related work was discussed in section III. In section IV we discuss the Blockchain and smart contracts. In section V we discussed the Cuckoo Process and cyber-deception. In section VI, we discussed Protecting Blockchain nodes from Crypto-jacking. In section VII we discuss the Novel Malware Detection Protocol. In section VIII we discussed Testing and Validation. In section X, we summarized our work and concluded this paper.

II. Hash function, blockchain Address in Proof-Of-Work

A digital transaction comprises a set of key pairs, each with a private key and a public key. The private key (k) is an arbitrary number. We use the private key to create a public key with elliptical curve multiplication, a one-way cryptographic feature (K). The public key (K) is used to generate the bitcoin address using a one-way cryptographical hash function (A). The elliptic curve is used to measure the public key from the private key, which is irreversible:

$$K = k \times G(1)$$

where k is a private key, G is a constant point called the generator point, and K the corresponding public pin code is. The reverse procedure, known as the "finding of the discrete logarithm" - k calculating if you know K - is as tricky as any possible K values - i.e., the brute force search - can be checked. Blockchain uses a particular elliptical curve and a set of mathematical constants, specified by the National Institute of Norms and Technology Standards, SectP256k1 (NIST). The curve secp256k1 is set according to the function generating an elliptical curve:

$$y^2 modp = (x^3 + 7) modp(2)$$

The mod p (modulo prime number p) indicates that this curve is over a finite field of prime order p,

We presented with a mathematical description of the hash function used in blockchain works. The proof-of-work is a mathematical problem, the purpose of which is to create a link between two blocks. This link will be made within the header of the second block. Someone who is trying to work out a proof-of-work is called a miner. Let's consider two blocks, Bprev and B, and a number called bits and b. b estimates how difficult the proof-of-work is from b, and the target number can be directly computed. This target is a 64-digit hexadecimal number with multiple 0 for its left-digits, e.g.: (0000000000000000000021047879c065745de75 af6dcd473556dced2bcedd4qa71).

Assuming that the hash of the previous block is known, H(Bprev); we'll see how to define the hash soon after. It's a block. Solving the proof-of-work for block B, Known as mining block B, this amounts to finding a number, It was called the "nonce" such as:

$$H(H(Bprev) \oplus b \oplus timestamp(t) \oplus RH(B) \oplus nonce) \leq target(3)$$

Where: $\oplus$ denotes an operation of concatenation; Timestamp(t) is the current time up to seconds. Since the block hash is recursively defined following the above procedure (we assumed that H(Bprev) was already known), we need an initialization: if B is the first block, there is no previous block, so H(Bprev) is a mere convention.

III. Related Works

As one of the most disruptive digital innovations in recent years, Cryptocurrency and the blockchain technology that drives it has been attracting a lot of interest. Blockchain technology offers tremendous potential for more transparent, accountable, and efficient means of storing government data and managing transactions. However, before the system can be scaled, there exist many challenges to overcome. Legal frameworks require reform to control digital currency markets and harness blockchain technology's full potential.

By integrating cuckoo processes, our work addresses blockchain challenges. The cuckoo will be used here as a trusted method to classify the invader. Many works have been suggested in many applications to leverage blockchain technology [14].

Koteska et al. To understand the contribution of the current research on the quality of the implementation of Blockchain, we examined the current quality issues in the implementation of Blockchain and identified the quality attributes of blockchain. Despite that, this topic is still immature. The results indicated that in terms of scalability, latency, throughput, cost-efficiency, authentication, privacy, and security, etc., the blockchain implementation needs to be improved [17]. Our job is special in that it provides a new tool and integration to block any invader using malware residing on a computer and was intended for attackers to access it.

This report explores many paths for future studies to foster comprehensive blockchain work that answers meaningful questions. It shows where research can benefit from multidisciplinary collaborations given the wide range of open-ended questions, and presents data sources as starting points for empirical research [3]. In the past, these consideration were ignored, but in our research, we identify the problem and establish the solution for the blockchain.

Vitalik Buterin, Ethereum's founder, a network that aims to go beyond Bitcoin to create a decentralized business blockchain, came up with a different approach. The author maintained that privately managed technology could be quicker, less costly, and better designed to enable faster interventions when repairs are needed. Nevertheless, it must be added that he acknowledged such centralized technology management will limit access rights [4]. In our work, we set out to solve the malware issues aimed at protecting miners and users.

Hong et al. provides the first systematic study of the size and effect of cryptojacking attacks. To support the automatic detection of malicious behaviors, They built CMTracker, which outperforms state-of-the-art detectors with two behavior-based runtime profilers. They've gathered 2,770 malicious samples from 853,936 popular web pages, and there subset testing manual shows that they are all true positives [19]. This paper only did studies but we did identify the problem and we introduce a solution with a real prove through our application.

Marchetto, Victor. examines crypto-jacking malware samples using both static and dynamic methods and address their potential impact on services running on enterprise servers and operating systems, including critical infrastructure industries. The paper also lays out methods for how to identify and protect against these threats [20]. This work examines the impact of crypto-jacking and shows the impact in these industries. We show the problem and offer a solution supported by an application to do the work.

IV. Blockchain Mining and Consensus

Digital cash was designed long before the emergence of the blockchain. In a central server system that was trusted to remove duplicate Expenditure [5]. Despite significant cryptographic advances, the lack of continuity between centralisation, anonymity and the prevention of double spending inevitably brings into question the viability of this new type of currency.

In 2008, Bitcoin gained recognition on the global marketplace by replacing the signature of the central server with a consensus system based on work evidence [6]. The novelty and enhancement of Chaum's conceptual experiment is the decentralized nature of the payment system created by blockchain technology, leading to a new age that stretches beyond global payments to corporate governance, social institutions, democratic participation and the functioning of capital markets [7].

