УДК 004.056
DOI: 10.31673/2409-7292.2023.010003

Vyshnivskyi V. V., Danylov I. D.

VIRTUAL CLOUD ENVIRONMENT PROTECTION METHOD BASED ON ATTACK GRAPH STRUCTURE

The article proposes the creation of an effective system for detecting and responding to external influences in order to minimize the consequences of breaching the protection of cloud virtual resources in a timely manner. Mathematical models have been improved to ensure the protection of virtual cloud resources for software-configured networks, namely: a mathematical model of the impact of an attack on virtual cloud resources, a mathematical model for assessing the state of virtual cloud resources, a mathematical model for choosing a countermeasure based on a complex indicator for software-configured networks. Based on the obtained mathematical models, a graph of attacks on the virtual cloud environment was developed. This graph allows you to get information about all known system vulnerabilities.

Keywords: Cloud, Cloud Technologies, Attack Graph, Alert Correlation Graph, Attack Scenario.

Introduction

Clouds open up a new approach to computing where hardware and software are not owned by the enterprise. Instead, the provider provides the customer with a ready-made service. Using the cloud in many cases allows you to reduce costs by two to three times compared to maintaining your own developed IT structure. Also, the main advantage of using the cloud is the ability to quickly adapt to changes in the environment of any institution in the conditions of rapid development of all branches of science and technology [1].

It is appropriate to note some disadvantages of the cloud. One of the significant disadvantages is the need for high-speed Internet access. The rapid increase in the number of Internet providers and the constant improvement of the quality of Internet services should solve this problem, however, possible interruptions in the work or problems with the providers can lead to the stoppage of the work of departments or entire enterprises for a short time. Also, the limited functionality of programs when working with them via the Internet can be attributed to the disadvantages of the cloud.

Problems of compromising information in a virtual environment

The goal of the attacker is to gain access to certain resources, information or, at least, disrupt the normal operation of the network. The problem with link layer security is that by breaking the network at the link layer, an attacker can bypass defenses at higher layers. As a rule, attacks are carried out in a complex, and not one at a time. Depending on the result that will be obtained, attacks can be divided into the following several types: man in the middle (Man in the middle, MitM); denial of service (DoS); unauthorized access to the network; disruption of the network or its sections.

Cloud Alliance Security (CSA) research shows that among all the challenges, unauthorized access and use of cloud computing is seen as a security threat in which an attacker can use vulnerabilities in the cloud and its system resources to launch attacks. In traditional data centers, where system administrators have full control over host machines, a vulnerability can be discovered and patched by a system administrator in a centralized manner. However, patching known security holes in cloud data centers, where cloud users typically have the privilege to manage the installed software on their managed virtual machines, cannot work effectively. In addition, users can install vulnerable cloud software on their virtual machines, which significantly contributes to cloud security breaches. The challenge is to create an effective attack detection and response system to accurately detect attacks and minimize the consequences of a breach of reliability and security for cloud users [2].

The purpose of the study is to develop a methodology for ensuring the security of virtual cloud resources based on the detection of attacks and reconfiguration of virtual networks in a virtual cloud environment.

Development of the attack graph

An attack graph is a modeling tool for illustrating all possible paths of attacks, identifying them and adopting appropriate countermeasures. In the attack graph, each node represents the

© Vyshnivskyi, V. V., & Danylov, I. D. (2023). Virtual Cloud Environment Protection Method Based on Attack Graph Structure. Сучасний захист інформації, 1(53), 24–33. https://doi.org/10.31673/2409-7292.2023.010003.

vulnerabilities of the virtual environment, or the consequence of exploiting these vulnerabilities. It is also necessary to understand that the normal operation of the protocol can be used as a vulnerability. The attack graph helps identify potential threats, as well as possible attacks and known vulnerabilities in cloud virtual environments. Countering attempts to breach the protection of virtual cloud resources is urgent. Attackers' actions can be described by an attack graph.

Informally, an attack graph is a graph representing different sequences of actions by an attacker to achieve his goal. Such sequences of actions are called paths (paths) of attacks. In fig. 1 shows an example of an attack graph [3].

