Detection of repetitive and irregular hypercall attacks from guest virtual machines to Xen hypervisor

MOSTAFAVI, Mojtaba and KABIRI, Peyman <http://orcid.org/0000-0001-5143-0498>

DETECTION OF REPETITIVE AND IRREGULAR HYPERCALL ATTACKS FROM GUEST VIRTUAL MACHINES TO XEN HYPERVISOR

Mojtaba Mostafavi, Peyman Kabiri

***Abstract*— Virtualization is critical to the infrastructure of cloud computing environment and other online services. Hypercall interface is provided by hypervisor in order to offer privileged requests by the guest domains. Attackers may use this interface to send malicious hypercalls. In the reported work, repetitive hypercall attacks and sending hypercalls within irregular sequences to Xen hypervisor were analyzed and finally an Intrusion Detection System (IDS) is proposed to detect these attacks. The proposed system is placed in the host domain (Dom0). Monitoring hypercalls traffic the system operates based on the identification of irregular behaviors in hypercalls sent from guest domains to hypervisor. Later on, the association rules algorithm is applied on the collected data within a fixed time window, a set of thresholds for maximum number of all types of the hypercalls is extracted. The results from the implementation of the proposed system show 91% true positive rate.**

***Index Terms*— Hypervisor security, Xen, Hypercall, Virtualization, Hypercall security**

I. INTRODUCTION

IN recent years, cloud computing has been increasingly grown as a flexible and cost-effective technology to provide business and application services on the Internet or wide private networks. The risk of using cloud computing is higher than other conventional methods. This is because part of client resources (such as storages and processors) are owned and operated by a third party and thus protecting its confidentiality, integrity, and privacy will be difficult since cloud computing uses a combination of different technologies such as Virtualization, Web 2.0 and so on [1]. Therefore, it is natural to have vulnerabilities of these technologies inherited by the cloud computing. Virtualization is the most important technology used in cloud computing network infrastructures. As a result, many threats and attacks in the field of cloud computing are related to this technology. Threats in the field of virtualization are very wide and diverse. Hypervisor is one of the most valuable targets for hackers in the cloud computing. This is due to the fact that if the attacker takes the control over a hypervisor on a server, then he can access rest of all the Virtual Machines (VMs) on that server. Hypervisor i.e. Virtual Machine Manager (VMM), attacks can be generally categorized into two types: attacks from host OS to hypervisor and attacks from guest VMs to hypervisor. The reported work is intended to detect attacks from guest VMs to Xen hypervisor [2].

Xen is a well-known open source hypervisor widely used in cloud environments. The logic used in the Xen architecture is not complicated. Xen uses a privileged kernel to manage, monitor and control all guest VMs on a hardware server. This privileged kernel is called domain0 (or dom0) and all other domains managed by dom0 are known as guest domains (or unprivileged domains) [3]. Once the hypervisor is conquered, the attacker can either control VMs by accessing them or can force them to fail. In other words, Denial of Service (DoS) attacks can be successfully implemented in Dom0. In recent years, researches on the security of Xen VMs revealed that the virtual interfaces are vulnerable to different types of attacks [4][5]. Various researches and strategies are proposed for the security of VMs, among which, Intrusion Detection System (IDS), attack tracking, workload isolation and monitoring system are some examples. In this study, monitoring the hypercalls sent from VMs to the hypervisor is used to implement and detect a number of attacks on the Xen hypervisor, including repetitive hypercall attacks and irregular sequences in hypercall generation. Consuming hypervisor resources, these attacks often reduce the system performance. In other words, they are considered as a DoS attack. The proposed detection methods presented in this paper are a combination of signature-based methods and identification of anomalous hypercall sequences. Therefore, the proposed method can detect a variety of attacks. Efforts have been made to adapt search and sequence identification methods with the least overhead for making the proposed method applicable to large cloud computing networks.

In the second section of this paper the reported work is introduced and its significance is discussed. The third section explains the problem and classifies the existing hypercall attacks. The architecture of the proposed system is presented

M.Mostafavi is with the Iran University of Science and Technology (IUST), Tehran, 1684613114, Iran (email: mjt.mostafavi@gmail.com)

P.Kabiri is with the School of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, 1684613114, Iran (email: peyman.kabiri@iust.ac.ir)

in the fourth section. In fifth and sixth sections, the proposed methods to detect the irregular order and repetitive hypercall attacks are described, respectively. The seventh section is about the application of intrusion detection system in cloud computing environment. The eighth section talks about implementation of the attacks in the experimental environment and the experimental results. Conclusions and future works are presented in the final section.

