UDC 004.681
DOI: 10.15587/1729-4061.2023.273363

A MODEL OF DECOY SYSTEM BASED ON DYNAMIC ATTRIBUTES FOR CYBERCRIME INVESTIGATION

Sviatoslav Vasylyshyn
Postgraduate Student*
Vitalii Susukailo
Corresponding author
Postgraduate Student*
E-mail: vitalii.susukailo@gmail.com
Ivan Opirskyy
Doctor of Technical Sciences*
Yevhenii Kurii
Postgraduate Student*
Ivan Tyshyk
PhD*
*Department of Information Security
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013

The object of research are decoys with dynamic attributes. This paper discusses the impact of decoys involving blockchain technologies on the state of information security of the organization and the process of researching cybercrime. This is important because most cybercrimes are detected after the attacker gains access to sensitive data. Through systematic analysis of the literature focused on assessing the capabilities of decoy and blockchain technologies, this work identifies the main advantages of decoys that utilize blockchain technology. To assess the effectiveness of attacker detection and cybercrime analysis, controlled experiments were conducted using a blockchain-based decoy system that we developed aimed at determining network performance.

As part of the study reported here, a technique is proposed to detect cybercrime using decoys based on blockchain technology. This technique is based on the fact that the attributes of the system change dynamically. Such a technique has made it possible to obtain a system model that solves the task of detecting decoys by intruders. In addition, the developed scheme reduces the load in contrast to the conventional fixed solution.

The results indicate that the response time of services is significantly reduced in the environment of decoys with dynamic attributes. For example, Nginx's response time in a static host is twice as high as dynamic, and an Apache dynamic server can still respond to an intruder's attack even if a static server fails. Therefore, the results reported in the article give grounds to assert the possibility of using the solution in the infrastructure of information systems at the public and private levels

Keywords: decoys, cybercrime, security, analysis, deception, blockchain, Honeypot, Deception, network, cybersecurity

Received date 02.12.2022
Accepted date 05.02.2023
Published date 28.02.2023

How to Cite: *Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. (2023). A model of decoy system based on dynamic attributes for cybercrime investigation. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 6–20. doi: https://doi.org/10.15587/1729-4061.2023.273363*

1. Introduction

Despite the significant efforts that organizations are making to prevent compromise, the reality is that if cybercriminals attack a particular organization, they will find a way to infiltrate the internal infrastructure. It is also important to investigate and correctly document both the cybercrime and the evidence base. Making an attacker think that s/he has access to valuable data is not a new idea in the field of information security. The first Honeypot network was developed back in 1999 as part of the Honeynet project. At that time, the idea was innovative and effective but over the past 20 years, the IT infrastructure of companies has become much more complicated while attackers have gained experience.

Where conventional products seek to respond to a cyberattack and isolate it as soon as possible, Honeypots and next-generation deception systems take a more active stance on protecting information. They detect not the attack but the cybercriminals themselves during their work, which makes it possible to prevent cybercrime, even before it occurs.

Due to the spread of complex attacks on the infrastructure of information systems, in particular the combination of exploits and social engineering, the detection time of an attacker, according to IBM statistics, in 2022 averages 277 days. During this period of time, an attacker can gain access to confidential information of the organization. This poses a threat to the reputation of the organization and in general its existence, and an undetected attacker, having gained access to state secrets, can be a threat to state security. A conventional decoy system can be used to detect an attacker and study his/her behavior, but the conventional decoy system is static and can be easily detected by an attacker, as well as its attributes (configuration files, user lists, software). As part of the study described in this paper, a technique to detect cybercrime using decoys based on blockchain technology is proposed. This technique is based on the fact that the attributes of the system change dynamically. Such a technique made it possible to obtain technology with interchangeable elements of the decoy system, which complicates the possibility of detecting decoys. Also, the developed scheme reduces the load in contrast to the conventional fixed solution. From a practical point of view, the developed system makes it possible to reduce the total response time of information systems services and reduce the load on the network infrastructure of companies. Therefore, it is now quite important to use decoys and other technologies that will direct the attacker to a fake infrastructure node.

Also, in most cases, when an attack is detected, it is correct to stop it immediately. But with the help of deception, organizations have the ability to detect a wide range of suspicious activities that do not depend on known signatures, search the database, or compare templates. This allows deception technology to search for suspicious activity, learn more about the nature of the attack, and better understand the way in which attackers intend to spread.

2. Literature review and problem statement

Honeypot is designed to attract attackers to exhaust attacking resources and to protect the real system. There are new applications of this technology, for example, wireless networks, social networks, or industrial control networks. It can be used for denial of service (DoS), distributed denial of service (DDoS), ransomware, bandwidth attack, and more. In terms of DoS attacks, honeypot was used to mitigate DoS on the Internet of Things (IoT) devices. Comparing with the information of the log library, the system isolates abnormal requests trapped in honeypot and records the data of the source of the attack [1]. For a DDoS attack, a honeypot architecture with an automatic response was proposed [2]. Any suspicious traffic will be forwarded to an isolated honeypot, which further protects the real system. However, in both papers, the systems did not have dynamic properties, which makes them vulnerable to repeated attacks.

In addition, it is worth noting that some honeypot schemes are dynamic. The dynamic property is mainly displayed on the configuration and deployment. The dynamic honeypot circuitry has also been considered and it uses Nmap, P0f, and Snort for active detection and passive recognition of attacks. Honeyd and some very interactive honeypots are used to model the network and redirect the network flow accordingly [3]. The dynamic honeypot engine interacts with the modules mentioned above, dynamically configuring Honeyd, and providing a customizable interface. To simulate a real industrial network in real time (i.e., honeypot is a fictitiously real system), honeypot was dynamically configured, which allocates unused Honeyd cluster IP addresses [4].

Dynamic control of decoys was also presented in study [5]. According to data collected from routers, firewalls, IDS, and honeypot, the honeypot configuration is dynamically adjusted to adapt to the network environment. Another study combined highly interactive honeypot with a low-interactive one. Adaptive honeynet scheme is implemented by modeling some operating systems [6]. The key module of this scheme is the Honeybrid gateway, which contains parts of decision making and redirection. The first is used to capture and transmit certain network traffic in Honeyd. The second aims to redirect the Honeyd stream to a highly interactive software decoy. There are some works on the dynamic deployment of honeypot that offer a honeypot deployment automation scheme [7]. To monitor the network, active and passive network flow detection technologies are used. User configuration information is stored in a database that can serve as a classification criterion for creating a new honeynet network, bandwidth limits, and the target IP range of the network. Honeypot Honeyvers dynamic circuitry is based on machine learning. The network environment is scanned, and the equipment is classified to determine the exact number of honeypots, thus automatically generating information about the configuration and subsequent deployment of honeypots [8]. To solve the problem caused by the uneven deployment of honeynet, a multi-virtual network management architecture is put forward that generates specific honeynet information based on different requests. Individual honeynet is automatically deployed by a set of tools [9].

These dynamic honeypot schemes pretend to fit into the network environment self-adaptively and focus on attacker fraud. However, the location of these software decoys is fixed after determining the configuration or deployment information. With the development of anti-honeypot technology, all these projects are likely to already be found and calculated. Due to the property of transforming the location in the proposed scheme of this work, these dynamic configurations of honeypots differ from others. In the proposed scheme, even if attackers detect honeypot, they cannot find real nodes and users.

Blockchain technology offers great potential for the development of various sectors due to a unique combination of characteristics, such as decentralization, immutability, and transparency [10]. So far, the technology has attracted the most attention thanks to industry news and media about the development of crypto currencies. Examples are Bitcoin, Litecoin, Dash, and Monero, all of which have excellent market capitalization. However, blockchain is not limited to crypto currencies. In industry and the public sector, blockchain-based applications already exist, such as crowdfunding for tracking goods in supply chains [11], authentication [12], and voting services [13]. Many others are under development. The Fraunhofer Institute for Scientific and Technical Trends Analysis (INT) in Germany has published a study [14] showing that blockchain is currently most commonly found in applications used in the financial sector.

The cybersecurity, analytics and detection sector can also use systems built on blockchain technologies for own purposes not only to strengthen the security of existing systems but also to investigate the behavior of attackers and identify patterns of their behavior.

3. The aim and objectives of the study

The aim of our study is to determine the possibility of using a decoy system based on the dynamic attributes of the blockchain. This will increase the efficiency of cybercrime detection and improve system resilience by reducing the load on the network infrastructure and the response time of services.

To accomplish the aim, the following tasks have been set:

– to analyze the problems of using decoys and deceptions to protect data in computer networks;

– to develop a decoy system based on blockchain technology;

– to determine the effectiveness of the decoy system with dynamic attributes of the blockchain system.

4. The study materials and methods

The object of research are decoys with dynamic attributes built on the basis of blockchain technology. The main hypothesis of this study is the possibility of mitigating the risk of detecting a decoy system built on the basis of blockchain technology with dynamic attributes by an attacker. During the development of the decoy system model, it was assumed that a dynamic system should reduce the load on the service components of the system infrastructure by distributing the load between its elements. Also, it was assumed that if the response time of the service of the proposed system is reduced, the likelihood of detecting the decoy would become low.

During this study, a systematic analysis of the literature was carried out to determine the optimal components of the decoy system based on blockchain technology. A prototype decoy system based on blockchain technology was also designed and a controlled experiment was conducted to test the load on the developed decoy system.

This study was conducted using the following software:

– Nginx web server (USA), Apache web server (USA), MySQL database (USA), FTP server VsFTPd (USA) – used to study the load on the decoy systems;

– Iperf (USA) was used as a network performance measurement service;

– netsniff-ng (USA) – network analyzer;

– Jmeter(USA) served as a tool for load testing;

– Hping (USA) was used to generate open-source network packets.