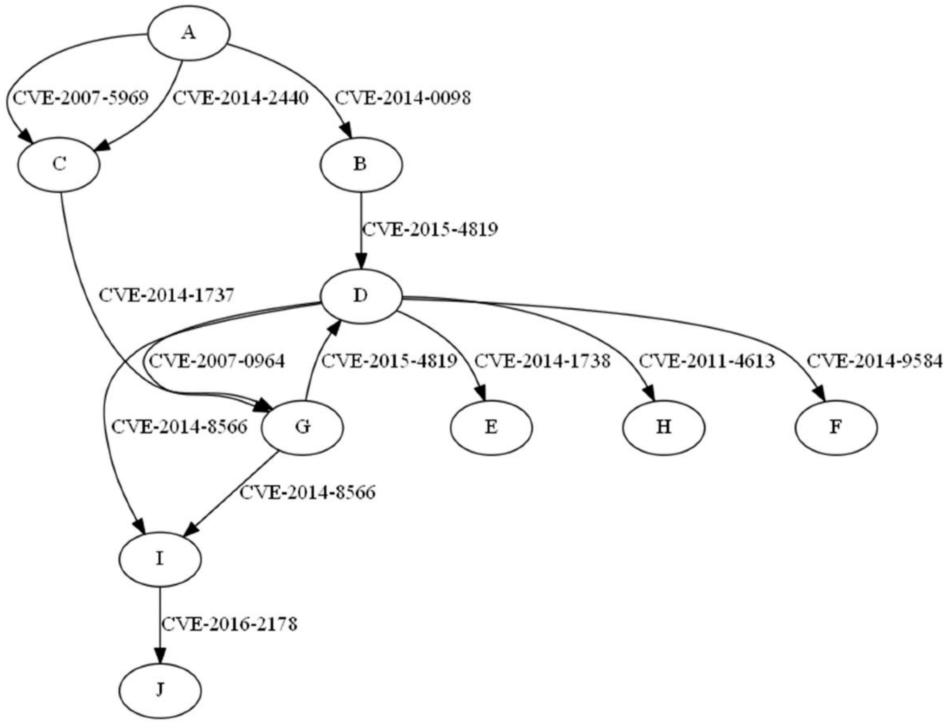

Fig. 1. An example of an attack graph [3]

As a rule, an atomic attack is understood as the use of a vulnerability by the violator. During the synthesis of the attack graph, the following tasks arise: formalization of the concept of attack, development of a formal language for modeling attacks and a computer system (which includes the offender, his goals, networks, means of protection, the attitude of reachability of hosts, etc.), the selection or development of means of construction attack graph and its visualization, development of tools for automating graph construction and analysis.

Attack graphs are mainly considered in the context of network security analysis. Usually, such an analysis is reduced to sequential scanning of all network hosts for the presence of known vulnerabilities. The result is a report that contains a list of vulnerabilities found and recommendations for their elimination. Currently, another paradigm of security analysis is gradually being introduced, which takes into account the "topology" of the computer system - the interconnection of computer system objects, their properties and characteristics. Such security analysis is called topological. Topological analysis of security involves the construction of an attack graph based on the results of a network scan, a model of the intruder and data on the network configuration (ME filtering, routing, attack detection, reachability of hosts, etc.) and its analysis (probabilistic, minimization, etc.).

Built on the basis of security analysis, the graph contains all known attack scenarios for the attacker to achieve threats. The result of its analysis can be: a list of successful attacks that are not detected by IDS; the ratio of implemented security measures and the level of network security; a list of the most critical vulnerabilities; a list of measures to prevent the use of vulnerabilities in software for which there are no updates; at least a set of measures, the implementation of which will make the network protected. Attack graphs are also used in the investigation of computer incidents, to analyze the risks and correlate the warnings of attack detection systems.

Development of an Alert Correlation Graph

Since the attack graph provides information about all known vulnerabilities in the system and about connections, we get a complete picture of the current situation in the field of system protection, where we can predict possible threats and attacks by correlating detected events. If an event is recognized as a potential attack, we may apply specific measures to mitigate the effects of the attack or take measures to prevent the event. In order to describe the attack and the result of its actions, it is necessary to develop an Attack Graph Scenario (AGS) [4].

AGS is a tuple $G_{\mathrm{AGS}} = (V, E)$, where V is the set of vertices of the attack graph, and E is the set of directed edges connecting the vertices of the attack graph. The vertices of the attack graph can be of three types: conjunction nodes B_{c} (vulnerability), disjunction nodes B_{d} (the result of exploiting the vulnerability) and the root node B_{cor} (the initial stage of the attack scenario).

The set of vertices of the attack graph V is defined as follows:

$$V = B_{\mathrm{c}} \cup B_{\mathrm{d}} \cup B_{\mathrm{cor}}. \qquad (1)$$

A set of edges $e \in E_{pre} \subseteq B_{\mathrm{d}} \times B_{\mathrm{c}}$ reflects that B_{d} must be performed to achieve B_{c}. Edge $e \in E_{post} \subseteq B_{\mathrm{c}} \times B_{\mathrm{d}}$ means that B_{d} must be received to B_{c} was performed:

$$E = E_{pre} \cup E_{post}, \qquad (2)$$

where E_{pre} are edges that reflect the relationship between the result of exploiting a vulnerability in the previous node with the vulnerability itself in the next node, E_{post} are edges that represent the relationship between vulnerabilities in a previous node with the possible outcomes of exploiting vulnerabilities in a subsequent node.

The Alert Correlation Graph (ACG) is a set of vertices that reflect vulnerabilities and those that reflect the use of the vulnerabilities (Fig. 2). It is a set of oriented edges that connect the vertices that are possible paths of the attack taking into account the time stamps, the correspondence of the received alerts from the network agent to the corresponding nodes in the system; and the correspondence of the class of these protections to the possible vulnerabilities in the corresponding vehicle. We define a new ACG to display warnings in the corresponding nodes. In order to monitor the development of the attack, it is necessary to monitor the IP addresses of the source and the activity of the attack. ACG is a tuple $G_{\mathrm{ACG}} = (A, E, P)$.