II. RELATED WORK

In recent years, due to the high popularity and development of the cloud computing services throughout the world, many efforts have been made to increase security of the VMs. However, the domains available in virtual servers are still unreliable because of several reasons, including unawareness of developers about all existing threats and invention of new attack methods. Since Dom0 of hypervisor is a high-value target, it is targeted in many attacks. In general, cloud computing security can be considered from different aspects. For example, Chonka et al. [6] investigated DoS attacks based on XML and HTTP, and proposed a way to deal with these attacks. In another work Bacon et al. [7] proposed a solution for information flow control specifically in Platform-as-a-Service (PaaS) cloud services. Patel et al. [8] examined challenges of using Intrusion Detection and Prevention System (IDPS) in cloud computing environments. Hashizume et al. [1], studied cloud computing security from different aspects and classified vulnerabilities and threats in cloud environment. The work reported in the current text is aimed on the security of Xen virtualization infrastructures in cloud environment. In their reported work specifically considered hypercall attacks in Xen hypervisor and proposed a solution to handle certain types of these kind of attacks. In some of the earlier solutions recommended for hypervisor security, e.g. [9] and [10], intension was to reduce domains vulnerability by using the hypervisor to monitor them. But one cannot overcome all the threats using these methods, and some new exploits may be discovered that above approaches cannot prevent. In another reported work, Zhang et al. [11], assumed that Dom0 might be converted into a destructive domain. Since Dom0 has access to other domains, guest domains must be protected against this domain. They have proposed a mechanism called CloudVisor which is a small security monitoring system to protect these sources. Difficulty in commercialization is one of the most important challenges in this case. This problem is due to the fact that they need to apply changes on the existing virtualization architectures. Therefore, it will make their implementation difficult and may create problems for future hypervisor updates. Colp et al. [12], suggest a solution called Xoar that divides management domain into single-purpose components to reduce the likelihood of attackers' success in capturing hypervisor. Hoang in his thesis [13] proposed two different approaches to increase hypervisor security and to encounter hypercall attacks. The first strategy is authentication of hypercalls received by hypervisor using a MAC method and the other strategy is to implement a hypercall access table. Given the restrictions on the number of variables can be sent to hypervisor by a hypercall, it is not practically possible to use MAC method for the authentication of the hypercall. Implementation of access table has its own difficulties. The most important problem is access table volume that is proportionate to number of domains in virtual server. This is in contrast with the principle of Xen machines design concept. Jingzheng et al. [14] proposed a mechanism called XenPump that prevents time channel attacks. As the greatest problem of their proposed method, it will significantly reduce system efficiency and creates excessive lag in hypervisor performance. Bharadwaja et al. [15] designed and proposed an IDS called Collabra which works based on the detection of irregular behaviors in order to deal with hypercall threats. The proposed system allocates an anomaly score to every hypercall and if the score is above a specified threshold, the hypercall is considered anomalous and otherwise a normal hypercall. In another reported work, Wang et al. [16] proposed a new approach to make hypercall interface useless to the attackers. In their proposed method, legal hypercalls are first encrypted in the guest operating system using RC4 encryption algorithm and then sent to the hypervisor. Another plugin located in the hypervisor decrypts all incoming hypercalls and then delivers them to the hypervisor. Thus, if incoming hypercalls are not previously encrypted by the same key, they will be converted into unclear words, and consequently they cannot be processed by the hypervisor. This approach is costly and affects the performance of the system since it requires performing Encryption/Decryption operations repeatedly. In [17] an architecture was developed that split Dom0 privileges into two parts: First operations that can violate users privacy, removes to a per user management domain that called DomU Manage_VM. Other privileges was remain and form Thin Dom0. So in the event of a successful attack to DomU, only can compromise users in that DomU and no spread to other DomU. Also once Dom0 was compromised then attacker can't access to private information of DomU. But they do not address DDoS attacks in their work. In another work [18] they design a secure execution environment to ensure the confidentiality and integrity of guest virtual machine that running on untrusted management virtual machine. They categorized hypercalls that are used for management of DomU into three groups: 1. Hypercalls that are harmful to the privacy and integrity of DomU but not necessary, so should be prohibited. 2. Hypercalls hat are not harmful to the privacy and integrity, these hypercalls be unmodified. 3. Hypercalls that are harmful but necessary for management of DomU, so should be used with some restrictions. The proposed architecture use encryption and hash functions to protect integrity and privacy of virtual machine memory and vCPU. Milenkoski et al. [19] proposed a hypercall attack simulator software installed on a guest VM that can execute various hypercall attacks. Using this software, one can evaluate performance and efficiency of the IDS.

III. STATEMENT OF THE PROBLEM

Hypervisor is an attractive target for hackers and attackers.

Therefore, they constantly try to find new ways to attack hypervisors. Attacks may have different effects on the target system, such as preventing hypervisor operation, which can be considered a state of crash for the hypervisor or its clients. In fact, these attacks are in the category of DoS attacks. Some other hypercall attacks may disrupt the hypervisor state in such a way that the targeted hypervisor may not be crashed; instead it may be exposed to conditions desired for attacker. For instance, if the victim hypervisor's memory is in a certain pathway, then it might allow the attacker to implement a destructive code or high-level access operation. Hypercall attack may even cause access to the unauthorized information. For example, reading data from a portion of memory that is allocated to the hypervisor or some other guest VM. In rare cases, it is possible for the attacker to penetrate the system executing a specific hypercall (by executing a destructive code using hypercalls). A real example of this type of attack is the reported incident CVE 2017-8903 [20]. Hypercall attacks can generally be classified as follows [21]:

Running a single hypercall:

Executing a hypercall with an ordinary parameter (a perfectly legal hypercall)

Executing a hypercall with modified parameter(s) to exploit an available vulnerability.

Running series of hypercalls with a specific order:

Repetitive implementation of a specific hypercall.

Repetitive implementation of several consecutive hypercalls.