5. Results of the study of system decoys based on dynamic attributes

5. 1. Issues of using decoys and snags to protect data in computer networks

Honeypot can be considered the first embodiment of Deception technology, and they appeared in the late eighties – early nineties. Honeypot is a network object whose sole purpose is to attract an attacker and be attacked. When Honeypot is attacked, it logs it and saves all the actions of the attacker. In the future, these data help to analyze the path of the attacker. The second goal of Honeypot is to delay the promotion of an attacker by the network, forcing him/her to spend time studying a fake resource [15]. We present the scheme of the Honeypot system in Fig. 1.

Honeypot can be a full-fledged operating system that emulates an employee's workplace or server, or a separate service. Understanding the abilities of intruders is important for building a protection system that can detect them [16, 17].

Let's introduce several ways in which attackers determine the presence of Honeypots:

– if access to the system seems too simple, possibly fake;

– typically, systems connected to the Internet do not have unnecessary ports and services; any deviation from this configuration may indicate a trap;

– if the system still has a default setting, it increases the likelihood of using Honeypot;

– if there is a lot of free space on the hard disk or very little software, perhaps it is decoy;

– if the names of the folders are trivial (for example, "Salaries", "Customer Data", "Passwords"), it is obvious that the systems are aimed at luring intruders.

All these signals tell attackers that the system may be fake. Also, these systems have several disadvantages:

– one needs to separately configure each fake server;

– Honeypots do not interact with each other and with elements of real infrastructure. They leave no trace and are difficult to detect by a hacker;

– Honeypots are generally not integrated into a centralized system.

This technology was gradually replaced by another, more advanced and smarter one – Deception [18].

Social engineering and phishing attacks are an example of how you can get around any class of solutions, including decoys.

Many modern attacks begin with the delivery of "decoy" to the user, such as a phishing email that they open on their computer. This allows malware to infiltrate the internal network and allows the attacker to proceed to plan and execute the next stage of the attack [19].

Honeypots are unable to handle a phishing attack the way users do. Therefore, Honeypots will not be able to provoke and detect an attack using such a vector. Unlike Honeypots, next-generation cheating technologies can automatically change the decoy environment without leaving it static, as befits a real network in which user and network data change naturally. At the same time, deception technologies detect an attacker in just three to four steps in the network, even if the elements of deception are not deployed on each node [20].

Next-generation deception technologies provide users with powerful real-time attack detection and forensic collection functionality, with virtually no false positives, and attackers will never know they're under surveillance. Decoys are also effective for detecting attackers, but they have a much lower level of detection of real threats, generate much more false positives and do not provide forensics from real nodes that attackers use to attack [21, 22].

Deception refers to the solutions of the Intrusion Detection System (IDS) class – an intrusion detection system. The main purpose of such a system is to detect unwanted attempts to access the network. In other words, Deception helps detect network attacks. Honeypot is a separate network resource that does not interact with anyone but only waits for the attacker to record his/her actions [23, 24]. On the other hand, Deception technologies are a centralized system for managing fake network objects, commonly referred to as traps (decoys). Each trap is essentially a separate decoy but they are all connected to a central server. The scheme of Deception technology is shown in Fig. 2.

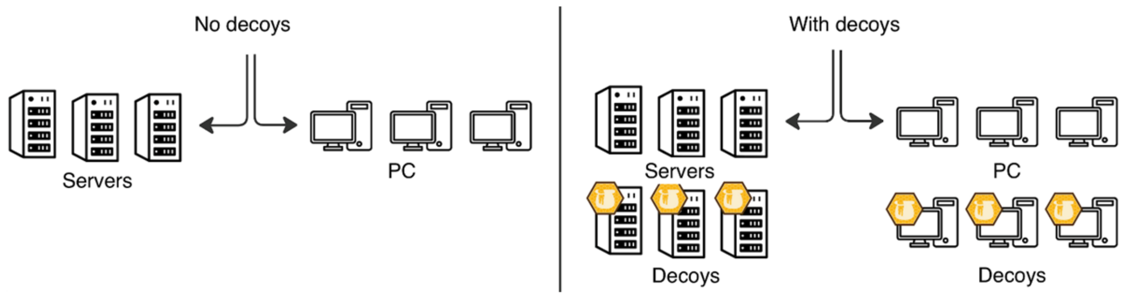

Fig. 1. Honeypot system diagram

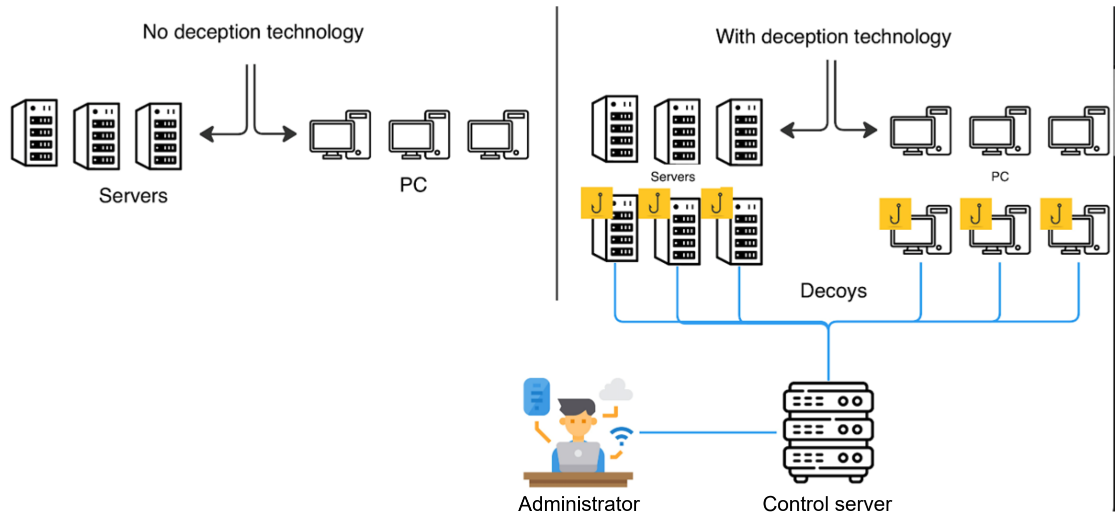

Fig. 2. Cheating system scheme

Such solutions usually have a convenient interface for managing traps. The operator can create traps with the right set of emulated network services, on the selected subnet, with the desired method of obtaining an IP address, etc. Traps and services emulated on them maintain constant communication with the server. Like Honeypots, Deception traps do not provide legitimate interaction with the network (except for interacting with other components of Deception [25–27]). The trap will notify the server of any attempts to interact with it: this serves as an attack indicator. In this scenario, the operator can instantly receive a notification of the event. It will indicate the details of what happened: the address and port of the source and target, the protocol of interaction, the response time, and so on. Additional modules in Deception technology can also provide manual or automated incident response capabilities (Fig. 3).

The concept of deception may include other things. Some components help simplify the configuration and automation of deployment, others make traps more like real network services, and still others draw the attention of hackers to fake targets [28, 29]. Some components can perform related tasks, such as responding to incidents, collecting compromising indicators from workstations, and looking for vulnerable software [30].

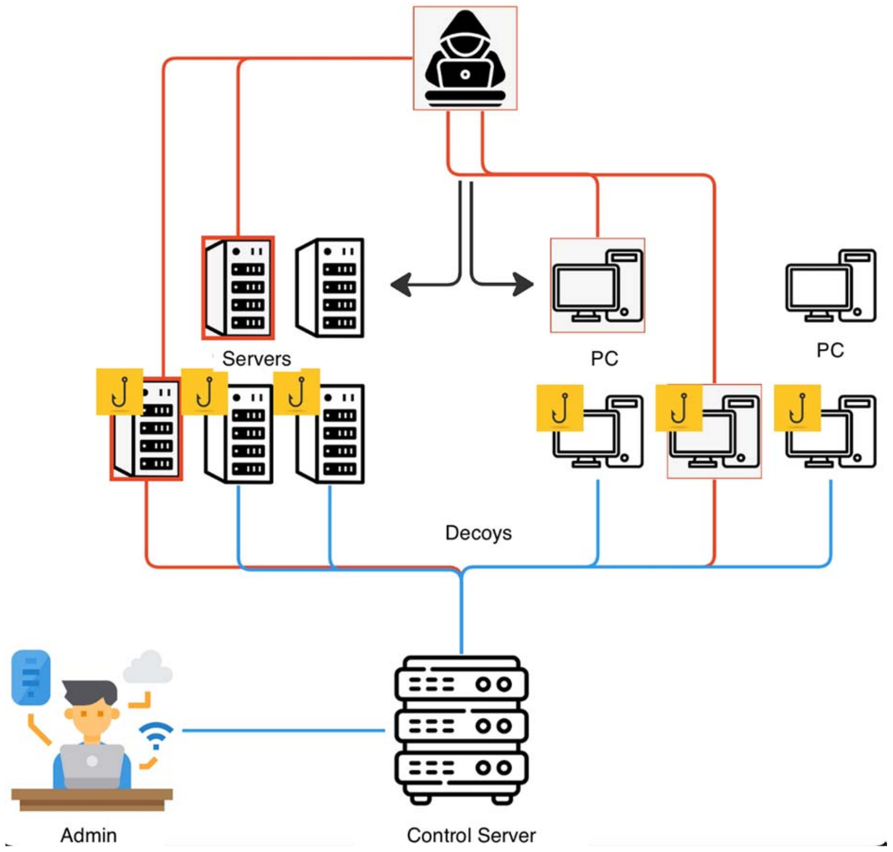

Fig. 3. Decoy system scheme

An agent is a program that is installed on real workstations or users' servers. It is able to communicate with the deception server, execute its commands or transmit user data to the control center. Among the solutions of the Deception class are both products containing the agent and those that do without it (Fig. 4) [31–33].