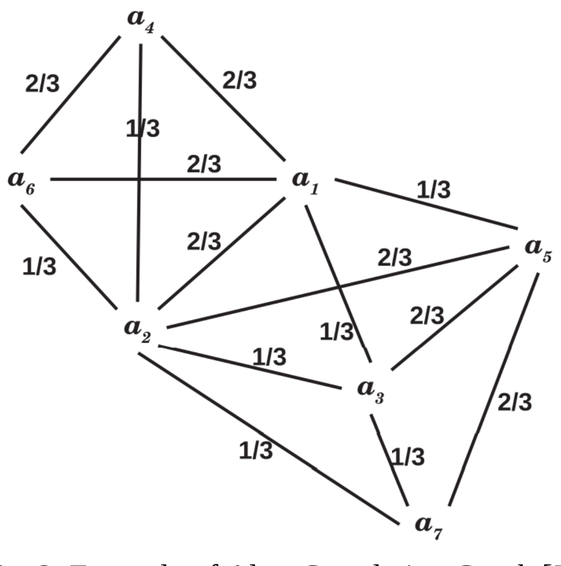

Fig. 2. Example of Alert Correlation Graph [5]

Set A contains all alerts. Alert, $a \in A$ is a data structure representing the source IP address, destination IP address, alert type, and timestamp [5].

Each message refers to a pair of vertices (v_c; v_d) in ACG using the function *map*(*a*), i.e. reflects:

$$map(a)\colon a \rightarrow \{(vc, vd) | (a.source \in vc.node) \wedge (a.dest \in vd.node) \wedge (a.class = vc.vuln)\} \quad (3)$$

where v_c is a vertex that represents vulnerability; v_d is a vertex that represents the exploit of the vulnerability; *a.source* is alert with the IP address of the source; v_c*.node* is a vertex that corresponds to a specific node in the cloud environment; *a.dest* is alert with the IP address of the destination node; vd.node is a vertex that corresponds to a specific node in the cloud environment associated with v_c*.node*; *a.class* is alert with vulnerability; v_c*.vuln* is vulnerability in the node under consideration.

Directed edges represent the correlation between two warnings (*a, a'*) in the event that the criteria below are met:

$$(a.time < a'.time) \wedge (a'.time - a.time < \text{border value}), \quad (4)$$

where *a.time* is alert with timestamp in the previous node; *a'.time* is a timestamped alert at the next node

$$\exists\ (v_c, v_d) \in E_{pre}\colon (a.dest \in v_d.node \wedge a'.source \in v_c.node). \quad (5)$$

ACG includes P as a set of attack paths. Route $S_i \subset P$ is a set of related alerts in chronological order that belong to the same attack scenario.

Let's skip that A contains aggregated alerts, not empty ones. Raw signals that have the same source of sender and receiver address, attack type and timestamp within a given window, which are aggregated as "aggregated alerts". Each ordered pair (*a, a'*) in ACG is mapped to two adjacent vertices in AGS with the difference in timestamps of the two warnings, within a predetermined threshold value. ACG shows the dependencies of alerts in chronological order and we can find alerts related in the same attack scenario by searching for an alert in ACG. The set of sets of attack paths in the alert correlation graph are used to store all paths from the root of the graph to the target node in the AGS, and each path $S_i \subset P$ are alerts that belong to the same attack scenario.

The alert correlation algorithm for each alert detects and returns one or more paths to the database S_i. Each alert that is received from the intrusion detection system is added to the ACG if it does not already exist there. For this new alert, the corresponding vertex in the AGS is located using the function *map*(*a*). For this vertex in AGS, the alerts associated with its vertex type B_c correlated with new alerts. This creates a new set of alerts that belong to the route S_i in ACG or create a new route S_i+1 in S_i. At the end of the algorithm, the network identifier is added to the alert, so that the algorithm can be run more quickly next time. At the end, the algorithm returns the path of the attack in the ACG.

For all virtual machines in the network, depending on the number of vulnerabilities inherent in them, and the assessment of the impact of vulnerabilities on the virtual cloud environment, an appropriate index is assigned. Vulnerability Impact Assessment [6] provides an opportunity to determine the confidentiality, integrity, and availability impact of a vulnerability.

Development of a countermeasure selection model

It is necessary to develop a mathematical model for the selection of countermeasures. To select countermeasures for an attack scenario. When vulnerabilities are discovered or some virtual machines are identified as suspicious, countermeasures can be taken to limit the attacker's ability. The purpose of countermeasures is to protect virtual machines from compromise.