The second group of attacks, namely running a series of hypercalls, is also considered in this research. Anomaly detection methods are used to deal with these types of attacks. In recent years a variety of these attacks have been reported, for example, reported incidences such as CVE-2013-4494 [22], CVE-2013-1920 [23] worth mentioning. In reported work by Shropshire [24] he run successful CPU DoS attack from a malicious guest to hypervisor with a simple infinite loop that invoke hypercall 13. As a result malicious guest consumes 100% of CPU processing cycles. In [25] most of this vulnerabilities and also some others, was exploited using hInjector to evaluate IDSes in virtual environment.

In contrast to the approaches proposed by other researchers, the method proposed in this paper maintains the main concept of Xen hypervisor design [26] and applies the lowest possible changes into the system. Another important point here is that guest operating systems remain unchanged. According to the above, the proposed method causes no problems in updating either Xen hypervisor or guest operating systems.

There is a plugin in the solution proposed in this research located at Dom0 kernel responsible for monitoring all hypercalls sent by the guest VM. Once transformed into a suitable format, data generated by this plugin is sent to the IDS installed in the Dom0. The IDS learns rules and thresholds that exist in normal traffic as for their signature. Later on, this signature is used to identify any anomalous behavior within the received hypercall traffic."

IV. ARCHITECTURE

Hypercalls can be monitored in different ways and this monitoring can be performed in different parts of the Xen virtualization system. In this study, as presented in Fig.1, monitoring all the hypercalls is performed within Dom0 kernel. Monitoring hypercalls in Dom0 provides high efficiency, and low overhead.

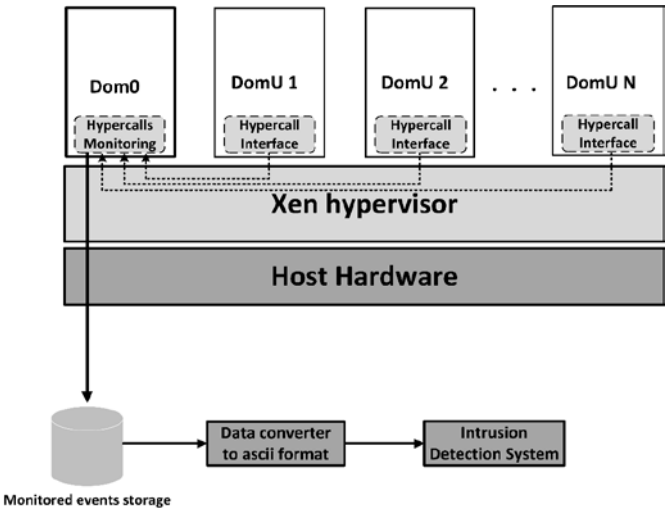


Fig. 1. Monitoring hypercalls sent to Xen hypervisor.

As depicted in Fig.1, all hypercalls sent from guest VMs to hypervisor are monitored in Dom0 using Xentrace plugin and stored in a storage area. Later on, stored data is converted to ASCII format using Xenalyze Plugin and finally it is send to IDS.

In the reported work, Xentrace and Xenalyze plugins [27] are used to monitor hypercalls in Dom0 kernel. These plugins are open-source and their coupled operation is needed for the hypercall monitoring.

Xen hypervisor has various features helping developers to capture events occurring within the hypervisor. Tracing the data, Xen stores them in special buffers and then Xentrace plugin, which has been installed and activated in Dom0, can read data from the buffers periodically and store them on hard disk for further use. Xentrace plugin is a part of Xen hypervisor project associated with each of its distribution. This tool is fairly simple, yet very useful and practical. Information stored by Xentrace is unreadable to humans and thus a tool is needed to convert it's data into a format that is readable for humans [27]. Xenalyze plugin is used to convert the data stored by Xentrace to the ASCII format which is usable by IDS. A sample of Xenalyze output that is converted to ASCII format is presented in Fig.2.

```
hypercall 1a (mmuext_op) eip c0101347
hypercall 1a (mmuext_op) eip c0101347
hypercall  d (multicall) eip c01011a7
hypercall  d (multicall) eip c01011a7
hypercall  e (update_va_mapping) eip c01011c7
hypercall  1 (mmu_update) eip c0101027
hypercall  1 (mmu_update) eip c0101027
```

Fig. 2. A sample of converted Xenalyze output.

Prior to the process carried-out by IDS, the captured data is preprocessed to filter excess items from the hypercalls. Any expression other than the hypercall's hexadecimal number is considered as an excess item. For example, pre-processed information is presented in Fig.3. Function of IDS is explained in sections 5 and 6.

```
1a  1a  d  d  e  1  1  1a
```

Fig. 3. A sample pre-processed information by the data obtained from Xenalyze plugin.

A. DETECTION OF SEND SEQUENCE OF HYPERCALLS WITH IRREGULAR ORDER

Hypercalls with irregular order do not have any typical appearances. Therefore, an intelligent method is needed to detect irregular sequences. The approach used in this study identifies anomalous sequences using apriori algorithm. Based on apriori algorithm, if {a,b,c} is a frequent itemset, then all of its subsets ({a,b}, {a,c}, {b,c}, {a}, {b}, {c}) be frequent [29]. In this paper frequent itemsets (hypercall sequences) and accordingly frequent rules are identified using apriori algorithm and then extract rules from them.