Tasks for agents may include:

– collection of data on the state of the AWP;

– distribution of decoys;

– emulation of activity in the network;

– response to the incident (manual or automated);

– data collection for forensic science;

– other – according to the needs of customers and the imagination of the developer.

The activity of agents must be hidden from the person who works at the computer. First, the user can intentionally or accidentally remove the agent or its components. Secondly, the presence on the workstation of unknown (or to some extent known – if the user is warned about it) software can cause a feeling of discomfort. Thirdly, everything that the user sees will be seen by an attacker who gained access to this computer [34].

Agency decisions within the framework of deception should be made in such a way that the user does not see either the agent or traces of its vital activity (or at least tries to minimize this). Therefore, agents usually work in privileged mode, like a driver for Windows or a kernel module for Linux. This enables, for example, to intercept system calls to ensure secrecy, and also does not allow the user to remove the agent or prevent its operation [35, 36].

Decoy is an object that is imperceptibly placed on a real workstation. The decoy looks like something ordinary and attractive to an attacker ("accidentally" forgotten password file, a saved session, a browser bookmark, a registry entry, a mounted share, etc.). Honeypot contains links and data to access a fake network resource. An attacker, having found such a link and authorization data, of course, wants to check what kind of service it is. It falls into a trap, and then the signal about the event is triggered (Fig. 5).

The types and methods of placing the decoy depend on the type of trap to which the decoy leads. Decoys can be distributed in several ways. If agents are present in the deception, they are tasked with scattering decoys. In this case, the process can be easily automated: the control server sends a command to the agent, and the latter performs the necessary actions to install the decoy [37].

So, we want to substitute the authorization data into a decoy that is as similar as possible to the real ones. At the same time, in each organization, user data looks different. Everyone has different entry formats (for example, logins of the form "the first letter of the name-period-surname" in Latin are often found). Everyone has their own password policy. For some decoys, you may need a mailing address, domain address, or something else. The problem can be solved by maintaining a database of fake network users. There are different approaches to maintaining such a database (Fig. 6).

For example, Deception can be integrated with a traffic analysis system. This makes it possible to recognize the presence of authorization data in network traffic, find common features in them, and generate users similar to real ones according to identified rules [38].