We will use the attack graph as a model of the metric security system to assess security risks. In order to assess the state of network security risk and for the current network configuration, it is necessary to use the security indicators in the attack graph (to measure the probability of risk). The attack graph contains information about system vulnerabilities. For initial or external nodes (i.e., the root of the graph, $B_{\text{cor}} \subseteq B_{\text{d}}$), always the a priori probability is defined as the probability of the threat source. An indicator is used to denote the a priori risk probability of the root node of the graph *GV* from the vulnerability database. As a rule, for *GV* a high probability is assigned, such as 0.7 to 1.

For each node, each attack step $E \in B_{\text{c}}$ will have the opportunity to be exploited; vulnerability is denoted as $GM[e]$. GM[*e*] is assigned according to the basic indicator $H(v)$ from CVSS (Common Vulnerability Scoring System). The basic assessment, as shown in formula (6) [6], is calculated by the impact factor and the use of vulnerability. A baseline score can be obtained directly from the National Vulnerability Database [7] by searching for vulnerabilities by their CVE ID

$$H(v) = k_w I_v + (1 - k_w) I_v, \tag{6}$$

where
$$I_v = 1 - (1 - C)(1 - I)(1 - A),\ 0 \le k_w \le 1. \tag{7}$$

The impact (I_v) is determined by three main security parameters, namely confidentiality (C), integrity (I) and availability (A).

In an attack graph, relations between vertices can be disjunctive or conjunctive, depending on how they are related to each other due to the conditions of their dependencies [7]. Such relationships can be represented in the form of conditional probability.

We will apply mitigation strategies in accordance with received vulnerability alerts. To ensure the protection of virtual cloud resources, it is necessary to store all known countermeasures instructions in the countermeasures database.

$$Q = \{Z_1, Z_2, ..., Z_n\}, \tag{8}$$

where Q is countermeasures database; Z is countermeasure.

Every $Z \in Q$ and is a tuple $Z = (B, I)$, where B is the cost, i.e., the costs required to apply the countermeasures in terms of resources and operational complexity, and is defined on a scale of 1 to 5 and above, where the highest score means a higher cost; I is intrusiveness, which defines the negative effect that a countermeasure has on a Service Level Agreement (SLA). For example, if the countermeasure has no impact on the SLA, then the intrusiveness value is 0.

In general, a database of countermeasures may contain a large number of countermeasures that can be applied to a cloud virtual environment, depending on the existing methods that can be used to implement countermeasures in a cloud virtual environment. The optimal selection of countermeasures is a multi-objective optimization problem to calculate the minimum value of the intervention i.e. MIN intrusiveness or cost, and MAX benefit.

The countermeasure that gives the smallest value of the complex countermeasure selection indicator is defined as the desired one. The attack graph scenario and alert correlation graph are also updated before the algorithm finishes.

To achieve the goal of choosing a countermeasure, a comprehensive countermeasure selection indicator Y is proposed

$$Y = [(k_1 I) + (k_2 B)] / n, \tag{9}$$

where k_1, k_2 are parameters of the influence of countermeasure indicators; I is intrusiveness, which reflects the magnitude of the negative effect that the countermeasure causes; B is a cost that reflects the costs that are required to apply countermeasures in terms of resources and operational complexity (the higher the figure, the greater the cost); n is normalization factor.

Using the virtual network reconfiguration approach, the upper-layer applications of the OSI model will experience minimal impact from lower-layer changes. In particular, this approach is possible only when using specialized software in network switches with automation of their reconfiguration into dynamic networks using the Openflow protocol.

Countermeasures such as traffic isolation can be implemented by leveraging OVS and OFS engineering capabilities to limit attack potential and configure the virtual network to further isolate the flow of suspicious traffic. When suspicious activities such as network port scans are performed, it is important to determine whether attackers are active. For example, attackers can intentionally hide their actions through port scanning to prevent NIDS from detecting their actions. In such a situation, changing the network configuration will force the attacker to perform more actions to explore the network, and in turn, show that he is active.

To evaluate the degree of protection of the virtual machine, we will use the indicator

$$F_{\text{vm}} = (O_{\text{vm}}+O_{\text{Avm}})/2, \tag{10}$$

where O_{vm} is virtual machine vulnerability assessment; O_{Avm} is virtual machine vulnerability exploit assessment.

A virtual machine's vulnerability score is the average base score of a set of vulnerabilities in a virtual machine

$$O_{\text{vm}} = \min \{10, \ln \Sigma\, eH(v)\}, \tag{11}$$

where $H(v)$ is the average base score for each vulnerability of a specific virtual machine; v – a specific virtual machine.

Virtual machine vulnerability exploit assessment is

$$O_{\text{Avm}} = (\min \{10, \ln \Sigma\, eJ(v)\})(N(v) / Z(v)), \tag{12}$$

where $J(v)$ is the average vulnerability exploitation score from each vulnerability of a specific virtual machine; $N(v)$ is the number of services provided by the virtual machine; $Z(v)$ is the number of services that can be provided to a virtual machine.