Bitcoin was the first increase in cryptocurrencies, built based on blockchain's revolutionary technology, which previously lacked a decentralized ledger. The Bitcoin blockchain does not permit conditions to be imposed in a new block for finishing a transaction, as it includes only knowledge about the process itself. Nonetheless, the emergence of technology was the reason why smart contracts were created. Later on, the Ethereum blockchain platform allowed smart contracts to be used in operation.

There is no centralized authority for the blockchain network, as it is the real key to a decentralized system. The information in it is open to anyone to see because it is a transparent and permanent ledger. Therefore, by its very definition, everything on the blockchain is about transparency and everyone involved is responsible for their actions [15]. The blockchain has no transaction costs and is a simple but intelligent way of automatically and securely passing information from one system to the other.

The mechanism is initiated by one party to a block transaction. This one,Thousands are testing blocks, maybe millions of computers scattered around. It's the net. The verified block is connected to the chain and stored over the net. It's only making a single record, but also a single record with a specific past. Falsifying a single record would mean, in millions of cases, forging a record, the whole chain.

Blockchain technology guarantees that the dual-spending problem is addressed. The help of public-key cryptography gives each agent a private key (Maintained as a password) and a public key, exchanged with all other officers. When the future owner of the coin (or digital tokens) sends a public key to the coin. The original owner, the transaction is started. The

digital signature of a hash pushes the It's coins. Public keys are cryptographically generated addresses that are stored in the blockchain, guy.

Each coin is associated with an address and a crypto-economic exchange. It's just a move of coins from one address to another. Pilkington et al, Ghost Stratégically, the blockchain's striking characteristic are that the public keys are never. It is connected to the identity of the real world [8]. Although traceable, there are transactions. Allowed without the disclosure of one's identity; this is a big difference with the fiat currency transactions connected, except non-traceable cash transactions to particular economic agents of legal personality, whether human or physical morality.

Nick Szabo introduced this idea in 1994 and established a smart contract "a computerized transaction protocol is executing contract terms" [11]. Smart Contracts are scripts that are stored on the blockchain within the blockchain. History. For relational database management systems, they can be considered Slowly similar to the stored procedures [12]. They've got a particular address as They're living on the chain. When debating a deal, we are building a smart contract.

It then runs independently and automatically on each node in the network. According to the data involved in the transaction's triggering, in the stated case, the way. Smart contracts allow us to have calculations for general purposes. It was on the chain. However, when it comes to handling data-driven interactions [12] between network entities, this is where they excel.

Furthermore, when they are tasked with managing data-driven interactions [13] between network entities, they also excel. Imagine a blockchain network including Alice, Bob, and Carol, and exchange of Type X and Y digital assets. Bob deploys a smart network contract that defines: (a) a "deposit" function that allows him to deposit X amount into the contract, (b) a "trade" function that returns 1 X amount (from the contract's deposits) for each 5 Y amount he earns, and (c) a "withdrawal" function that allows Bob to withdraw all the assets held by the contract, as shown in Figure 1. We are looking at creating a smart contract in our work which can create a black list and add the IP address to the black list. When the IP address adds to the list, in the smart contract, we can create a logic to search and compare every IP in the black list. If the IP was found in the list, we can exclude any transaction from the node type and add the transaction to a block and avoid creating and transmitting any block into the ledger and adding it to the blockchain.

V. Cuckoo Process and Cyber-Deception

We suggest threatening actors who can deliberately monitor user space processes and potentially leak information to users via a Skype call. It is worth noting that this concept is not completely fresh, as it had been described. It's the word pit. A pit, a cuckoo piece of data that only exists to signify a malicious entry. Take for example, a database of bank client information. In order to preserve customer protection, the bank needs to be able to monitor access to a bank records [9].

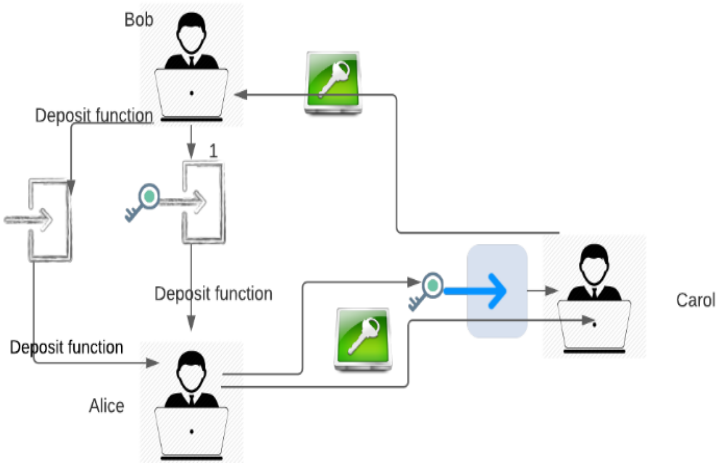

Fig. 1. Blockchain P2P transaction

In the field of network security, the notion of cuckoo systems is not new. Cuckoo Sandbox began as a Google Summer Code project in the Honeynet project in 2010. It was initially designed and built by Claudio "nex" Guarnieri, who is still the leading developer and lead developer [21]. The first beta update was released in Feb after initial work during the summer of 2010.

In one of the works, they present a TPM application protocol that detects a particular person in the middle attack where the adversary captures and replaces a legitimate computing platform with an imposter that forwards the challenge of platform authentication to the captive via a high-speed data link. This revised Cuckoo attack allows the imposter to satisfy the user's query of platform integrity by tricking the user into disclosing sensitive information to the imposter. Their protocol uses a typical smart card to verify the boot platform's integrity through TPM quote requests and verify TPM proximity by measuring TPM tick-stamp times required to respond to quotes [22].