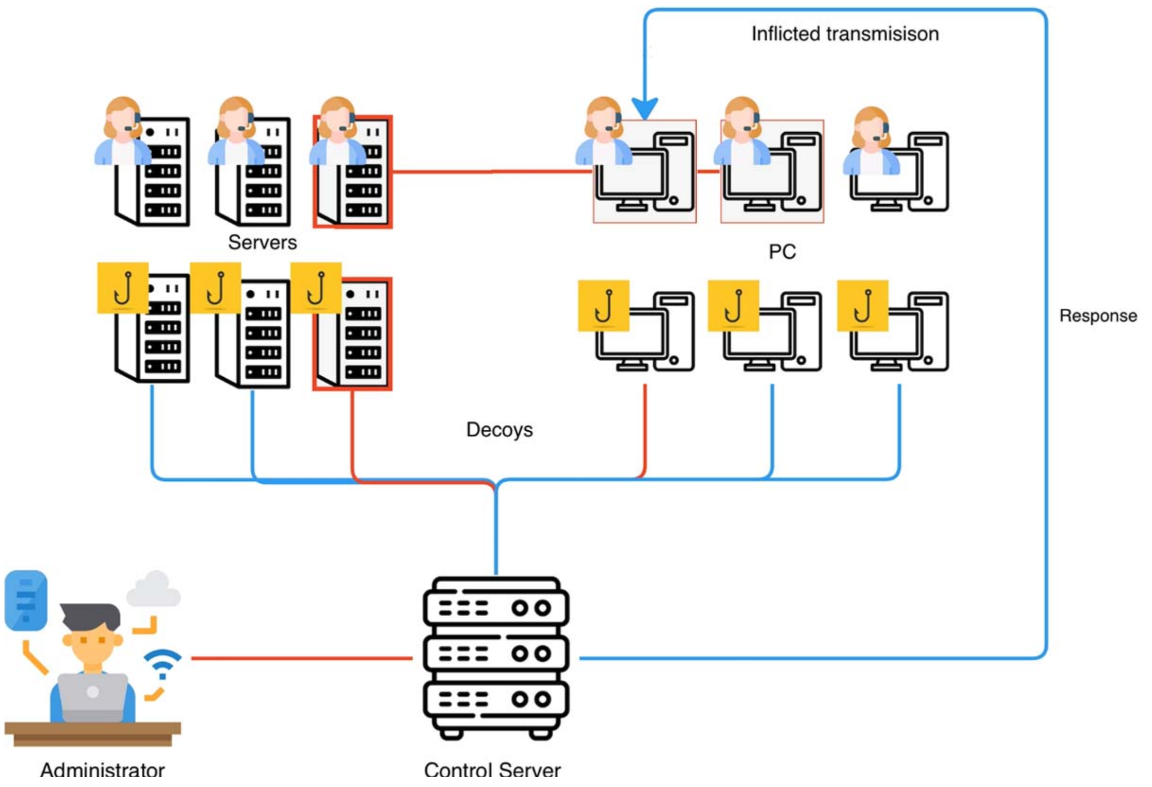

Fig. 4. Scheme of the agent system

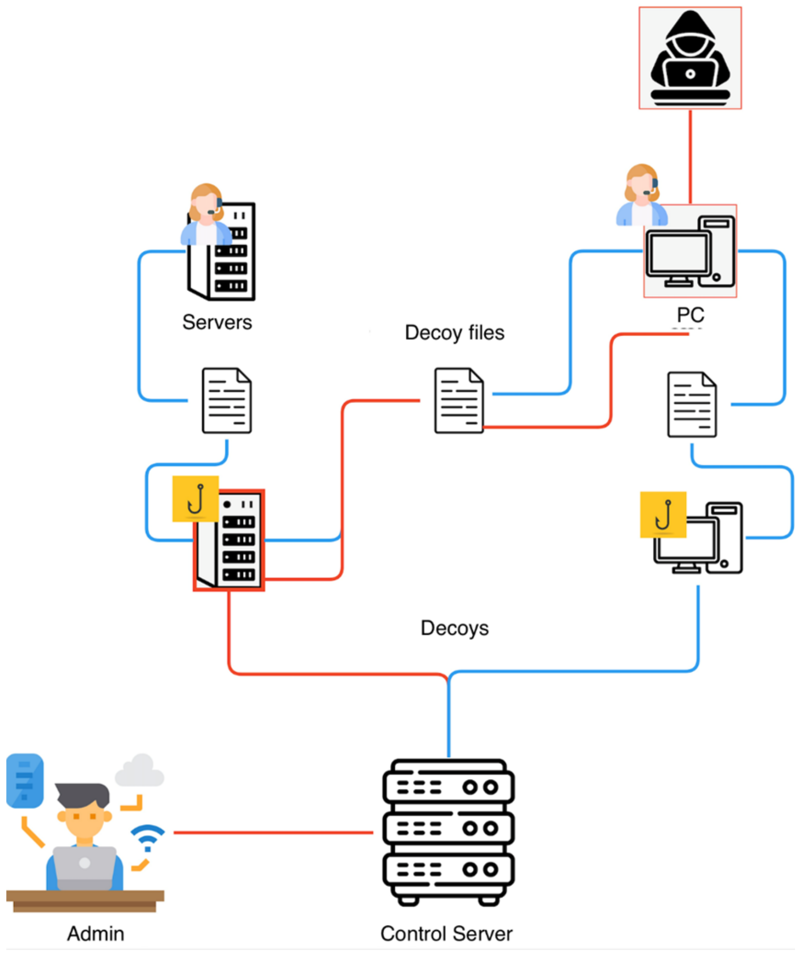

Fig. 5. Decoy, agent, and bait in the system

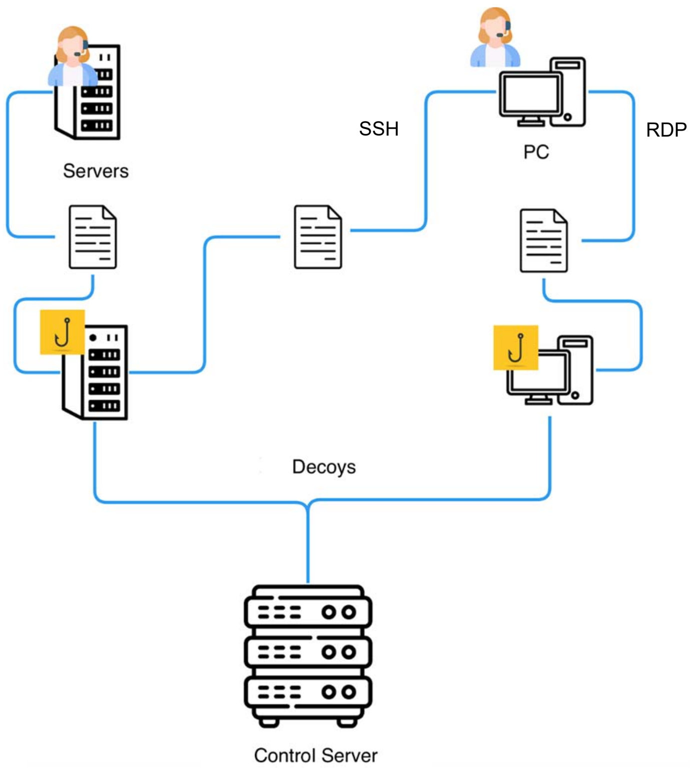

Fig. 6. Scheme of the system of false users

5. 2. Dynamic system modeling with honeypots

Analysis of deception systems and its predecessor Honeypot in the previous chapters showed the prospects for the development and evolution of this technology and its possibilities for expansion. However, both Deception and Honeypot are centralized systems that still have all the disadvantages of a centralized approach, namely control server. If the main link is detected, the hacker can adjust his/her actions. One can level this risk by building a protection system that will not depend on only one central node. Blockchain is a multi-node system in which each node must confirm the information that goes to one of the links before letting it into the data stream. The property of dynamic change and validation of blockchain nodes can repeatedly strengthen security systems and prevent the problem of centralized management. Based on this, it is necessary to simulate a dynamic distributed management system using the dynamic properties of the blockchain and investigate the parameters of this system. Therefore, we shall introduce a dynamic dynamic distributed model of Honeypot formed by N hosts and four services. As shown in Fig. 7, there are two participants: a hacker and a legitimate user who is synchronized with a real service (that is, the client can save the location using a real service and knows the exact location). N hosts make up a private blockchain, which is a P2P network and does not open its doors to the outside world [39–41].

Solana (i. e., blockchain platform) serves as the lower level of the system. N hosts form a private blockchain that forms the P2P network. By calculating the hash value of a block, a host in a private chain can extract a potential block and load it into a chain. This mechanism ensures the distribution and decentralization of the deployment architecture. The temporary host executes a service allocation algorithm and sends the corresponding encrypted information to other hosts. As shown in Fig. 8, in our system, Host0 block miner (a host that successfully calculates a particular hash) becomes the main host in the first block property period. Another host (Host1) can replace Host0 in the next period. A host that has more computing power is likely to be an intermediate center controller. If a narrowly configured host is attacked and its performance decreases, it cannot serve as a central one due to the lack of sufficient computing power, and other hosts will replace it automatically. Therefore, the failure of the host Host0 is irrelevant to the entire system (i. e., the system is functioning normally). Attack logs recorded by one host are uploaded to the blockchain while other nodes synchronize these logs in our private chain. Thus, each node has complete data stored in a secure form for further analysis of attacks [42–45].

There are only four types of services in this scheme, and each service has both genuine and fake attributes (i. e., four genuine services and four relevant (fake) services). Periodic switching of services is performed for each period.

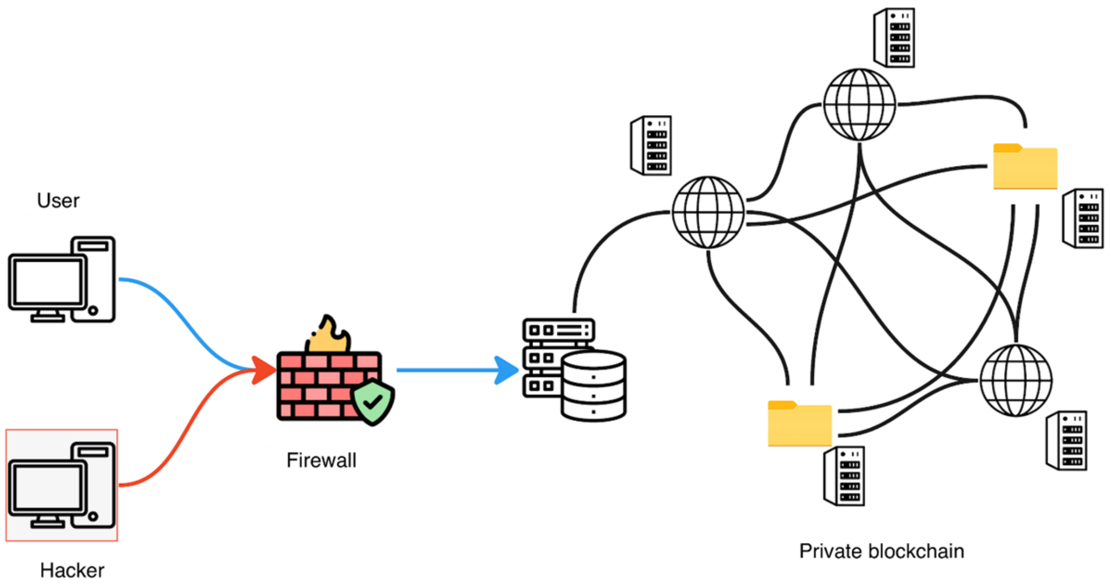

Fig. 7. Honeypot dynamic distributed system model

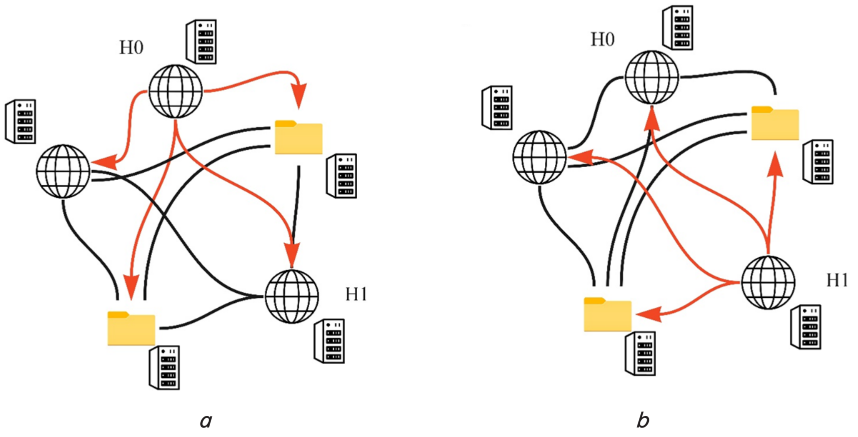

Fig. 8. Different main hosts: *a* – principal host 1 in the first period; *b* – principal host 2 in the second period

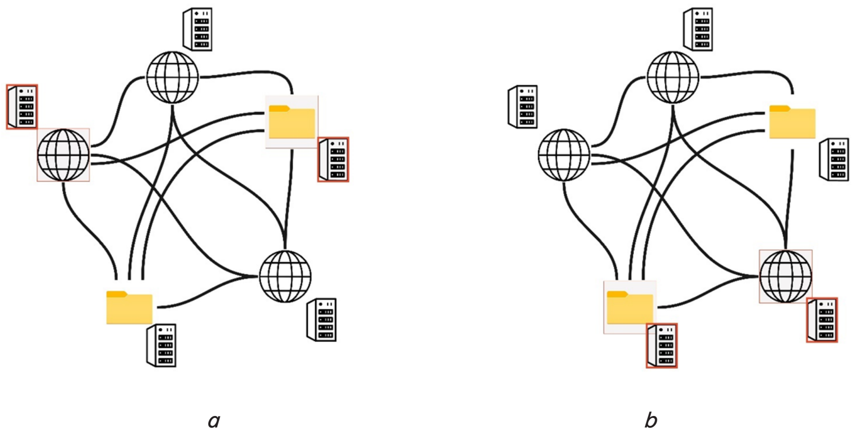

Fig. 9. Distribution comparison: *a* – principal host before moving; *b* – principal host after moving

Periodic switching of services is performed for each period. Comparison of the distribution of services is shown in Fig. 9.

Both types of services are constantly changing. There are three types of applications in the security system:

– if the service on Host0 is genuine, the service may become a decoy in the next period. Once converted, the attacker cannot gain access to the real resources of the service in the present period;

– if the service on Host1 serves as a host service in the first period, according to the promises of anti-Honeypot technology, when an attacker discovers that the service is a trap, s/he will avoid a service that may change to a genuine service in the second period. Thus, it prevents the attacker from accessing real resources;

– if the service on Host0 is genuine in the second period, through synchronization with real users, the customer will only send requests to the real service. Since there are some fake services (such as Honeypots), any traffic accessing Honeypots will be marked as an attack record.

Thus, the transformation and movement of services confuses attackers and protects the developed system.

5. 2. 1. Description of host communication in the built blockchain network

The host that mines the block acts as a non-permanent centering controller. This central host generates conversion information that assigns each host to run different services (i. e., to run a real service or honeypot service) according to the random generation algorithm. The data contains service numbers and 01 encoding, which will be encrypted using the 2048-bit RSA encryption algorithm. The encrypted data is then sent to other hosts, the host of the temporary center on this private network. Upon arrival at the appropriate host, the information is decrypted, and plain text is received. For encoding 01 – zero is the startup symbol of the honeypot service, and one represents the real service. Using the text, a bit comparison is performed, then the specified service is launched to complete the execution procedure. For an authorized user, synchronization is performed to maintain normal operation. By sending the user encrypted information of the real service, the server can provide a regular service. In addition, the user can send encrypted "whois" request data + server name" to actively obtain the desired address of a particular service. Thus, a valid user can access real system resources while using the service.