That is, O_{vm} takes into account the baseline scores of all vulnerabilities on the virtual machine. Each baseline vulnerability assessment takes into account the degree to which an attacker can exploit that vulnerability, as well as the amount of damage it can cause. The exponential sum of the basic indicators allows you to estimate the deviation of their values on a logarithmic scale based on the number of vulnerabilities. Virtual machine vulnerability assessment O_{Avm} on the other hand, it reflects the ability of the attacker to use the vulnerability of the virtual machine and depends on the ratio of the number of used network services to the total number of possible network services. High F_{vm} means that vulnerabilities have a large number of possible paths to achieve an attacker's goal.

So F_{vm} represents a quantitative assessment of the level of protection of each virtual machine in the virtual cloud system. In order to prevent attacks from using other vulnerable VMs to VMs with higher values F_{vm} it is necessary to apply higher degrees of protection. To decrease the value F_{vm} justified strategies for mitigating the impact of the attack, depending on the magnitude of the negative impact.

Basically, the vulnerability score takes into account the baseline scores of all the vulnerabilities on the virtual machine. The baseline score shows the degree of difficulty for an attacker to exploit this vulnerability and the amount of damage that the virtual machine and the system as a whole can sustain. Exponential addition of baselines allows for higher value vulnerabilities to be evaluated and to increase the score on a logarithmic scale based on the number of vulnerabilities. The use of vulnerabilities, on the other hand, reflects the presence of a virtual machine that is targeted by the attack and depends on the ratio of the number of network services provided by the virtual machine to the number of network services that the virtual machine can accept [8]. A higher score indicates a greater number of vulnerabilities, as well as possible paths to achieve the attacker's goal.

The F$_{vm}$ indicator can be used as an indicator to demonstrate the protection status of a virtual machine, that is, a virtual machine with a higher F_{vm} value means that it can be attacked more easily. In order to prevent attacks from using other vulnerable virtual machines, virtual machines with higher Fvm values should be inspected more frequently and appropriate countermeasures should be applied to them to reduce the F_{vm} value.

For each implemented virtual cloud environment, as a protection criterion, the value of the indicator, when reached, the virtual cloud environment will not be able to meet the specified requirements regarding its compromise, is substantiated. The countermeasure selection algorithm shows how to select the optimal countermeasures for a given attack scenario. The input data of the algorithm are alerts, an attack graph, and a database of countermeasures. It begins by selecting a node that responds to an alert from a network agent. Before choosing countermeasures, we calculate the distance to the target node. If the distance is greater than the threshold, no countermeasures are selected, but the ACG is updated and the alert tracking in the system continues.

Since an alert is generated only after the attacker has performed an action, we set the probability of receiving an alert to 1 and calculate the probability for all of its child (down the graph) nodes. Then, for all selected nodes, the comprehensive countermeasure selection indicators are calculated, compared and accepted. Changing the probability of the target regime provides an advantage for applying countermeasures. At the end of the algorithm, the SGA and ACG will also be updated before the algorithm ends.

Evaluation of the effectiveness of the method of protection of the virtual cloud environment

To evaluate the performance of the virtual cloud environment system, a virtual cloud system was developed (Fig. 3), consisting of open (public) virtual servers and private virtual machines (VMs), where virtual domains are installed.