Because we're operating in blockchain, we know the data is well-secure in the context of the blockchain. In any way, however, the blockchain may be Attacked by crypto-jacking malware that used to be restricted to download cryptocurrency program that is secretly mining unknowingly. When the machine is running, we can use the filter driver if we don't have access to our user. The filter driver supposes the machine was not in operation during times when it was not in use. Action coming from the desktop machine and designed to give a signal to the cuckoo machine that was going to start the operation. Therefore, we can redirect the crypto-jacking to the cuckoo system, deceive the malware actor, and quarantine the mining process in the cuckoo sandbox method.

VI. Protecting Blockchain Nodes from Crypto-Jacking

We describe the methodology of our work in this section. The logical approach and methods will be discussed. Cryp-

tocurrency, like Bitcoin, makes people rich and opens up something new that can use computers, resources, and power to help acquire money. Cryptocurrency is a former digital currency that exists only digitally instead of having printed, paper money. Units of money are made through a process called mining which uses the computer processing power to perform a complex calculation.

What happens in this mathematical calculation is complicated and it ensures that transactions between people using the cryptocurrency are documented safely and the accurate record of the ordering in which the transaction was carried out is identified and double expenditure is prevented. The user who completes this calculation will receive a reward of a small amount of the cryptocurrency, and this is how money is received. However, when used for malicious purposes, it is illegal to accept, but the cryptocurrency itself is legal. Since cryptocurrency undertakes complex calculations, it will harness the computer's power and network; it is very beneficial, however, from the miner's side, it is very expensive.

Therefore, crypto-jacking happens because the intruder wants to use someone else's machine and energy to generate better income. Therefore, they use the malware to measure the hash signature on another device, get the result to their computer, and transmit the new block to blockchain cryptocurrency. Next, they receive compensation from the blockchain cryptocurrency without using their electricity, resulting in less money to pay with more profit to gain. The crypto-jacking in the blockchain is a security threat. We need to make all the nodes in the blockchain safe in this matter and also make all miners safer. What we need to learn is who the miner is and what block the crypto-jacking miner has made.

Thus, this a big issue in the blockchain. Our novel idea is to inject an application into each node that will detect if an odd process is occurring while the actual miner has not accessed the system. The inserted application finds the unusual behavior that can define the block header parameters that the crypto-jacking and the IP address need to produce. Then, we can submit the method to the cuckoo device which will deceive the crypto-jacker and the block can be stored in the cuckoo that will not return the result to the miner jacking the system.

Therefore the cuckoo sandbox will give the information to the attacked system, the attacked system will publish it in the blacklist, and the IP address and the blockchain address will be entered in this list. Therefore, any transaction that will have to be tackled by a miner the blockchain will search the list to verify if the miner's IP address is contained therein. If miner's IP exists in the blockchain list, no transaction will be chosen by the miner. It is in this way that the crypto-jacking protects the good miner from any hacker. This will be done using a smart contract. See Figure 2 and the displayed device consumer application in Figure 3.

Building user activity cuckoo sandbox. In this case, cuckoo dodger closely watches to determine the system's IP addresses with which the compromised computer interacts over the network. Instrumentation of the network interface alone cannot make it appear as if the compromised system

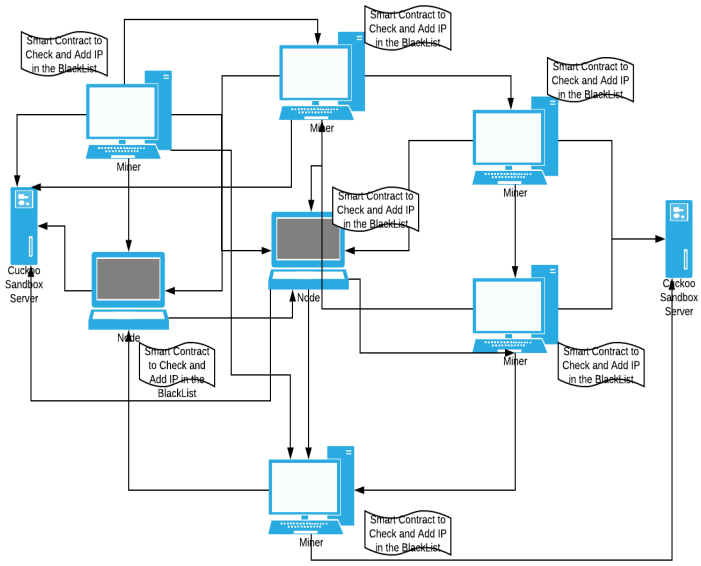

Fig. 2. Secure user in secure system

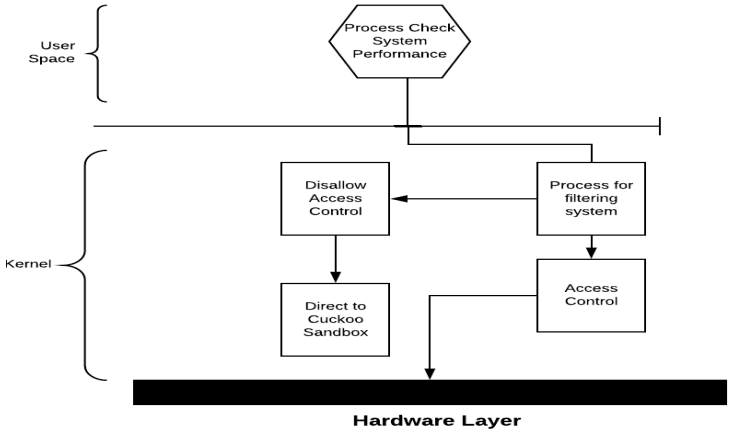

Fig. 3. System User

interacts with cuckoo sandbox. We need a mechanism in the user space, like all the network applications, to send synthetic network traffic through the driver stack, although it cannot be regarded as a real operation. This is mainly due to the high overhead of real processes, and also due to the visual appearance on the display of a real process, which definitely confuses the user and can also result in action on its graphical user interface.