A formal description of the mechanism of decentralized communication in Fig. 3 is as follows:

– at some point in time, the temporary main host $mHost_j$ asks about a new coin base to the blockchain via the web3J interface. Coinbase introduces a host that successfully mines a block. After that, $mHost_j$ generates a command $Command_{update}$ to update it. $Command_{update}$ has a specific format. $K_{public}(E, N)$ and Enc_1=(($Command_{update})^E$mod$N$) is calculated. An encrypted Enc_1 message is sent to every other host on the private blockchain. After receiving the message, these hosts from $K_{private}(D, N)$ calculate Dec_1=(($Enc_1)^D$mod$N$). After checking Dec_1 in a specific format, the coinbase will be updated on each host. The host combined with it acts as a new temporary principal host. Meanwhile, j in $mHost_j$ changes to the new value;

– the new main host $mHost_j$ has the right to execute the distribution algorithm. Service numbers and 01 codes are generated, which direct other hosts to open or close. They are considered service codes. A $Command_{changeSRV}$ message is sent containing the service codes. Different hosts receive different $Command_{changeSRV}$ messages. $Enc_1 = \left(Command^E_{changeSrv} modN\right)$ is calculated and sent to $cHost_i$, which represents the shared host. $cHost_i$ executes Dec_2=(($Enc_2)^D$mod$N$) and receives a simple message. Dec_2 is installed and the host will open and close the corresponding services;

– the client host sends a request command to one of these servers. The $Request_{srv}$ command contains the message: 'who is Apache'. $Request_{srv}$ is encrypted as enc_1=(($Request_{srv})^e$mod$n$) with the public key $k_{public}(e, n)$ and forwarded to the server;

– the server decrypts enc_1 messages via its private key $k_{private}(d, n)$. dec_1=((enc 1$)^d$mod$n$) is output and verified. The server has a set of request messages $\{R_0, R_1, R_2, R_3\}$. If $S = dec_1 \oplus R_a = 0,\ a \in [0, 3]$, the requested IP address IP_r in enc_1=(($IP_r)^e$mod$n$) will be returned to the client, and its IP address will be added to the main list $List_{client}=\{IP_{c0}, IP_{c1}, ..., IP_{cc}\}$. Otherwise, dec_1 value will be ignored;

– after obtaining IP_r in dec_2=(($enc_2)^d$mod$n$), the client host will connect to this IP address to obtain real resources.

Due to variables in different periods $\{T_1, T_2, T_3, T_t\}$, the real IP address will be updated to IP_f. A host configured with IP_r sends an $Update_{src}$ command to clients in accordance with$List_{client}$. Updated IP_f is encrypted as enc_3=(($Update_{src})^e$mod$n$);

– one calculates dec_3=(($enc)^d$mod$n$). There are four commands $\{C_0, C_1, C_2, C_3$ that follow a special format in clients. If $s = dec_3 \oplus C_a = 0,\ \in\ [0, 3]$, the client connects to the new IP_f. Periodically switching services, the mentioned steps will be cyclically executed.

5. 2. 2. Dynamic system architecture with honeypots

The system will perform a set of atomic actions, i. e., actions={generate, send, receive, wait, open, close, restore, compromise}. Designations used to model the system described in Table 1. System actions and their parameters are summarized in Table 2.

Table 1

Designations used to model the system

Name	Designation
States	$\{s_n, s_c, s_b\}$
Channels	$\{c_1, c_2, ..., c_c\}$
Hosts	$\{h_1, h_2, ..., h_h\}$
Ports	$\{p_1, p_2, ..., p_p\}$
Identifiers	$\{id_1, id_2, ..., id_h\}$
Services	$\left\{srv_1^R, srv_1^H, ..., srv_s^R, srv_s^H\right\}$

Table 2

Description of the system actions

Function	Description
generate(data)	host generates data
open(srv_i)	host opens the service
close(srv_i)	host closes service
send (data, c_i)	the host sends data through the channel
receive (data, c_i)	the host receives data through the channel
compromise()	host compromised
recover()	host restored
wait ($reply_i$)	the host is waiting for a response after sending $reply_i$

Services on the host are abstracted by I/O events. The generate(data) and send (data, c_i) events represent the source data of the services, and the receive event (data, c_i) represents the receipt of data. In terms of security, each host can operate in normal or compromised mode at the same time. Normal mode means that the host is running without malicious data and supports normal operation. However, the compromised mode indicates that the host is working in a malicious way and harming itself. The states relating to the running $host_i$ are divided into two categories: $Service_i^N$ for normal mode and $Service_i^C$ for compromised mode. The worst situation is that the host broke down and stopped working in breakdown mode $Service_i^B$. Thus, states consist of three types {$service_n$, $service_c$, $service_b$}. The host will work as the current mode until there is a transition→InterT, which represents the transition relationship between the three different modes. Transitions shown in Fig. 10 can be defined as follows:

$$Service_a \to T, InterTService_b : Service_a \in Service^N \wedge$$
$$\wedge Service_b \in Service^C,$$

$$Service_b \to T, InterTService_a : Service_b \in Service^C \wedge$$
$$\wedge Service_a \in Service^N,$$

$$Service_b \to T, InterTService_c : Service_b \in Service^C \wedge$$
$$\wedge Service_c \in Service^B,$$

$$Service_c \to T, InterTService_a : Service_c \in Service^B \wedge$$
$$\wedge Service_a \in Service^N.$$

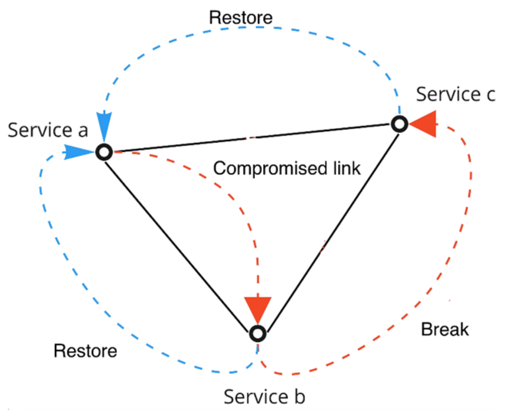

Fig. 10. Transitional states of links

A host consists of five parts $host_i$=(id_i, $Ports_i$, $Services_i$, $States_i$,→T), where id_i is the host ID, $Ports_i$ is a set of ports, $Services_i$ is a set of services, $States_i$ is a set of states i→T is a set of transition relations given in Table 3.

Table 3

Transient dependences of link states

Transition	Designation
$\to Table$	$\to IntraT \cup \to InterT$

$\to IntraT$	{normal state – normal state}
	{attacked state – attacked state}
	{compromised – compromised}

$\to InterT$	{attacked state – normal state}
	{compromised – normal state}
	{compromised – normal state}
	{compromised – attacked state}

Since the data transmitted between hosts guarantee the normal operation of the system, they play an important role in security analysis. It is assumed that each piece of data is generated by only one host. The data can be harmful and contain some commands that lead to malicious activity. A host that creates malicious data is considered hacked. They are described as follows:

$$malicious(data) = \exists s^{generate(data)} \to$$
$$\to Tables' : service \in Service^C,$$

$$compromised(h_i) = \forall s \to$$
$$\to Table, IntraTs' : service \in Service^C \wedge$$
$$\wedge service' \in Service^C.$$

Host h_i with normal behavior is in normal mode $normal(h_i)$=$compromised(h_i)$. It is assumed that if an ordinary host receives malicious data, it will enter compromising mode, further compromising itself. To simulate the spread of unauthorized data during an attack, the following is obtained:

$$service_i^{compromise} \to$$
$$\to Table, InterTService_i \exists Service_{i-1} : Service_{i-1} \to$$
$$\to Table, IntraTservice_i.$$

In order for the compromised host not to intercept the data transmitted during communication, the communication channel must be protected:

$$secure(c) =$$
$$= \forall (h_j, h_i) connectedState(h_j, h_i, c, States) \wedge$$
$$\wedge \neg \exists s^{accept(data)} \to$$
$$\to InterTs' \wedge (compromised(h_i) \vee compromised(h_j)).$$

The system consists of h hosts, as indicated in Table 1. All states in these hosts illustrate the general state of the system, i. e., $Service_{system} = Service_{h1} \cup Service_{h2} \cup ... \cup Service_{hh}$. Continuous authorized behavior of each host (for example, sending data through a channel) ensures the normal functioning of the system. Any hosts communicate with each other by sending and receiving data through communication channels:

$$host_i, send(c, data) \to host_j, receive(c, data).$$

This indicates that sharing a single channel allows you to both connect and transfer shared data. Data exchange can be carried out only when it is connected through one communication channel. So, we define the following statement:

$$connected(host_i, host_j, c, States) =$$
$$= \exists host_j^{connect(c)} \to host_i \wedge host_i^{connect(c)} \to host_j.$$

During the communication process, the behavior of $host_i$ and $host_j$ is shown in Fig. 11.

The $host_i$ generates data and sends data to $host_j$ via the communication channel. After receiving the data from $host_i$, $host_j$ can decide whether to accept or reject this data. Once $host_j$ receives and accepts malicious data, it becomes compromised:

$$host_i, generate(data),$$

$$host_i, send(c, data) \to host_j, receive(c, data),$$

$$host_j, accept(data) \vee host_j, discard(data).$$

If: $malicious(data) \wedge accept(data)$.

Then: $host_j, compromise$.

Thus: $compromisedState(h_j)$.