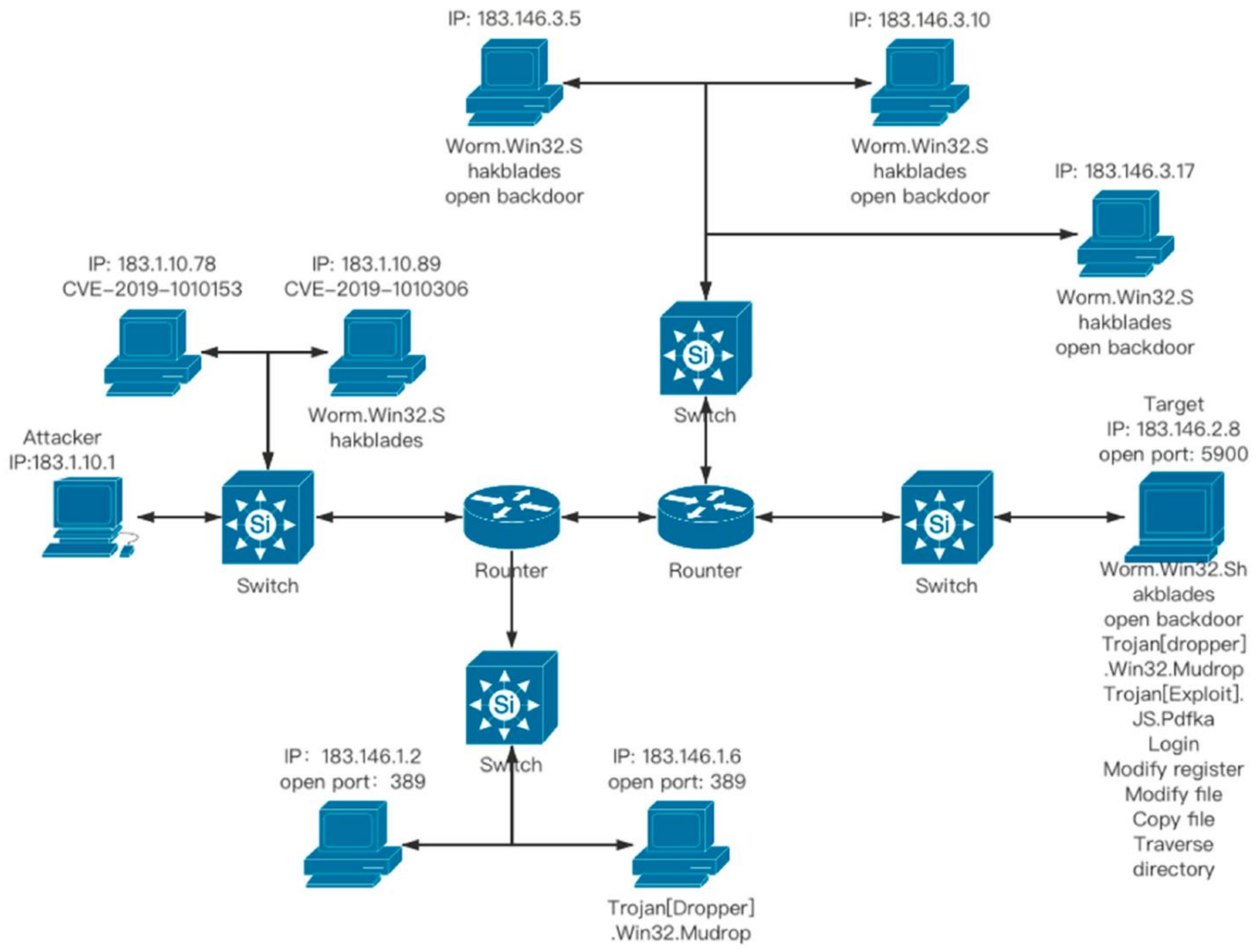
IP: 183.146.3.5
Worm.Win32.S hakblades open backdoor
IP: 183.146.3.10
Worm.Win32.S hakblades open backdoor
IP: 183.146.3.17
Worm.Win32.S hakblades open backdoor
IP: 183.1.10.78 CVE–2019–1010153
IP: 183.1.10.89 CVE–2019–1010306
Worm.Win32.S hakblades
Switch
Attacker IP:183.1.10.1
Switch
Rounter
Rounter
Switch
Target IP: 183.146.2.8 open port: 5900
Worm.Win32.Sh akblades open backdoor Trojan[dropper] .Win32.Mudrop Trojan[Exploit]. JS.Pdfka Login Modify register Modify file Copy file Traverse directory
Switch
IP: 183.146.1.2 open port: 389
IP: 183.146.1.6 open port: 389
Trojan[Dropper] .Win32.Mudrop

Fig. 3. Scheme of the test stand [9]

Creating an attack graph requires knowledge of network connectivity, running services, and information about vulnerabilities. This information is provided to the attack graph as input. Each time a new vulnerability is discovered or there are changes to the network connections or to the services operating through these connections, the information provided to the attack graph is updated and the old attack graph is changed to a new one. SGA provides information about possible ways that an attacker can use.

The next important stage of the research is the construction of the alert correlation graph scenario. ACG serves to confirm attacker behavior and helps identify false positives and false negatives.

To confirm the effectiveness of the VHS protection methodology, it is advisable to conduct an experiment in a private cloud environment. Penetration testing scenarios in a virtual private cloud environment were created using Metasploit [10] and Armitage [11] software as attackers. This approach allows emulating an attack from different locations on internal and external nodes, as well as launching various attacks based on vulnerabilities in each virtual machine. Fig. 4 presents VM vulnerability estimates O_{vm} and virtual machine vulnerability utilization estimates O_{Avm} [12].

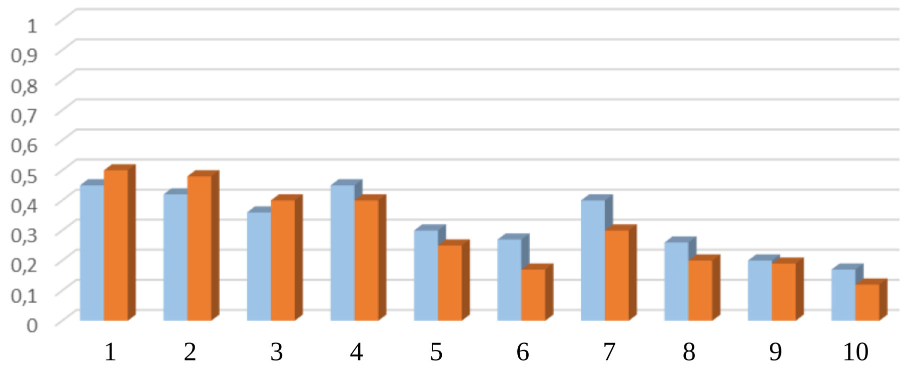

Fig. 4. Evaluation O_{vm} (blue) and O_{Avm} (orange) for virtual machines

The diagram shows how many vulnerabilities in VMs can be used by an attacker to compromise them. Note that the lower these indicators are, the lower the risk that the attacker will be able to achieve his goal. If one of the indicators is higher than 0.5, it can be assumed with high probability that the overall system protection indicator will also be high. This, in turn, means that there is a high probability of its compromise and appropriate countermeasures will be taken against it. VM estimates depending on the protection indicator are shown in Fig. 5.