In their previous work, they explored cuckoo processes [18]. We extend the principle of cuckoo process in this research to produce observable, cuckoo sandbox user activity that sends the synthetic network traffic we need for our cuckoo tactics. The authors found two main factors related to cuckoo processes, namely the dynamics of life and the efficiency of run-time. Through instrumenting with the data structures

used by the OS kernel to handle processes and threads, they make a cuckoo process noticeable. Consequently, in their performance, the task manager method, the task-list command, and the ps command all showed an entry for the cuckoo operation.

A cuckoo process' run-time performance dynamics are generated by instrumenting with the output counter data structures in the OS kernel. A performance data provider in the OS kernel collects these data structures, which in turn makes performance counter-data available in user space to users, including potential malware. Such performance counter-data relates to multiple components, including the cuckoo sandbox mechanism, network interface and diagnostics of TCP/IP performance.

Integration with the interface of the network. The binary instrumentation module generates a placeholder that is allocated a set of virtual memory addresses to the cuckoo method. The cuckoo process usually does not run and therefore does not use CPUs, physical memory frames, secondary storage, networking, and other resources. The binary instrumentation module places synthetic network packets and a small block of code in the memory of the cuckoo process when the filter driver finds a time window with low or missing network activity and decides to switch to the cuckoo network interface. This can work in TCPREPLAY method [9]

Because the synthetic network packets are already presented in the memory, they are simply sent over the network by the code block. Since it is the currently active cuckoo network interface, the synthetic network packets reach the cuckoo I/O module, which generates replies and sends them back through the driver stack to the cuckoo process. Responses are network packets that also tend to originate in sandbox. The cuckoo process stops at this stage and gives its physical memory frames back to its state of 0-resource usage. The physical network interface is reinstalled by the filter driver.

Impact of the outcome. A network sniffer reveals a two-way communication on the compromised computer between a device and honeypots. These honeypots seem to be active in the network. Once again, they are indistinguishable from the production equipment. To prevent legitimate users from seeing fake network traffic in their network sniffing devices, we rely on a monitor filter driver that eliminates all cuckoo data bound for the monitor before showing those data [16]. As seen in Figure 4.

Most information can be collected by the network emulator. Additionally, some of the details that the emulator requires is the invader's IP address to pass to the output of the process control system. This information is required to identify the invader and to be able to publish the IP address for protection so that all nodes are aware of the illegal work done by the bad node or miner, as shown in Figure 4.

Smart contract process for blacklist. This method tests system performance after obtaining the emulator's IP address and blockchain address that invades the crypto-jacking program. The output of the process control system would submit the required data to a blacklist of smart contracts. This smart

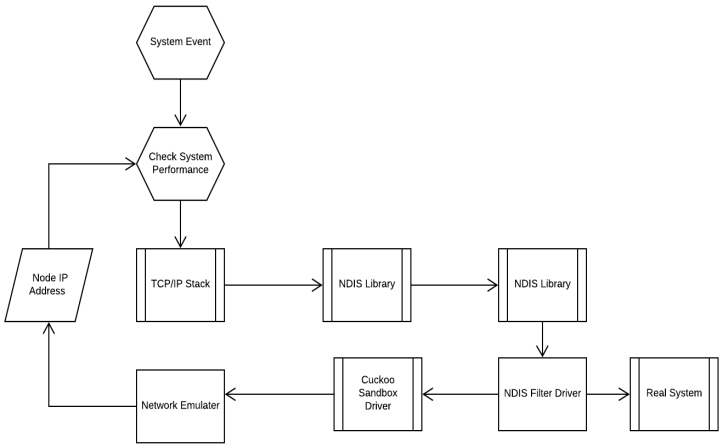

Fig. 4. Cuckoo Sandbox process

contract will get the address of the blockchain and the IP address and add it to the blacklist. This blacklist functions to verify if any node or miner has performed illegal activities, such as using the crypto-jacking and attempting to mine in another network or system. This is because they rely on other schemes to use the resources to reduce costs and make the profit illegal. If this information is saved in the blacklist, it will be forbidden if they need to get any blockchain transactions to block it and post the block to the blockchain. This way, if we have achieved security of good miners and nodes, we won't be allowed to do that, as seen in Figure 5.

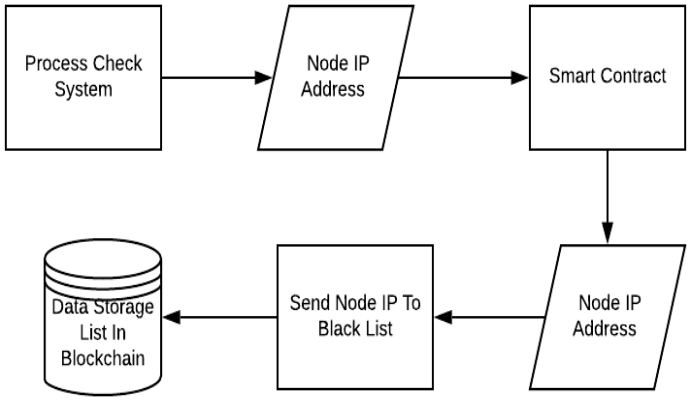

Fig. 5. Smart Contrat Process

What is a smart contract? Smart contracts are computer programs that only operate upon the fulfillment of specific conditions. Simply put, it's a protocol that implements a contract's terms. Like a conventional contract, the conditions and penalties around an arrangement are specified by a smart contract. Unlike traditional contracts, such obligations are executed automatically by a smart contract, often through an

escrow-like account. Also, unlike a conventional contract, they can not be changed once the terms of a smart contract have been documented on the blockchain.