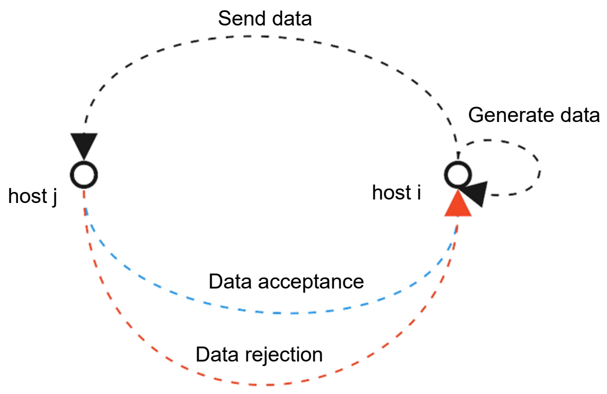

Fig. 11. Behavior during communication

A regular host serves as a legal part of the system and ensures the normal functioning of its services for users. As mentioned above, {compromised – normal state} indicates that a compromised host becomes common during the recovery action:

If: $compromisedState(host_j)$.

Then: $host_j, recover$.

Thus: $normalState(host_j)$.

The system abstracts and focuses on data transmission for subsequent analysis of security attacks.

5. 3. Analysis of the effectiveness of decoys with dynamic attributes of the blockchain system against Deception solutions

Determining the effectiveness of a decoy system with dynamic attributes of a blockchain system involves assessing whether a dynamic decoy system can reduce the burden on the infrastructure of the decoy system, which would ensure the availability of the system during significant loads and provide time for information security specialists to detect an attacker and collect evidence of cyberbullying. Evaluation should be carried out according to the following criteria: data transfer rate, bandwidth, and response time of software services to blockchain systems. It is also necessary to compare the reaction of the proposed system to DOS attacks and requests for services by existing solutions, such as static decoys.

To evaluate the performance of the network with decoys on the blockchain system against Deception solutions, an assessment of network performance and the response time of static hosts and dynamic servers (that is, the proposed scheme) during a SYN DDoS attack is carried out. The implementation of the prototype system is carried out in Python, Java, and Solidity (that is, in the blockchain programming language). In addition, experiments are conducted on five personal computers (PCs) of Windows 16 GB on which WM is installed and they simulate the Linux operating system (OS) with 8 GB of RAM to run services, one PC with Windows 32 GB on which WM is installed and it simulates Linux OS with 16 GB of RAM to launch an attack of various scales, and one Windows PC with 32 GB of RAM for an authorized user. Services (MySQL v8.0.27, Apache v2.4.51, Vsftpd v3.0.5, and Nginx v1.24.4) and Solana v1.6.7 (i. e., the blockchain platform used to form a private blockchain) are installed on five server hosts. The total number of real services on different hosts is calculated to illustrate their average distributions. Three types of attack tests are conducted: sniffer, scan attack, and DDoS attack. Testing the attack is carried out by continuously sending SYN packets at different speeds. The size of the SYN packet for the attack is set to 73695 bytes in Hping3 v3.2.2, indicating that the packet is divided into certain TCP packets. Network performance measurement is carried out using Iperf v3.10.1.

A DDoS attack is an attack model for sending a large number of requests to the target host. The host receives a temporary surge in requests and there will be a breakdown. An illegal attack on a host generates many requests and sends them to $host_3$ using $normalState(host_3)$. After receiving these requests from $host_{attack}$, $host$ will wait for their responses. There will be no response from them, which indicates the waste of system resources and the subsequent consumption of $host_3$ resources until it is broken. Such an attack can be described as follows:

$$[host_{attack}]generate(request1),$$

$$[host_{attack}]generate(request2),$$

...

$$[host_{attack}]generate(requestn),$$

$$[host_{attack}]send(c, request1),$$

$$[host_{attack}]send(c, request2),$$

...

$$[host_{attack}]send(c, requestn),$$

$$[host_3]receive(c, request1),$$

$$[host_3]receive(c, request2),$$

...

$$[host_3]receive(c, requestn),$$

$$[host_3]send(c, reply1),$$

$$[host_3]wait(reply1),$$

$$[host_3]send(c, reply2),$$

$$[host_3]wait(reply2),$$

...

$$[host_3]send(c, replyn),$$

$$[host_3]wait(replyn),$$

$$[host_{attack}]breakdown(host_3).$$

Failure of $host_3$ leads to a single point of failure. To solve the problem, you should take into account the distributed scheme. Compared to a conventional centralized host, a distributed system can handle the problem of a single point of failure. The distributed system contains h hosts and $host \geq 2$. When $host_{attack}$ sends n requests, there are two possible situations for a distributed system:

DDoS attack on one host. In this case, the host *host*$_3$:*breakdown*(*host*$_3$) fails. Even if *host*$_3$ can't function, other hosts (i. e., *host*$_1$, *host*$_2$, *host*$_4$, ..., *host*$_h$ with can still provide user service and maintain the normal functioning of the entire system, thus avoiding a single point of failure.

DDoS attack on all hosts. In such a case, n is accepted as the maximum number of host crash requests, and each host shares the attack traffic. If *host*$_{attack}$ sends n requests, each host receives n/h requests. h distributed hosts significantly reduce illegal flow compared to a single host, indicating that hosts in the system are not crashing.

If a conventional system encounters a DDoS attack, then:

$$host_{attack}^{DDoS^n} \to h.$$

If a distributed system encounters a DDoS attack, then:

$$host_{attack}^{DDoS^{n/h}} \to h_1,$$

$$host_{attack}^{DDoS^{n/h}} \to h_2,$$

$$host_{attack}^{DDoS^{n/h}} \to h_3,$$

...

$$host_{attack}^{DDoS^{n/h}} \to h_h.$$

The prototype system has five distributed and decentralized hosts to effectively mitigate a DDoS attack.

Fig. 12, 13 illustrate the effect of attack speed on network performance on effective bandwidth and TCP traffic. When the attack speed is 0 (i.e., there is no attacking packet), both types of hosts reach their maximum values of 736 MB/s and 100 Mbps in TCP bandwidth and TCP traffic, respectively. However, with an increase in the speed of attack, there is a sharp slowdown from 0 to 1000 packets per second. Apparently, the angle of incidence in static hosts is greater compared to the broken line of dynamic hosts. There is a slow growth in the range of 1000 packets/s to 3000 packets/s, and dynamic host values are still greater than static hosts. Thus, the dynamic honeypot system has an advantage over stationary hosts in terms of network performance.

Trafgen in netsniff-ng v0.6.7 is used to run the SYN attack test. Unlike the SYN package mentioned in the network performance assessment, this type of packet consists of 64 bytes for a SYN flood attack. Since there are four types of services in the developed system, the average response time of a service becomes an indispensable indicator of evaluation. The response time measurement is performed by Jmeter v5.4.2 for each service.

The database query operator "select * from school" is used to measure the time it takes to receive the corresponding data.

As shown in Fig. 14, the static host does not respond at an attack rate of 14 Kbps. However, the response time of dynamic hosts seems to remain unchanged from 0 to 10 Kbps on the X axis and reaches an infinite value after 60 Kbps on the X axis. Comparison with dynamic hosts is impressive, so the MySQL server of a static host suffers from a DDoS attack. Since five distributed hosts distribute the load on the attack, the experimental dynamic host curve demonstrates their superiority in protecting against DDoS attacks.