The first step in this procedure is to monitor and collect the hypercall traffic sent by VMs in an experimental environment within a relatively long period of time. It has been assumed that there were no attacks during this period. Hence one can call this traffic a white traffic. White traffic is used to train normal behavior of the environment to the IDS. Properties and behavior of the white traffic is assessed in two steps as described in the following sections.

1) The first step of hypercall traffic analysis

This analysis aims to identify abnormal hypercall sequences sent out by a VM in a Xen environment. Traffic analysis is very simple at this stage. Various hypercalls sent by all types of guest VMs in white traffic are tagged and stored in a table as normal hypercalls. Thereafter, whenever the IDS detects any hypercalls other than the values in this table, it will send a warning message to the system administrator. Two types of sabotage can be identified using this method. Type one is sending abnormal hypercalls undefined to Xen hypervisor that may aim to put the system in an undefined state. Type two is detection of legal hypercalls in the desired environment that are not regularly present in normal operating conditions and are assumed suspicious once detected. Table I provides samples of normal hypercalls.

TABLE I
SOME SAMPLES OF NORMAL HYPERCALLS IN THE XEN ENVIRONMENT.

21	18	e
11	a	1d

2) The second step of hypercall traffic analysis

Purpose of the hypercall traffic analysis is to identify attacks executed by sending several consecutive hypercalls with irregular order. Since detection of repetitive hypercall DoS attacks of a particular type (such as "d d d d d" sequence) is performed by another approach explained in the section 6, only non-repetitive orders are considered in this section. Therefore, a simple program is developed to receive the hypercall traffic as for input and stores all available non-repetitive orders in a file as for output. The output file of this program is input to the Weka software [30] that is used to run apriori algorithm and to easily produce the output with a suitable format. Due to a variety of monitored unique hypercall values, the basket length is specified dynamically based on the number of non-repetitive hypercalls sequences and each non-repetitive sequence is placed in a separate basket. The empty space of the basket is denoted by "?" and all the values of basket are separated by comma, which is in the standard format of Weka software. An example for hypercall traffic (on the top of the figure) conversion into the suitable input format for the Weka software (at the bottom of the figure) is presented in Fig.4. In the reported work, it is assumed that there are at the most 8 hybercalls in each line, indicating that in sampled file only 8 types of different hypercalls are transmitted to hypervisor. Thus, no line can be permutated with unique value patterns larger than 8.

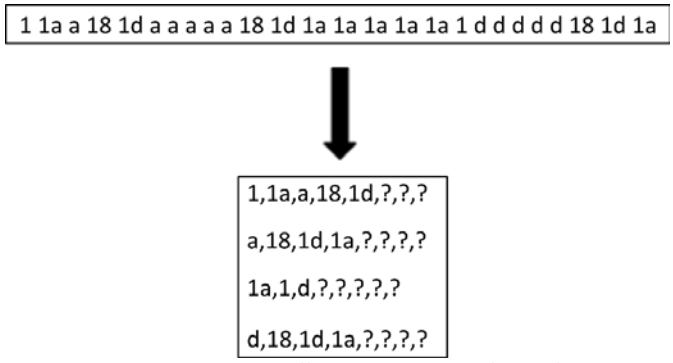


Fig. 4. Converting hypercall traffic to suitable input format for the Weka software.

Executing the program, the whole white traffic is segregated into numerous similar sequences and stored in an output file. Using Weka software and apriori algorithm, popular sequences in the traffic are identified. The repetitiveness of the sequences displayed by the apriori algorithm is determined by minimum support variable. Given the high variety nature of hypercalls traffic, the highest value for minimum support variable can be set to 0.5. In other words, often we can't find a hypercall sequence that has a support value greater than 0.5.

Examples of popular sequences obtained from apriori algorithm using Weka software running on a sampled traffic are presented in Fig.5.

```
hypercall2=11 hypercall3=18 hypercall4=1d hypercall5=21 12648
hypercall1=d hypercall2=18 hypercall3=a hypercall4=1a 15376
hypercall2=21 hypercall3=1d hypercall4=1a 23498
```

Fig. 5. Some examples of apriori algorithm output.

In Fig.5, the number at the end of each 'hypercall' word indicates that hypercall permutation in the basket. For example, hypercall2 indicates the second permutation in the basket. Reader should be reminded that each basket had 8 permutations. A number at the end of each line represents the number of occurrences of a given sequence in all the baskets. For example, the first set in Fig.5 has a population for 12,648 occurrences. In the final step, these repetitive items are considered as the determinant of traffic's normal behavior and stored as a signature. The IDS will use these signatures to separate normal and anomalous behaviors of the hypercall traffic within fixed time windows. Fig.6 shows the format of several sample signatures with different sequence lengths and how they are stored in the table.

e,1d,18,1a,1
11,d,24,1a,20
1a,18,d
24,1d

Fig. 6. A few signature examples of normal traffic.

The IDS records the traffic of hypercalls within fixed time windows and compares various hypercalls against items stored in normal hypercall table. An alert is generated once any unusual hypercall type is detected. In the next step, traffic is segregated into non-repetitive pieces and all the pieces are compared versus the signatures of normal traffic stored in a separate table that was generated during the training stage. If no match is found, the sequence will be identified as an anomalous sequence. Boyer-Moore algorithm [31] was used in this study as for the search algorithm.