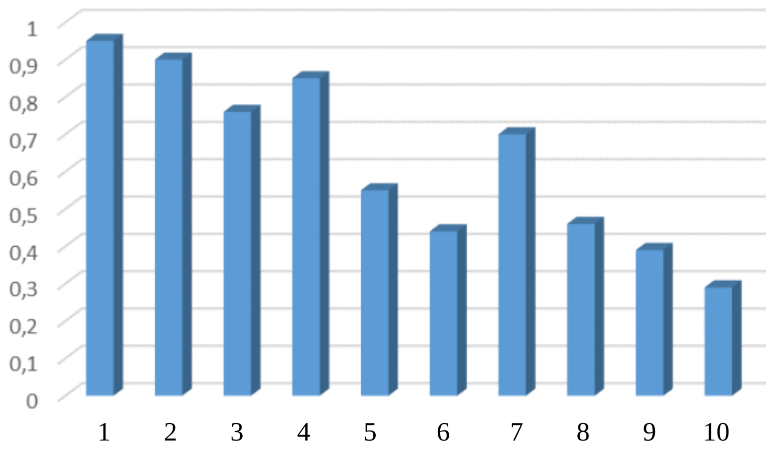

Fig. 5. Evaluation of virtual machines depending on the protection indicator

The diagram of the assessment of VMs depending on the protection indicator demonstrates the general state of protection in a private cloud environment. Note that the lower the value of the protection indicator, the higher the state of protection of the virtual machine in the private cloud environment.

Below is a diagram of the complex countermeasure selection indicator *Y* (Fig. 6), which reflects the magnitude of the negative impact that the countermeasure has on the SLA service level agreement. This indicator was proposed above and represented by formula (12). Also, this graph reflects the costs that are required to apply countermeasures in terms of resources and operational complexity (the higher the indicator, the greater the cost). These indicators together reflect the comprehensive countermeasure selection index *Y*. It should be noted that countermeasures with a high countermeasure selection index *Y* correspond to more serious and rarely used countermeasures that can significantly change almost the entire infrastructure of the virtual cloud environment.

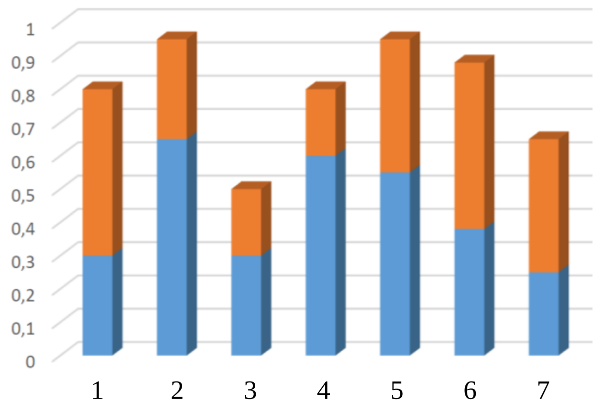

Fig. 6. Diagram of the complex indicator of countermeasure selection *Y* (blue – cost, orange – intrusiveness)

Fig. 7 shows the value of the virtual machine protection score for the test virtual machines before and after the countermeasure was adopted. The analysis of the diagram shows that the average rate of reduction in the vulnerability of virtual machines according to the F_{vm} indicator is 8%, which makes it possible to increase the efficiency of the virtual cloud environment.

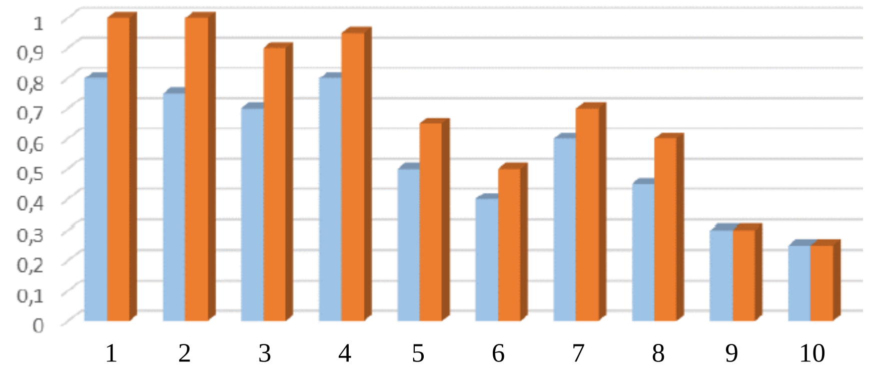

Fig. 7. Changing the F_{vm} protection indicator of the virtual machine before (blue) and after (orange) countermeasures

The larger the volume of the cloud environment that is planned to be used, the greater, according to the law of large numbers, the probability of false alarms. A cloud virtual environment system with hundreds of nodes will have a huge number of alerts generated by Snort. Therefore, it is necessary to develop an effective mechanism for checking the veracity of alerts. Since Snort can be programmed to generate an alert with a CVEID, it is recommended that you check the vulnerability database for the vulnerability. If this condition is met, then the existence of this vulnerability in the SGA means that a real attack is likely to be carried out. Thus, each new false positive will not lead to an increase in the percentage of non-productive use of resources of the virtual cloud environment.