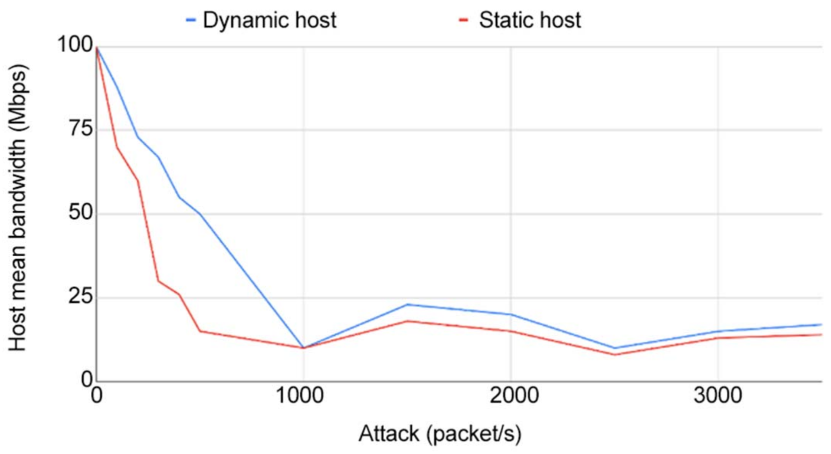

Fig. 12. TCP bandwidth comparison

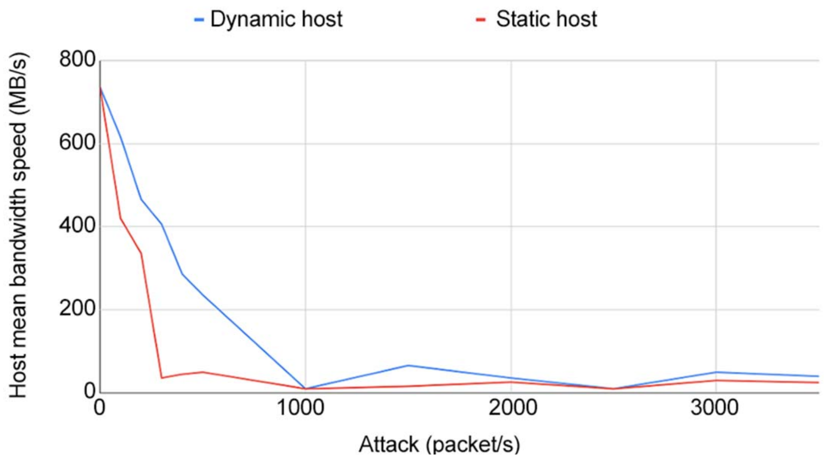

Fig. 13. Comparison of average bandwidth speed

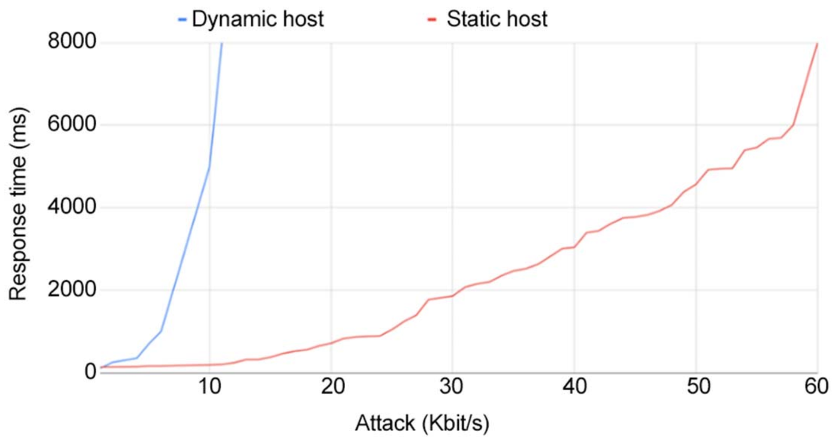

Fig. 14. Response time: MySQL

The loading time of the entire Apache web page is checked. In Fig. 15, dynamic hosts spend more time loading a web page than a static host. This is because blockchain mining work depletes some system resources, which becomes a key factor influencing server response time. The operating times of static and dynamic hosts are almost the same from 1 Mbps to 10 Mbps, slightly increasing along the *X* axis. In this case, both types of hosts are exposed to a DDoS attack. The static server is unresponsive starting at 8.5 Mbps, and the dynamic server response time is higher at the same attack speed. This indicates that a dynamic Apache server can still respond even if the static server crashes.

The Vsftpd and Nginx curves by response time are shown in Fig. 16, 17. The response time of downloading a txt file from a Vsftpd server is measured during a DDoS attack. In Fig. 15, Vsftpd response time in a static host is growing rapidly and reaches its infinity at 11 Kbps. Due to blockchain mining, the overall trend from 0 to 10 Kbit/s has little impact. However, an equal trend in the dynamic host curve indicates resistance to a DDoS attack. Since the dynamic host mining operations deplete system resources, Nginx is exposed.

As shown in Fig. 17, the average response time of Nginx in a static host outperforms the dynamic one by 1 to 2.5 Mbps along the *X* axis. After 2 Mbps, a DDoS attack becomes a major factor affecting response time. From 2 Mbps to 4 Mbps, the dynamic host curve is always lower than the other, which means that the time on a static host is longer than in dynamic hosts.

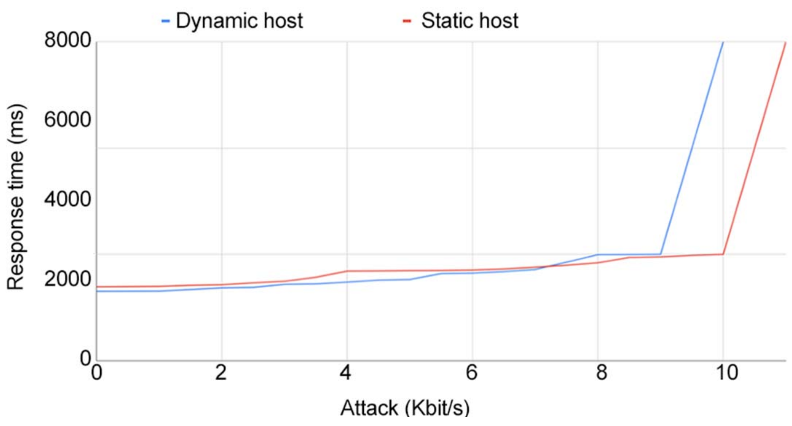

Fig. 15. Response time: Apache

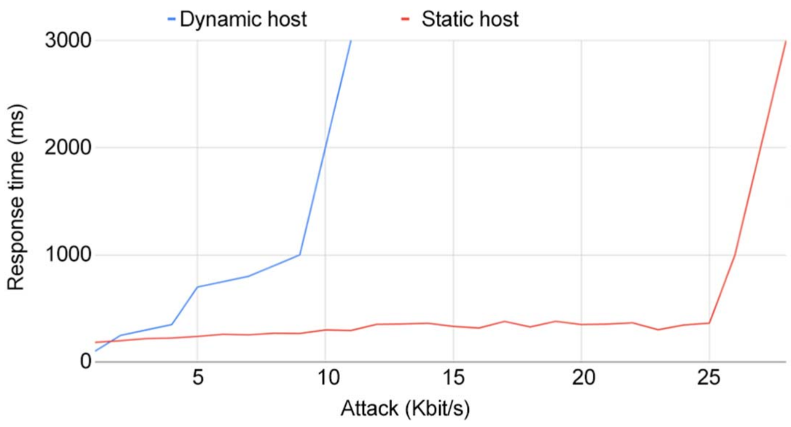

Fig. 16. Response time: VsFTPd

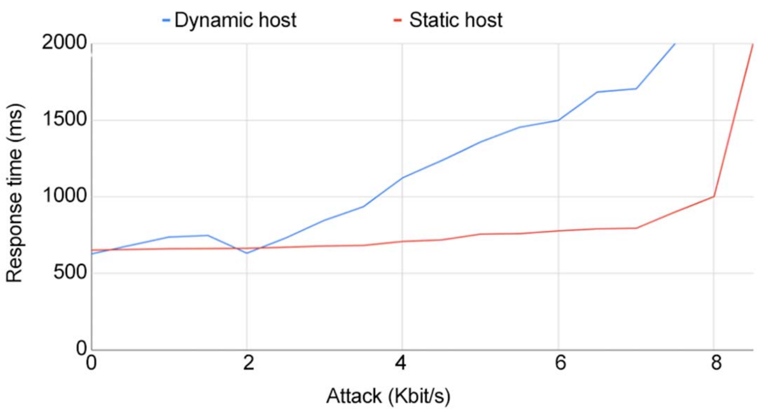

Fig. 17. Response time: Nginx

Nginx creates characters with less memory and high parallelism, so both curves retain their soft characteristic. However, the downtime of a static server occurs earlier than a dynamic one, which indicates the effectiveness of the developed scheme.

6. Discussion of results and areas of further research

As a result, all experimental findings show that the proposed dynamic Honeypot system is superior to the conventional fixed system. Since blockchain consumes system resources, experimental data has little impact on dynamic hosts. Nevertheless, the developed scheme reduces the load in contrast to the conventional fixed solution. Due to the performance evaluation above, it can be argued that the overall response time and network performance of the developed scheme have advantages over the conventional scheme. This can be clearly seen in the charts shown in Fig. 10–15. Honeypot's architecture was an advanced technology for its time. This laid the groundwork for a more proactive approach to cyber defense and allowed attackers to be kept at a safe distance. However, modern cybercriminals are confidently bypassing conventional IPS, avoiding decoys. Obviously, companies that want to defend themselves against advanced and targeted attacks can no longer focus all their resources on this defense alone. The approaches of Honeypots and Deception technologies differ significantly both from the point of view of an attacker and from the point of view of an information security specialist. Honeypots are static systems installed on the selected subnet. They act as a kind of sandbox, trying to lure intruders with confidential data and then control their activities.

On the other hand, Deception technologies represent the "realm of curved mirrors." False information is everywhere in the way of intruders who use it at the stage of horizontal advancement. Attackers are methodical – they collect data, analyze it, and calculate their next step, constantly moving around the network.

Based on observations of penetration testing of companies, it can be concluded that even experienced testers often cannot recognize decoys in the corporate network falling on set traps. This fact once again confirms the effectiveness of Deception and the great prospects that open up to this technology in the future.

However, if we compare the stock Deception system with a modified system of progam decoys, which uses dynamic properties and uses a blockchain platform for this, the picture changes significantly. Taking away one of the drawbacks of software decoys, we get a completely new look at the power of technology, which proves the importance of modifications and improvements to systems. The system of software decoys built on the dynamic properties of the blockchain has a much greater resistance to external attacks.

Unlike [5], where depending on the data of routers, firewalls, IDS, and honeypot, the honeypot configuration is dynamically adjusted, dynamic host offsets require less power. Energy refers to the time to reconfigure the system and the difficulty of observing the correct configuration files. As a result, they can be very easily replaced. With the dnimical replacement of the host, such a joke will not work because data substitution is validated by the entire blockchain network (nodes-validators). The offender needs to gain control over 51 % of the system or convince the system of the correctness of the replaced configurations or distorted data.

As a result of comparing dynamic decoys with static ones, we can conclude that dynamic decoys are a reliable preventive protection system. This means that they make it difficult for an attacker to access the data and increase the time for which you can react. However, the dynamic decoy built on the basis of blockchain further complicates the task for the attacker, which can be traced from the results obtained.

The limitations of this system are the network infrastructure in which it can be located and the computing power. It should be sufficiently extensive and consist of a large number of links, which will be the key to safety in this case. Weak systems, or those that consist of a small number of links (for example, there are only 5 of them in this study), will still be vulnerable to attacks, although much less than static ones.

The development of this study may be the improvement of the system of dynamic change of the host and algorithms for closing and opening ports.

7. Conclusions

1. The main disadvantages of the static decoy system have been identified, in particular, the possibility of detecting a static decoy system for an inexperienced attacker, the inability to centrally configure and change the configuration of the decoy system and its main attributes.

2. The blockchain-based decoy system has been presented and, based on the identified shortcomings, the assumption that the dynamic system is superior to the conventional fixed decoy system has been confirmed since, by distributing the load, it reduces it on the infrastructure of the information system.