B. DETECTION OF REPETITIVE OCCURRENCES OF SIMILAR HYPERCALLS ATTACKS

In this type of attacks, the attacker sends a large number of a certain type of hypercall (i.e. type 20) to the hypervisor and consumes hardware resources (CPU & memory). To detect this type of attacks as it was mentioned earlier, in this research IDS counts the number of occurrences of each type of hypercalls for a specified long period of time. This operation is performed within constant intervals to identify attacks and then considers the highest count of the each hypercall within a period of time as a threshold of that hypercall type occurrence in each time interval. Obtained thresholds are stored in a table such as Table II.

TABLE II
SAMPLE TABLE FOR HYPERCALLS OCCURRENCE THRESHOLDS.

Number of Hypercalls	Occurrence threshold
a	82
e	247
1a	36
1d	184

Determining the thresholds, entire traffic of the hypercalls within specified time windows are analyzed. Alerts will be generated once number of occurrences passes a predefined threshold.

C. APPLICATION OF IDS IN CLOUD COMPUTING ENVIRONMENT

As shown in Fig.7, the cloud computing infrastructure consists of many virtual hypervisors installed on different hardware servers [32]. In previous sections, the proposed approach for IDS was to install it on Dom0 of a number of servers to make it operate independently. One of the advantages of this approach is its low traffic overhead for each IDS. On the other hand, since every server may be assigned to a customer with certain needs, any one of the IDSs can be configured differently based on requirement of the servers they are installed on. Thus, one can expect that the number of

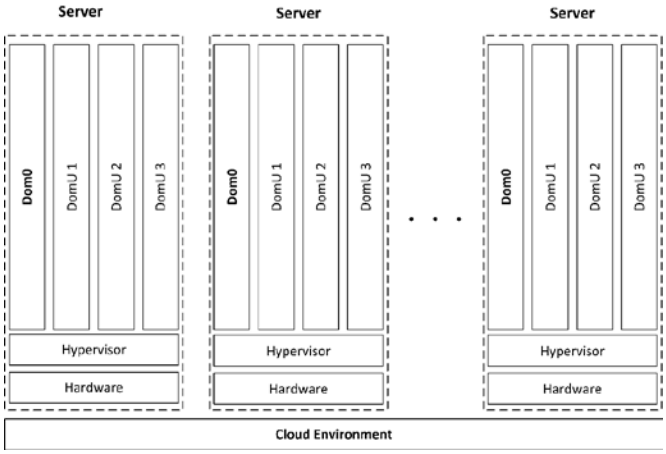


Fig. 7. Hypervisors in the cloud computing environment.

false positive and false negative alerts to reduce.

Another advantage of having independent IDSs on each hypervisor is that the traffic monitored by each IDS will be limited. Thus, speed and performance of the detection is increased.

V. EVALUATION RESULTS

In this section, results of the experiments and evaluation of the proposed IDS method is discussed using two scenarios. In the first scenario hypervisor is attacked by sending various hypercalls with irregular order. Second scenario is another type of attack on the hypervisor that is carried-out by sending repetitive hypercalls to the hypervisor.

A. The first scenario – sending various consecutive hypercalls with irregular order

In this scenario, as displayed in Fig.8, Xen hypervisor serves 5 virtual guest machines on a server with core i5 CPU and 4GB RAM. Each VM offers a different service to the users. One of these guest machines that is controlled by the attacker (one that is highlighted with a darker color), starts sending consecutive hypercalls with irregular order to disrupt Xen hypervisor operation or to reduce its performance. These attacks can lead to DoS as the order of hypercalls is out of the ordinary and not in a way hypervisor is expecting. This in turn causes hypervisor to encounter an unusual situation and to be unable to continue to serve its VMs. Hypercalls also may be sent with seemingly legitimate order, but in fact they are sending requests to the hypervisor that cause conflict and waste of resources. In the attack simulated in this research, VM number 2 starts sending several hypercalls with irregular order and IDS located in Dom0 must detect these attacks and generate alerts. Given that these attacks use special order of hypercalls, hypercall parameters are not important in this scenario. As mentioned in section 3, two examples of these attacks are reported as CVE-2013-4494 [22] and CVE-2013-1920 [23].

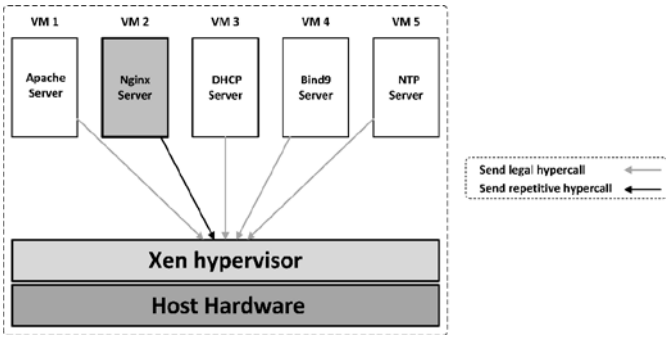


Fig. 8. Consecutive hypercall attacks with irregular orders.