Creating Smart Contracts. The Ethereum Virtual Machine (EVM) offers a decentralized virtual machine for developers to create multiple smart contract applications in Ethereum.Thought of the EVM as a machine that runs all smart contracts globally. Because of this smart contract, our goal will be to use it to define the blacklist in our research and make it available for blockchain transactions to mine a block and publish the block in the blockchain. And the aim of this contract is to make it accessible to all nodes and miners as a protocol in the blockchain as a decentralized application as can be seen in Figure 6.

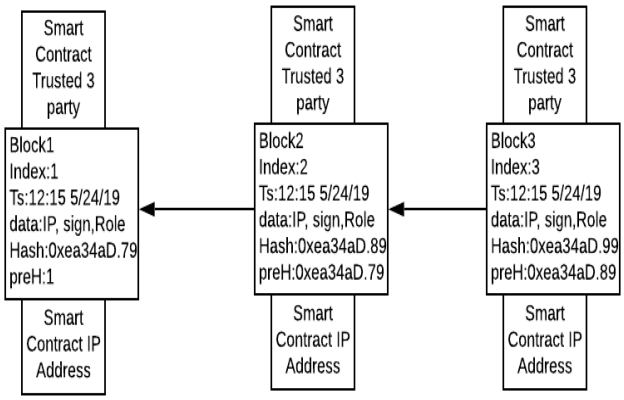

Fig. 6. Smart Contract in Blockchain

Algorithm 1: How to Get the miner invader

Input: Performance and get if system in use
Result: Performance and System in use
// This is code in the algorithm 1 will show how the process will get the performance, and base on the performance and the if system in use we can identify whether the system can reach the control access in the kernel of direct the execution for mining to the cuckoo sandbox emulator.
while *While performance value >30percent* **do**
  if *performance value >30percent And System in use* **then**
    System permited to control access;
  else
    System goes to cuckoo;
  end
end

Algorithm 2: How to add Address of a miner invader into blacklist

Result: Cuckoo capture mine invader
Input: IP and blockchain address.
// This code is intended to contain the invader mechanism in cuckoo, capture IP and blockchain address if the correct IP invader allows us to obtain both addresses and transfer them to a smart contract and then add them to a blacklist
while *Address Invader not null* **do**
  Address received by Agent Process check performance;
  Send address to Smart Contract;
  Smart Contract Add Address to blacklist;
  Exit;
end

VII. Novel Malware Detection Protocol

Before we test our model and recognize any possibility of malware attack we consider this model. As the user working on the computer doing Skype meetings or conferences, we started to test our model in line with what is depicted in Figure 7. Who going to discover the crypto-jacking in web browser? In the point we have two scenarios of solutions.

- Base on our previous work second order detection we have cuckoo I/O assuming bob operate cuckoo I/O, which means is going to discover the crypto-jacking through cuckoo I/O.
- Our scenario Bob doing Skype call using microphone and webcam also cuckoo is activated for every signal of request send from Bob machine.

This work is a continuous to our previous work second order detection. In this paper we do deep investigation based on Cuckoo Security Protocol. We develop an approach as another support for cuckoo based on most common usage scenario like transfer money using blockchain over Skype. Our solution based on the following as shown in Figure 7, we applied our solution to the blockchain technique to check malware possibility in the system. Since we want the system to be secured and we don't want to overload the user's computer, we turn the cuckoo system on and redirect the malware to it. More specifically we trace our model from user request. The request will be through the application called CpuProcessUsage. What this application does is that it will capture every process running in the system with CPU usage percentage. This will then add all the processes name and usage into the dictionary object list with typed string and float as showing in Figure 8. Also we can capture all the URLs from the user infected and transfers them to cuckoo.

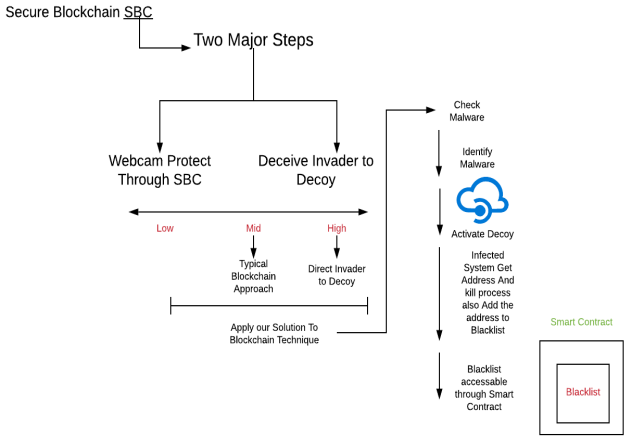

Fig. 7. Malware Scenario.

VIII. Testing And Validation

Based on the sequence of our work we introduced Blockchain's trust-based smart contracts to access a webcam in a network that provides better performance and robust

system security most safely with less overhead. Our prototype provides more secure webcam sessions that are used in Windows Operating System user and kernel mode. The kernel-mode controllable strategy has been developed to record any video based on the user's role that will be authorized by smart contracts. Our Blockchain solution allows the Windows operating system kernel to access the Webcam.

In the second approach, we proposed that cuckoo userspace activity, together with a case study, namely a cuckoo security protocol, involves malware in a series of interactions that lead them to access a cuckoo I/O device, i.e. a cuckoo webcam in this case. In practical terms, cuckoo user space activities are carried out by a coherent set of cuckoo I/O devices and cuckoo processes. This work makes cuckoo user space activity and cuckoo's I/O indistinguishable from their actual counterparts and thus increases the uncertainty of malware operations on a compromised machine to benefit from their detection.