3. The qualitative characteristics of a dynamic decoy system are determined, namely its effectiveness due to the ability of the system to respond to requests of various types. The dynamic system's response to network loads and system response time, with different network requests, is determined. A comparison of a static system with a dynamic one was performed using the following criteria: data transfer rate, bandwidth, and response time of software services in the blockchain system. This led to the conclusion that the use of solutions with dynamic attributes in the infrastructure of public and private information systems may be more appropriate than the use of static decoys.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

The data will be provided upon reasonable request.

References

1. Anirudh, M., Thileeban, S. A., Nallathambi, D. J. (2017). Use of honeypots for mitigating DoS attacks targeted on IoT networks. 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP). doi: https://doi.org/10.1109/icccsp.2017.7944057
2. Sardana, A., Joshi, R. (2009). An auto-responsive honeypot architecture for dynamic resource allocation and QoS adaptation in DDoS attacked networks. Computer Communications, 32 (12), 1384–1399. doi: https://doi.org/10.1016/j.comcom.2009.03.005
3. Kuwatly, L., Sraj, M., Al Masri, Z., Artail, H. (2004). A dynamic honeypot design for intrusion detection. The IEEE/ACS International Conference OnPervasive Services, 2004. ICPS 2004. Proceedings. doi: https://doi.org/10.1109/perser.2004.1356776
4. Artail, H., Safa, H., Sraj, M., Kuwatly, I., Al-Masri, Z. (2006). A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. Computers & Security, 25 (4), 274–288. doi: https://doi.org/10.1016/j.cose.2006.02.009
5. Saeedi, A., Nassiri, M., Khotanlou, H. (2012). A dynamic approach for honeypot management. International Journal of Information, Security and Systems Management, 1 (2), 104–109.
6. Fan, W., Fernández, D., Du, Z. (2015). Adaptive and Flexible Virtual Honeynet. Mobile, Secure, and Programmable Networking, 1–17. doi: https://doi.org/10.1007/978-3-319-25744-0_1
7. Hecker, C., Hay, B. (2013). Automated Honeynet Deployment for Dynamic Network Environment. 2013 46th Hawaii International Conference on System Sciences. doi: https://doi.org/10.1109/hicss.2013.110

8. Fraunholz, D., Zimmermann, M., Schotten, H. D. (2017). An adaptive honeypot configuration, deployment and maintenance strategy. 2017 19th International Conference on Advanced Communication Technology (ICACT). doi: https://doi.org/10.23919/icact.2017.7890056
9. Fan, W., Fernández, D., Du, Z. (2017). Versatile virtual honeynet management framework. IET Information Security, 11 (1), 38–45. doi: https://doi.org/10.1049/iet-ifs.2015.0256
10. Casino, F., Dasaklis, T. K., Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telematics and Informatics, 36, 55–81. doi: https://doi.org/10.1016/j.tele.2018.11.006
11. Hepp, T., Wortner, P., Schönhals, A., Gipp, B. (2018). Securing Physical Assets on the Blockchain. Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems. doi: https://doi.org/10.1145/3211933.3211944
12. Cruz, J. P., Kaji, Y., Yanai, N. (2018). RBAC-SC: Role-Based Access Control Using Smart Contract. IEEE Access, 6, 12240–12251. doi: https://doi.org/10.1109/access.2018.2812844
13. Swan, M. (2015). Blockchain Thinking: The Brain as a Decentralized Autonomous Corporation [Commentary]. IEEE Technology and Society Magazine, 34 (4), 41–52. doi: https://doi.org/10.1109/mts.2015.2494358
14. Schütte, J., Fridgen, G., Prinz, W., Rose, T., Urbach, N., Hoeren, T. et al. (2018). Blockchain and Smart Contracts. Technologies, Research Issues and Applications. Fraunhofer. Available at: https://www.iuk.fraunhofer.de/content/dam/iuk/en/docs/Fraunhofer-Paper_Blockchain-and-Smart-Contracts_EN.pdf
15. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V. (2021). Cybercrimes investigation via honeypots in cloud environments. CEUR Workshop, 2923, 91–96. Available at: https://ceur-ws.org/Vol-2923/paper10.pdf
16. Opirskyy, I., Vasylyshyn, S., Piskozub, A. (2020). Analysis of the use of software baits (honeypots) as a means of ensuring information security. Cybersecurity: Education, Science, Technique, 2 (10), 88–97. doi: https://doi.org/10.28925/2663-4023.2020.10.8897
17. Dudykevych, V., Prokopyshyn, I., Chekurin, V., Opirskyy, I., Lakh, Y., Kret, T. et al. (2019). A multicriterial analysis of the efficiency of conservative information security systems. Eastern-European Journal of Enterprise Technologies, 3 (9 (99)), 6–13. doi: https://doi.org/10.15587/1729-4061.2019.166349
18. Banafa, A. (2016). How to Secure the Internet of Things (IoT) with Blockchain. Datafloq. Available at: https://datafloq.com/read/securing-internet-of-things-iot-with-blockchain/
19. Pulling fraud out of the shadows. Global Economic Crime and Fraud Survey 2018. PwC. Available at: https://www.pwc.com/gx/en/news-room/docs/pwc-global-economic-crime-survey-report.pdf
20. McLaughlin, M.-D., Gogan, J. (2018). Challenges and best practices in information security management. MIS Quarterly Executive, 17 (3), 237–262.
21. Joshi, R. C., Sardana, A. (2011). Honeypots. CRC Press, 340. doi: https://doi.org/10.1201/b10738
22. Zhuravchak, D. (2021). Ransomware spread prevention system using python, auditd and linux. Cybersecurity: Education, Science, Technique, 4 (12), 108–116. doi: https://doi.org/10.28925/2663-4023.2021.12.108116
23. Gandotra, V., Singhal, A., Bedi, P. (2012). Threat-Oriented Security Framework: A Proactive Approach in Threat Management. Procedia Technology, 4, 487–494. doi: https://doi.org/10.1016/j.protcy.2012.05.078
24. Onaolapo, J., Mariconti, E., Stringhini, G. (2016). What Happens After You Are Pwnd. Proceedings of the 2016 Internet Measurement Conference. doi: https://doi.org/10.1145/2987443.2987475
25. Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., Welten, S. (2013). Have a snack, pay with Bitcoins. IEEE P2P 2013 Proceedings. doi: https://doi.org/10.1109/p2p.2013.6688717
26. How blockchain can transform defence assets and give armed forces an advantage on the battlefield (2020). PwC. Available at: https://www.pwc.com/gx/en/aerospace-defence/pdf/blockchain-defence.pdf
27. Beecroft, N. (2015). Emerging Risk Report – 2015. Bitcoin. Lloyds. Available at: https://assets.lloyds.com/assets/pdf-bitcoin-bitcoin-final/1/pdf-bitcoin-bitcoin-final.pdf
28. Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M. (2014). Paper 2014/452. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2014/452
29. BGP hijacking. Wikipedia. Available at: https://en.wikipedia.org/w/index.php?title=BGP_hijacking&oldid=820773357
30. Bissias, G., Ozisik, A. P., Levine, B. N., Liberatore, M. (2014). Sybil-Resistant Mixing for Bitcoin. Proceedings of the 13th Workshop on Privacy in the Electronic Society. doi: https://doi.org/10.1145/2665943.2665955
31. Bitcoin Block Reward Halving Countdown. Available at: https://www.bitcoinblockhalf.com/
32. Grafiki blokcheyna. Available at: https://www.blockchain.com/explorer/charts
33. Bitcoin Energy Consumption Index. Available at: https://digiconomist.net/bitcoin-energy-consumption
34. Cryptocurrency statistics. Available at: https://bitinfocharts.com/
35. How much of BIP 62 ("Dealing with malleability") has been implemented? Available at: https://bitcoin.stackexchange.com/questions/35904/how-much-of-bip-62-dealing-with-malleability-has-been-implemented
36. Blockchain and Distributed Ledger Technology (DLT). Available at: https://www.geeksforgeeks.org/blockchain-and-distributed-ledger-technology-dlt/
37. Bonneau, J. Why buy when you can rent? Bribery attacks on Bitcoin-style consensus. Available at: https://jbonneau.com/doc/BFGKN14-bitcoin_bribery.pdf

38. Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., Wustrow, E. (2013). Paper 2013/734. Elliptic Curve Cryptography in Practice. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2013/734
39. Boverman, A. (2011). Timejacking & Bitcoin. Culubas. Available at: http://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html
40. Bruce, J. D. (2014). The Mini-Blockchain Scheme. Available at: https://www.weusecoins.com/assets/pdf/library/The%20Mini-Blockchain%20Scheme.pdf
41. Buldas, A., Kroonmaa, A., Laanoja, R. (2013). Paper 2013/834. Keyless Signatures' Infrastructure: How to Build Global Distributed Hash-Trees. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2013/834
42. Buterin, V. (2014). A next-generation smart contract and decentralized application platform. White Paper, 3 (37).
43. Blockchain's Once-Feared 51% Attack Is Now Becoming Regular. Available at: https://www.coindesk.com/markets/2018/06/08/blockchains-once-feared-51-attack-is-now-becoming-regular/
44. Castro, M., Liskov, B. (1999). Practical Byzantine Fault Tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation. New Orleans. Available at: https://pmg.csail.mit.edu/papers/osdi99.pdf
45. Chen, T., Li, X., Luo, X., Zhang, X. (2017). Under-optimized smart contracts devour your money. 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). doi: https://doi.org/10.1109/saner.2017.7884650