1) Scenario implemented in the experiments and the results

An experiment that was conducted to implement this scenario and to evaluate performance of the IDS can be described in the following steps:

First step: Traffic of the hypercalls are monitored for 10 hours and saved in a file. Over eighteen million and five hundred thousand hypercalls were generated during this time which seems to be sufficient for analyzing the hypercall traffic behavior in normal conditions.

The recorded file that contains broken traffic data is fed into Weka software in order to implement apriori algorithm on its data. In the output of Weka, number of repetitive items with minimum support values of 0.5 and 0.3 is very limited. This result was expected and it is justifiable due to the wide variety of the hypercall sequences in the recorded hypercall traffic behavior. Therefore, the best way to cover diversity of the hypercalls traffic behavior is to use several different sets with lower support coefficients. According to the experiments, normal traffic behavior of hypercalls can be often separated with only 20 signatures from abnormal behavior with an acceptable accuracy (over 98%). In this study, a set of signatures with minimum support of 0.1 were used to extract the signatures. Signatures extracted from the output of apriori algorithm are reported in Table III.

TABLE III
SIGNATURES EXTRACTED FROM THE OUTPUT OF APRIORI ALGORITHM IN EXPERIMENT.

Nom.	Signature	Nom.	Signature
1	a , 1a	12	18, 1d, 21, 20
2	1a, d	13	18, 1d, 1a
3	d, 1a	14	18, 1d, d
4	d, 14	15	18, 1d, 21
5	18, 1d	16	d, 18, 1d
6	18, d	17	d, 24, 1a
7	1, 1a	18	14, 20
8	d, 18	19	20, 18
9	1d, 1a	20	18, 20
10	1d, d	21	24, 1a
11	d, 11		

Second step: The traffic related to the hypercalls generated within 10 windows of 10-seconds time interval was monitored in normal operating conditions and saved in a file. Later on, as for testing, different types of hypercalls that are recorded in these files are extracted and compared versus normal hypercalls in the table and no warning messages are generated. This traffic was analyzed using normal traffic signatures obtained in the 5th step. False positive rate is 0.76%. Here false positive rate represents wrong detections of a part of normal traffic as anomaly traffic.

Third step: As presented in Table IV, at random times in a 10 seconds time interval window, VM2 sends 11 irregular hypercall sequences to the hypervisor. As shown in Fig.9, the IDS issued 5 warning messages for anomalous hypercalls. The type of anomalous hypercall found at each message is displayed by the bracket signs "[ ]". Later on, these samples are analyzed using signatures that were obtained in step 1 and consequently total number of 17 warning messages were issued. 7 alerts out of 17 are false warnings generated before inserting the attacks (related to the 0.76% false positive rate for the reported approach).

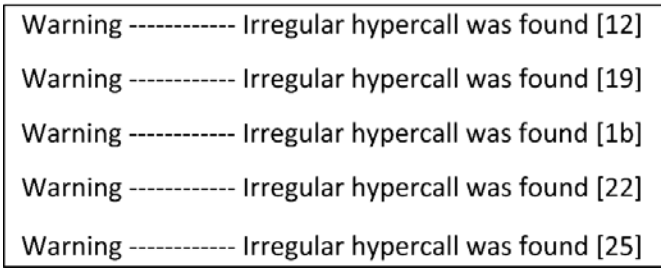


Fig. 9. Warning messages from anomaly hypercalls within 10 seconds time interval window after execution of the attacks.

TABLE IV
A SAMPLED IRREGULAR SEQUENCES SENT TO XEN HYPERVISOR.

No.	Irregular hypercalls sent to hypervisor by attacker
1	18, ,21, 1, d,18,18,18
2	22, d, 12
3	d, 21, 25, 11
4	24, a, 14, 21, 19, 1a, 1
5	1, 1d, 24, d
6	d, 21, 1d, 24, 18
7	1b
8	14, 1a, 21, a, 18
9	21, 18, 1a
10	14, 23, 23, 23, 23, 23
11	1d, 21, 24, 18

The irregular sequence 4 in Table IV (24, a, 14, 21, 19, 1a, 1) was not identified. Studying the cause, the recorded hypercall traffic was examined. This sequence is combined with its previous sequence (d, 1a, 24, a, 14, 21, 19) after being added to the initial traffic, and a new sequence is generated that matches with one of the signatures (d, 1a) given in Table III. Numbers of false and true warning messages are given in Table V.

TABLE V
CLASSIFICATION OF POSITIVE WARNING MESSAGES.

Alarm condition	Number of alerts
True Positive	10
False Positive	0
True Negative	915
False Negative	1

Using the resulted values, (1) to (5) are used to calculate error rates for the method proposed in this study.

True Positive Rate (TPR):

$$TPR = TP/(TP + FN) \quad (1)$$

Where TP is the number of true positive alerts and FN is the number of false negative alerts.

False Positive Rate (FPR):

$$FPR = FP/(FP + TN) \quad (2)$$

Where FP is the number of false positive alerts issued and TN is the number of true negative alerts.

True Negative Rate (TNR) or Specificity (SPC):

$$TNR \text{ or } SPC = TN/(FP + TN) \quad (3)$$

Where TN is the number of true negative alerts and FP is the number of false positive alerts.