In the current approach, this paper set out to secure blockchain (SBC)in two fundamental major steps:

- The webcam protects the user through the SBC by using the blockchain framework.
- The cuckoo process deceive the invader to cuckoo by using the second order webcam.

As shown in Figure 7, we applied our solution to the blockchain technique to check malware possibility in the system. Since we want the system to be secured and we don't want to overload the user's computer, we turn the cuckoo system on and redirect the malware to it. More specifically we trace our model from user request. The request will be through the application called CpuProcessUsage. What this application does is that it will capture every process running in the system with CPU usage percentage. This will then add all the processes name and usage into the dictionary object list with typed string and float as showing in Figure 8.

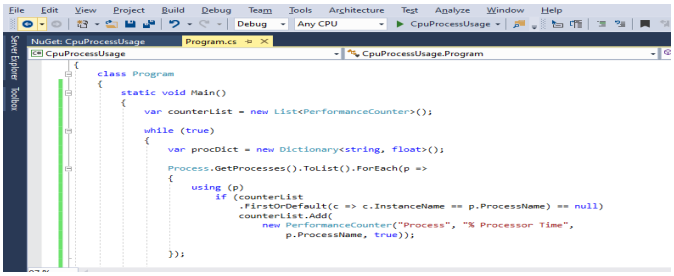

Fig. 8. Cpu Process Usage.

Figure 9 shows the analysis of the event process, in the CPU. After, that we add them to a list base on the performance counter object. The value will be calculated over the base line of the number of the logical CPU's multiply by 100, so this is going to be a calculated over the baseline of more than 100 as seen in Figure 10.

When we get the counter list for performance processes we get the list in descending order and this order will help us to simply get the highest process with CPU usage and get selected as shown in Figure 11 and the result for the process

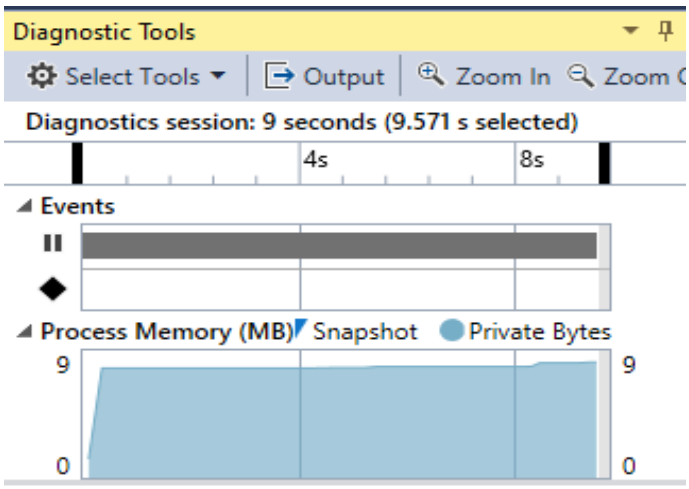

Fig. 9. CPU Usage and analysis.

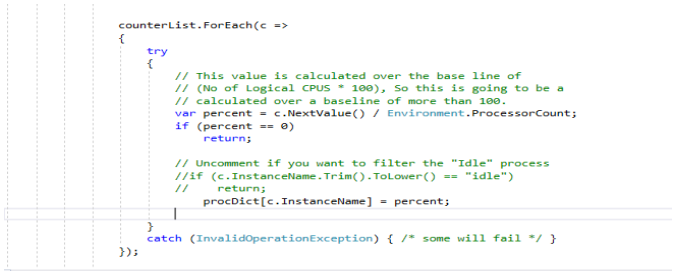

Fig. 10. CPU Processes Usage in the system.

showing the selected process with high CPU is depicted in Figure 12. When the process name is selected we can kill it after capturing the URLs list from the browser and redirecting the list of URLs to cuckoo is depict in Figure 12. The cuckoo will execute the URLs until it captures the infected website. We can run the infected website from cuckoo and deceive the malware and capture it through the static analysis tools. We can get the most important data which are the IP and the blockchain addresses because the malware we are targeting is the crypto-jacking, which is a malware that will run as a mining process to do mining with high calculation process which can exhaust the CPU. Because of that, we redirect the process to cuckoo and we can capture the IP and the blockchain addresses.

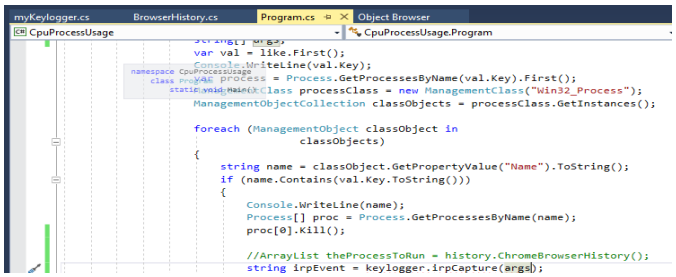

Fig. 11. CPU Process Usage to transfer process to cuckoo.

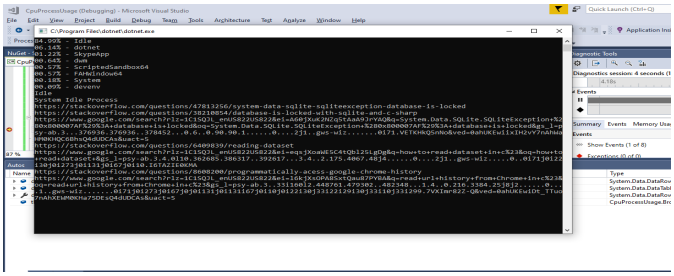

Fig. 12. Check Perform Processes.