It is impossible to correctly process alerts with a "zero-day" attack indicator, where a vulnerability detected by an attacker or system is not detected by a network vulnerability scanner. In this case, all alerts that have a "zero-day" attack identifier are considered real and need to be entered into the attack graph. Despite the fact that there is no corresponding node in the SGA. It is important to note that a vulnerability scanner should be able to detect the most recent vulnerabilities and sync from the latest vulnerability database to reduce the likelihood of zero-day attacks.

Conclusions

The developed method of providing protection of a virtual cloud environment detects and prevents intrusions into a distributed virtual network in a virtual cloud environment. It makes it possible to capture and inspect suspicious cloud traffic without interrupting the user's applications and cloud services. The technique uses an attack graph to detect attacks and prevent exploitation of vulnerabilities by correlating attack behavior and suggests effective intrusion countermeasures and optimizes implementation on cloud servers to minimize resource consumption. The use of this technique makes it possible to use a small amount of computing power.

The evaluation of system performance proves the possibility of the technique to reduce the number of compromises in the virtual cloud system from misuse by internal and external attacks. To confirm the adequacy of the obtained results, a test virtual cloud environment was developed, on which the developed methodology was tested. According to the results of the calculations, the average indicator of reducing the vulnerability of the virtual machine according to the F_{vm} indicator is 8%, which makes it possible to increase the efficiency of the virtual cloud environment.

References

1. Anatomy of a cloud storage infrastructure by M. Tim Jones http://public.dhe.ibm.com/software/dw/cloud/library/cl-cloudstorage-pdf.pdf
2. Rajan, Sreeranga & Ginkel, Wilco & Sundaresan, Neel & Bardhan, Anant & Chen, Yu & Fuchs, Adam & Kapre, Aditya & Lane, Adrian & Lu, Rongxing & Manadhata, Pratyusa & Molina, Jesus & Cardenas, Alvaro & Murthy, Praveen & Roy, Arnab & Sathyadevan, Shiju & Shah, Nrupak. (2013). Cloud Security Alliance report on the Top Ten Challenges in Big Data Privacy and Security. 10.13140/RG.2.1.1744.1127.
3. Shin, Gun-Yoon, Sung-Sam Hong, Jung-Sik Lee, In-Sung Han, Hwa-Kyung Kim, and Haeng-Rok Oh. 2022. "Network Security Node-Edge Scoring System Using Attack Graph Based on Vulnerability Correlation" Applied Sciences 12, no. 14: 6852. https://doi.org/10.3390/app1214685.
4. Hu, Hao & Liu, Jing & Zhang, Yuchen & Liu, Yuling & Xu, Xiaoyu & Jinglei, Tan. (2020). Attack scenario reconstruction approach using attack graph and alert data mining. Journal of Information Security and Applications, Volume 54, 2020, 102522.
5. Saad, Sherif & Traore, Issa. (2013). Semantic aware attack scenarios reconstruction. Journal of Information Security and Applications. 18. 53–67. 10.1016/j.jisa.2013.08.002.
6. I. Kotenko and A. Chechulin, "A Cyber Attack Modeling and Impact Assessment framework," 2013 5th International Conference on Cyber Conflict (CYCON 2013), 2013, pp. 1-24.
7. Booth, H., Rike, D. and Witte, G. (2013), The National Vulnerability Database (NVD): Overview, ITL Bulletin, National Institute of Standards and Technology, Gaithersburg, MD, [online], https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=915172 (Accessed December 21, 2022)
8. Su Zhang, Xinming Ou & Doina Caragea (2015) Predicting Cyber Risks through National Vulnerability Database, Information Security Journal: A Global Perspective, 24:4-6, 194-206, DOI: 10.1080/19393555.2015.1111961
9. Qi, Yulu, Rong Jiang, Yan Jia, and Aiping Li. 2020. "Attack Analysis Framework for Cyber-Attack and Defense Test Platform" Electronics 9, no. 9: 1413. https://doi.org/10.3390/electronics9091413
10. https://www.metasploit.com/
11. https://www.offensive-security.com/metasploit-unleashed/armitage-setup/
12. Вишнівський В.В. Дослідження технології захисту віртуального хмарного середовища на основі графа атак. Пояснювальна записка до магістерської роботи. – К.: ДУТ, 2021. – 78 с.

Надійшла: 26.12.2022 Рецензент: д.т.н., професор Гайдур Г.І.