Negative Predictive Value (NPV):

$$NPV = TN/(TN + FN) \quad (4)$$

Where TN is the number of true negative alerts and FN is the number of false negative alerts.

F1 Score:

$$F1 = 2TP/(2TP + FP + FN) \quad (5)$$

Where TP is the number of true positive alerts, FP is the number of false positive alerts and FN is the number of false negative alerts. Results are reported in Table VI.

B. The second scenario – Repetitive hypercalls

This scenario is implemented in an experimental environment similar to scenario 1, but this time the guest VM executes 2 attacks by sending a large number of hypercall 21 (physdev_op). The purpose of this attack is to consume or waste hypervisor resources by sending a large number of repetitive hypercalls. Therefore, hypervisor has to process all the incoming hypercalls that will lead to the waste of processor resources. Two examples of these attacks are reported as CVE-2015-7969 [33] and CVE-2015-7971 [34]. These attacks also have been referred to in various articles e.g. [19].

TABLE VI
THE REPORTED RESULTS.

Alarm condition	Number of alerts
True Positive Rate (TPR)	91%
False Positive Rate (FPR)	0%
True Negative Rate (TNR)	100%
Negative Predictive Value	99.8%
F1 Score	95%

1) Scenario implemented in the experiment and its results

This experiment is aimed on implementing a scenario to evaluate performance of the proposed IDS and can be described in the following steps:

First step: The traffic of hypercalls sent to hypervisor is monitored for ten hours. Later on, number of incoming hypercalls for each hypercall is counted within 5-seconds time interval windows and finally the count is stored in a table. A part of the table related to physdev_op hypercall is presented in Table VII.

TABLE VII
NUMBER OF INCOMING PHYSDEV_OP HYPERCALLS WITHIN 5-SECONDS TIME INTERVAL WINDOWS.

Time	Hypercall type	Received count
[0-5]	21	29
[5-10]	21	6
[10-15]	21	7
[15-20]	21	29
[20-25]	21	7
[25-30]	21	11

Second Step: Evaluating the performance of IDS, hypercall 21 was sent 100, 85, 70, 60 and 50 times for 5 times at random intervals, respectively, where the IDS detected the attacks in 4 of 5 cases. Only one attack was not identified. In the 4th attack, 60 hypercalls of type 21were sent out. In fact, total number of both malicious and legitimate hypercalls (the equivalent of 68 hypercalls) was lower than the threshold, i.e. 74. Thus, no alert is generated.

VI. CONCLUSION AND FUTURE WORK

Hypervisors in cloud computing environment are attractive targets for attackers and this is why their protection against security threats is very important. One of the common methods used to attack hypervisors is to use hypercall interface to send malicious hypercalls by guest VMs. The reported work investigated repetitive hypercall attacks and attacks carried-out by sending hypercalls with irregular order. In the reported work, hypercalls traffic in Dom0 is monitored. Executing a series of preprocesses, monitored traffic was broken into several non-repetitive rules. Apriori algorithm was applied to the resulted rule set. Number of repetitive hypercalls in each sequence defined is set to predefined

thresholds. In the last step, results are used to extract signatures of regular hypercalls behavior. Finally, the proposed IDS can detect irregular hypercall behaviors using these signatures.

The proposed method used in this paper searches for the signatures obtained from normal traffic for the exact match within a given traffic. A suggestion for future research is to use an algorithm that can match signatures using similarity threshold between existing signatures and different parts of hypercall traffic. If the similarity of each part of the traffic with existing signatures is more than the threshold, it would be considered as regular hypercall traffic; otherwise, it is an anomalous traffic. Using this method, it is expected to have a reduced false positive rate.

On the other hand, the approach reported in this paper only considers order of hypercalls and number of their repetitive generations. Another proposal to improve this approach is to the sent hypercalls parameters in addition to their order and number of repetitions to detect anomalies. Thus, it can be expected to have the proposed IDS covering more types of attacks.

REFERENCES

[1] K. Hashizume, D. Rosado, E. Fernández-Medina, and E. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications, vol. 4, pp. 1-13, 2013/02/27 2013.

[2] M. Zheng, "Virtualization Security in Data Centers and Clouds," 2011.

[3] W. von Hagen, Professional Xen Virtualization: Wiley Publishing, 2008.

[4] B. Cully, "The virtual monkey monitor," Technical report2006.

[5] T. Ormandy, "An empirical study into the security exposure to hosts of hostile virtualized environments," Technical report2007.

[6] Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," Journal of Network and Computer Applications, vol. 34, pp. 1097-1107, 2011.

[7] J. Bacon, D. Eyers, T. F. J. M. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch, "Information Flow Control for Secure Cloud Computing," IEEE Transactions on Network and Service Management, vol. 11, pp. 76-89, 2014.

[8] Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino Júnior, "An intrusion detection and prevention system in cloud computing: A systematic review," Journal of Network and Computer Applications, vol. 36, pp. 25-41, 2013.

[9] Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, et al., "Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems," ACM SIGARCH Computer Architecture News, vol. 36, pp. 2-13, 2008.

[10] Z. Wang, X. Jiang, W. Cui, and P. Ning, "Countering kernel rootkits with lightweight hook protection," presented at the Proceedings of the 16th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2009.

[11] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization," presented at the Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 2011.