Since we have identified the malware process as shown in Figure 12, we can connect to cuckoo. To connect to a Windows cuckoo, we can use A ConnectionOptions to connect to a remote computer with default connection options. For protection, we can use the credential bypassing the IP address username and password, with these parameters we can make remote connections for WMI v1 through the ManagementScope object as showing in figure 11, and then we can run the process from the invoking method in the cuckoo framework. Then cuckoo will do read the list of websites and get the infected website caused the crypto-jacking, through this can get the IP address from the mechanism by using Wireshark or another method that can collect network traffic or static analysis tools.

Wireshark is a free and open-source package analyzer. It is used for network troubleshooting, research, software and communication protocol creation, and education. So we can get the IP address via Wireshark, because the crypto-jacking would allow heavy contact from the IP network to another IP, which is the IP invader because we recognize the IP of our system that we can catch with which we communicate.

Since cuckoo possesses both addresses including the IP and the blockchain, therefore, we want to send the information to the invaded system. The reason is that we want to insert the information into the blacklist. The blacklist will generate through a smart contract and this smart contract is code written in solidity language which can reside in remix Ethereum. Within the Ethereum, a network of thousands of computer processes is used every time a program is used. Contracts written within smart contract-specific programming languages are compiled into bytecodes that can be interpreted and executed by the ethereum virtual machine (EVM) feature. Therefore, we created a smart contract to be inserted in the invader's addresses into the blacklist as depicted in Figure 13.

Now that we have all the applications existing, we create a JavaScript program through which we will be able to access the smart contract and add the bad miner addresses into the blacklist. To be able to do that we use the plugin for blockchain. Since we are using ethereum blockchain we use Web3.js plugin which is a series of libraries that allow you to communicate with a local or remote ethereum node through an HTTP or IPC connection. The JavaScript web3 library communicates with the Ethereum blockchain as depicted in Figure 14.

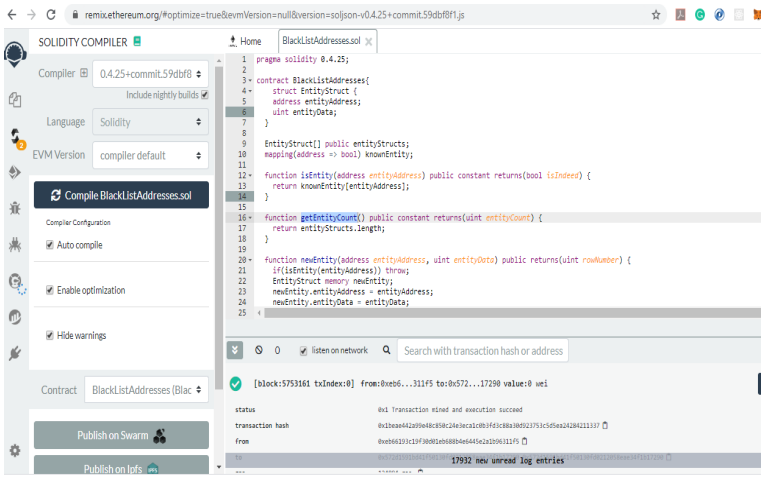

Fig. 13. Blacklist Smart Contract.

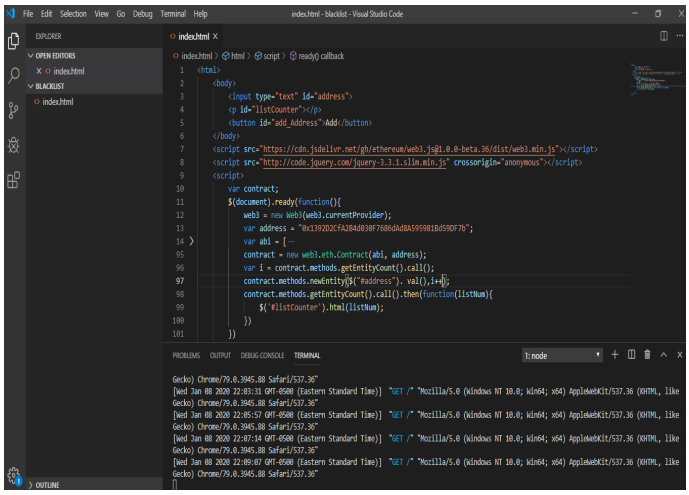

Fig. 14. web3 library communicates with the Ethereum.

IX. Performance Evaluation

We perform a The maximum deviation (D) in the cumulative is measured by the KS test of crypto-jacking scripts to analyze the performance and complexity of their code. Static analysis reveals standard code-specific features that provide a more in-depth insight into information flow to code execution. For static analysis, we collected crypto-jacking scripts from all major crypto-jacking service providers found in our datasets, such as Coinhive, JSEcoin, Crypto-Loot, Hashing, and DeepMiner. We noted that all service providers had unique codes specific to their platform. In other words, the websites using Coinhive 's services had the same JavaScript code template for all of them. As a result, 80 percent of our dataset websites used the same JavaScript template for crypto-jacking. Similarly, all JSEcoin-based websites used the same standard template for mining. However, each service provider's code template was different from each other, which led us to believe that each script had unique static features. With all this in mind, we have been performing a static analysis on the crypto-jacking;

the websites and the results were compared with other standard JavaScript for a baseline comparison.

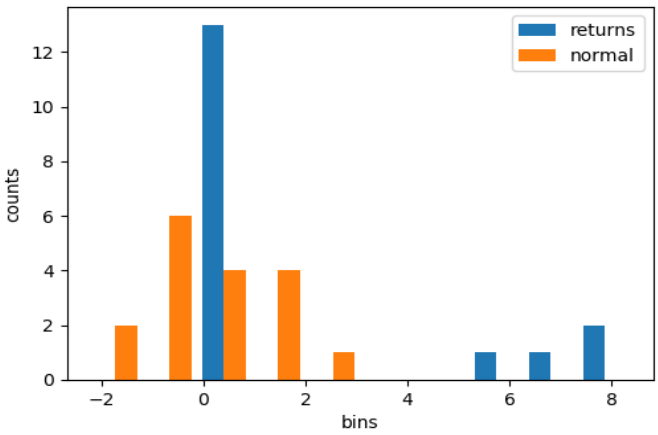

Fig. 15. The maximum deviation (D) in the cumulative is measured by the KS test.

X. Conclusion

This paper demonstrates a cyber-deception insight into the blockchain framework to identify the finger prints of the threat actor activities. Our approach keeps webcam protocol safer and more robust in the blockchain technology. The central component of our approach entailed a novel idea which is to inject into each node an application that can detect if an unusual process is going on while the actual miner does not have access to the system. Since the inserted application detects an unusual behavior that can determine the parameters of the block header that the crypto-jacking needs to create.

We can apply the method to the cuckoo device that can circumvent the crypto-jacking and the block can be stored in the cuckoo that does not return the result to the miner who is jacking the system. The cuckoo will highlight the significant information which is going to be a backbone for the blacklist such as the infected internet protocol and the blockchain address. This information is conveyed to the invaded node. Therefore, the invaded node will get important information and insert them into a blacklist.

Also, any transaction that needs to be tacked by a miner or node the blockchain will check the blacklist through the significant data such as IP and blockchain addresses of the miner is on the list. When node IP and blockchain addresses reside in the blockchain blacklist, the miner will not allow for any transaction to be picked. The solution that we have identified therefore assists in our understanding how to defend the successful miner from any crypto-jacking or hacker. The search will be open to being implemented in all nodes in the peer-to-peer network in a smart contract.

One type of webcam malware is clickjacking, which manipulates the website rendering making it invisible. We are entirely sure that we have resolved the malware as well.

References

[1] J. Rrushi, "Honeypot Evader: Activity-guided Propagation versus Counter-evasion via Decoy OS Activity", In Proceedings of the 14th IEEE International Conference on Malicious and Unwanted Software, Nantucket, Massachusetts, USA, October 2019.

[2] J. Yuill, M. Zappe, D. Denning, and F. Feer. "Honeyfiles: Deceptive Files for Intrusion Detection", IEEE Workshop on Information Assurance, West Point, NY, USA, pp. 116-122, June 2004.

[3] M. Risius, and K. Spohrer, "A Blockchain Research Framework: What We (don't) Know, Where We Go from Here, and How We Will Get There. Business and Information Systems Engineering, 59(6), 385-409." (2017)

[4] Giovanni P. "BLOCKCHAIN: IS SELF-REGULATION SUFFICIENT?."

[5] D. Chaum "Blind signatures for untraceable payments." Advances in cryptology. Springer, Boston, MA, 1983.

[6] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, and P. Wuille "Enabling blockchain innovations with pegged sidechains." URL: http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains 72 (2014).

[7] A. Wright, and P. De Filippi "Decentralized blockchain technology and the rise of lex cryptographia." Available at SSRN 2580664 (2015).

[8] M. Pilkington "11 Blockchain technology: principles and applications." Research handbook on digital transformations 225 (2016).

[9] S. Jajodia, V. Subrahmanian, V. Swarup,and C. Wang "Cyber deception.", Springer, 2016.

[10] Y. Yuan, and F. Y. Wang "Towards blockchain-based intelligent transportation systems." 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2016.

[11] N. Szabo, Smart Contracts. [Online]. Available:http://szabo.best.vwh.net/smart.contracts.htm (1994).

[12] MySQL Reference Manual—Using Stored Routines (Procedures and Functions), accessed on Mar. 15, 2016. [Online]. Available:http://dev.mysql.com/doc/refman/5.7/en/stored-routines.html

[13] Eris Industries Documentation—Smart Contracts, accessed on Mar. 15, 2016. [Online]. Available:"https://docs.erisindustries.com/explainers/smartcontracts/"

[14] C. Stoll, "The Cuckoo's Egg". New York: Doubleday, 1989.

[15] B. Liu, X. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data." In 2017 IEEE International Conference on Web Services (ICWS), pp. 468-475. IEEE, 2017.

[16] J. Rrushi, "Dnic architectural developments for 0-knowledge detection of opc malware," IEEE Transactions on Dependable and Secure Computing, September 2018

[17] B. Koteska, E. Karafiloski, and A. Mishev "Blockchain Implementation Quality Challenges: A Literature." SQAMIA 2017: 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications. 2017.

[18] H. Badih, Y. Alagrash, J. Rrushi A Blockchain and Defensive Deception Co-design for Webcam Spyware Detection. In 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) (pp. 593-600). IEEE. August 2020.

[19] G. Hong, Z. Yang, S. Yang, L. Zhang, and Y. Nan "How you get shot in the back: A systematical study about cryptojacking in the real world." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018.

[20] V. Marchetto "An Investigation of Cryptojacking: Malware Analysis and Defense Strategies." Journal of Strategic Innovation and Sustainability 14.1 (2019): 66-80.

[21] D. Oktavianto Digit, and I. Muhardianto. Cuckoo malware analysis. Packt Publishing Ltd, 2013.

[22] R. A. Fink, A. T. Sherman, A. O. Mitchell, and D. C. Challener "Catching the cuckoo: Verifying tpm proximity using a quote timing side-channel." In International Conference on Trust and Trustworthy Computing, pp. 294-301. Springer, Berlin, Heidelberg, 2011.