[12] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, et al., "Breaking up is hard to do: security and functionality in a commodity hypervisor," presented at the Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 2011.

[13] C. Hoang, "Protecting Xen Hypercalls: Intrusion Detection/Prevention in a Virtualized Environment," MSc, Computer Science, University of British Columbia, 2009.

[14] W. Jingzheng, D. Liping, L. Yuqi, N. Min-Allah, and Y. Wang, "XenPump: A New Method to Mitigate Timing Channel in Cloud Computing," in IEEE 5th International Conference on Cloud Computing (CLOUD) 2012, 2012, pp. 678-685.

[15] S. Bharadwaja, S. Weiqing, M. Niamat, and S. Fangyang, "Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System," in Eighth International Conference on Information Technology: New Generations (ITNG) 2011 2011, pp. 695-700.

[16] F. Wang, P. Chen, B. Mao, and L. Xie, "RandHyp: Preventing Attacks via Xen Hypercall Interface," in Information Security and Privacy Research. vol. 376, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 138-149.

[17] C. Yu, L. X. Li, K. Wang, and W. T. Yu, "Protecting the Security and Privacy of the Virtual Machine through Privilege Separation," Applied Mechanics and Materials, vol. 347-350, pp. 2488-2494, August 2013 2013.

[18] C. Li, A. Raghunathan, and N. K. Jha, "Secure Virtual Machine Execution under an Untrusted Management OS," in IEEE 3rd International Conference on Cloud Computing, 2010, pp. 172-179.

[19] Milenkoski, B. D. Payne, N. Antunes, M. Vieira, and S. Kounev, "HInjector: Injecting Hypercall Attacks for Evaluating VMI-based Intrusion Detection Systems," presented at the Annual Computer Security Applications Conference (ACSAC), 2013.

[20] (2017, 05 Nov). National Vulnerability Database (NVD). CVE-2017-8903. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-8903

[21] Milenkoski, B. D. Payne, N. Antunes, M. Vieira, and S. Kounev, "Experience Report: An Analysis of Hypercall Handler Vulnerabilities," in IEEE 25th International Symposium on Software Reliability Engineering (ISSRE) 2014, 2014, pp. 100-111.

[22] (2013, 27 Feb). National Vulnerability Database (NVD). CVE-2013-4494. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-4494

[23] (2013, 27 Feb). National Vulnerability Database (NVD). CVE-2013-1920. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1920

[24] J. Shropshire, "Hyperthreats: Hypercall-based DoS attacks," in IEEE SoutheastCon 2015, 2015, pp. 1-7.

[25] Milenkoski, B. D. Payne, N. Antunes, M. Vieira, S. Kounev, A. Avritzer, et al., "Evaluation of Intrusion Detection Systems in Virtualized Environments Using Attack Injection," in Research in Attacks, Intrusions, and Defenses: 18th International Symposium, RAID 2015, Kyoto, Japan,November 2-4, 2015. Proceedings, H. Bos, F. Monrose, and G. Blanc, Eds., ed Cham: Springer International Publishing, 2015, pp. 471-492.

[26] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, et al., "Xen and the art of virtualization," presented at the Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, USA, 2003.

[27] D. Faggioli. (2012, 08 Apr). Tracing with Xentrace and Xenalyze. Available: https://blog.xenproject.org/2012/09/27/tracing-with-xentrace-and-xenalyze/

[28] (2013, 27 Feb). Vulnerability Summary for CVE-2013-1920. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1920

[29] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to datamining, 1st ed.: Pearson, 2005.

[30] (2013, 10 Apr). Weka 3: Data Mining Software in Java. Available: http://www.cs.waikato.ac.nz/ml/weka/

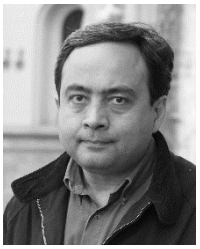
[31] R. Sedgewick and K. Wayne, Algorithms, 4th ed.: Addison-Wesley Professional, 2011.

[32] F. Rocha, T. Gross, and A. Van Moorsel, "Defense-in-Depth Against Malicious Insiders in the Cloud," in IEEE International Conference on Cloud Engineering (IC2E) 2013, 2013, pp. 88-97.

[33] (2015, 30 Oct). National Vulnerability Database (NVD). CVE-2015-7969. Available: https://nvd.nist.gov/vuln/detail/CVE-2015-7969

[34] (2015, 30 Oct). National Vulnerability Database (NVD). CVE-2015-7971. Available: https://nvd.nist.gov/vuln/detail/CVE-2015-7971

Mojtaba Mostafavi received M.Sc. in the Computer networks architecture from Iran University of Science and Technology on February 2015. His research interests include network security and forensics.



P. Kabiri was born in Tehran-Iran in 1968. He received B.Sc. in Computer Hardware Engineering from Iran University of Science and Technology, Tehran, Iran in 1992. M.Sc. in real time systems and Ph.D. in Computer Science both from the Nottingham Trent University, Nottingham, UK, in 1996 and 2000, respectively.

Currently, he is assistant professor at the School of Computer Engineering, Iran University of Science and Technology where he is actively pursuing research in different areas of computer science/engineering. He is a member of computer hardware engineering group in school of computer engineering. He is director of the Intelligent Automation Laboratory (IAL) at the school of computer engineering.