Community Security Information as Service in OpenStack

Wei Liu*, Qingni Shen, Wu Luo, Tianyuan Cao, Yaoyuan Zhang, Zhonghai Wu
School of Software and Microelectronics
Peking University
Beijing, China
{liuwei_pku, qingnishen, lwyeluo, tianyuan, Yaoyuan, wuzh}@pku.edu.cn

***Abstract*—With the rapid evolvement of cloud computing industry, security issues draw a lot of attention of the cloud service provider. As the most popular open-source cloud platform, OpenStack continuously improves its own security through a variety of security activities in the community. However, it is not easy to locate and make full use of these valuable security information scattered in the community. In this paper, we propose a community-driven approach by collecting and organizing security information to build a unified security information access portal, we call it SIR (Security Information Repository). Based on the SIR, we explore its application in intrusion detection and put forward the idea of extracting abnormal behavior characteristics from bug reproduction steps. With the intrusion detection scheme, we also present a SMSA (security migration scheduling algorithm) to prevent migrated-based attacks. The experimental results show that SIR effectively improve the utilization of security information in the OpenStack community and SMSA achieves good protection with acceptable overload.**

Keywords—cloud computing; openstack security; security information extraction; abnormal behavior recognition; migration security

I. Introduction

Cloud computing enables on-demand provisioning of elastic computing resources on a pay-as-you-go basis, which revolutionizes the information and communications technology industry. Despite the potential gains achieved from the cloud computing, its security is still questionable which impacts the cloud model adoption [1], [2], [3]. The security problem becomes more complicated under the cloud model as new dimensions have entered into the problem scope related to the model architecture, multi-tenancy, elasticity, and layers dependency stack [4], [5].

Although cloud security is such an arduous task, the OpenStack community has made a lot of effort to improve security. It uses RBAC (role-based access control) methods to provide authentication and authorization in the multi-tenant environment [6], [7]. In order to protect virtual machines and virtual routers, OpenStack uses iptables feature in the linux kernel to manage the flow and forwarding of network packets [8]. The community also regularly publishes security-related information on websites and mailing lists, such as security patches for new vulnerabilities, security instructions, security blogs, and so on.

There are also many studies that focus on OpenStack security, for example, access control [9], [10], [11], [12], compliance [13], [14], [15], [16], [17], network security [18], [19], [20], virtual machine security [21], [22], etc. However, only a small part of the work utilize the rich information in the community. Tong et al. [23] proposes a prediction method based on long tail model to characterize and predict bug assignment in OpenStack community. Robles et al. [24] presents an approach to estimate development effort by mining OpenStack software repository. They only use information from a particular activity in the community.

In fact, there are several security-related activities in the community, each of which contains different security theme. The reason why these security information are not being fully exploited is that they are scattered in the community. Meanwhile, version of the OpenStack community has changed rapidly [25], with different versions facing different security threats. These issues all increase the difficulty of accessing and using security information in OpenStack community and decrease the OpenStack cluster's security. Therefore, it is very meaningful to establish the SIR (Security Information Repository) for OpenStack.

Establishing SIR faces many challenges. Firstly, since there are many types of activities carried out in the community every day, it is essential to have a comprehensive and deep understanding for the community to identify security-related activities from fragmented activities. Secondly, different security activities are recorded in different formats, such as chat records, emails and archives. How to understand these information is a challenge. Thirdly, each security activities focus on exact contents, which have overlapping and different security information. To distinct and complement the security information, and establish the unified archiving, we will throughly analyze each data source.

In the paper, we present a community-driven security information mining method that can obtain the latest security information from the security-related activities in the OpenStack community and construct a secure information repository dedicated to OpenStack. In order to verify the validity of this method, a security component Miper is implemented according to the OpenStack component development standard. Using the SIR established by Miper, a migration security scheduling algorithm based on dangerous behavior recognition is proposed to explore SIR's application scene in dangerous behavior recognition. To the best of our knowledge, this is the first concrete realization of Security Information as Service in OpenStack.

Key contributions in this paper are:

- Summarizing the security-related activities in OpenStack community.
- Presenting a community-driven security information mining method in OpenStack community.
- Designing and implementing a security information as service component, Miper.
- Exploring the application of SIR in dangerous behavior recognition.

The remainder of the paper is organized as follows: Section II reviews the background about OpenStack Community and security related activities in the community. Section III presents the security information extraction method for OpenStack community. In Section IV, we implement a security information as service component, Miper, and propose a virtual machine security migration scheduling algorithm to verify the flexibility of Miper. Then the SIR and security migration scheduling algorithm are evaluated in Section V. Related works are discussed in Section VI. We conclude in Section VII at end.

II. Background

OpenStack is a global community consisting of developers, corporations, service providers, researchers, and users. According to the OpenStack community statistics [26], there are more than 40 public cloud service providers using OpenStack as the foundation platform in the world. Besides, OpenStack has been widely applied in finance, quantum physics, biological science and other fields.

The OpenStack Foundation is responsible for all management of the community. As the independent home for OpenStack, the Foundation is made of thousands of individual members and hundreds of different organizations from all over the world, secured more than USD 10 million in funding and is ready to become the ubiquitous cloud computing platform. In the community, Security Project undertakes both technical and governance activities within OpenStack, aiming to provide guidance, information and code that enhances the overall security of the OpenStack ecosystem [27].

In order to understand the security posture of OpenStack, an insight into existing security projects is imperative. The OpenStack Security Project has a number of ongoing activities that aim to enhance security of the OpenStack cloud ecosystem. For developers, community provides several security tools to improve the security of OpenStack, such as Bandit, which is a Python source analysis framework that can be used for security analysis of Python code. More security activities in the community are for cloud operators and security administrators. These predominantly break down into two groups: Advisory and Guidance.

Advisory. OpenStack community provides both OSSA (OpenStack Security Advisories) and OSSN (OpenStack Security Notes) for OpenStack users and vendors. OSSA is created to deal with severe security issues in OpenStack for which a fix is available. OSSA are issued by the OpenStack Vulnerability Management Team. Similar to advisories, OSSN are used for security issues which do not qualify for an OSSA, typically design issues, deployment and configuration vulnerabilities. They often address vulnerabilities in third party tools which are typically used within OpenStack deployments and provide guidance on common configuration mistakes that can result in an insecure operating environment.

Guidance. Security Guide and Security Blog are written by security experts to help developers and downstream consumers enhance the security of OpenStack. The guide provides best practices and conceptual information about securing an OpenStack cloud. And the blog offers latest of what has been happening in the OpenStack Security world.

For effective communication, contributors in Security Project regularly discuss security-related activities on IRC weekly. Important security activities are notified in the developer mailing list. Besides, contributors meet up at each OpenStack Summit and hold their own mid-cycle meet-ups too.

In OpenStack community, information in these activities has close relationship with bug repository. The community sets up a self-contained bug reporting and tracking mechanism. From submit, confirm, assigned, to the repair, the whole process will be detailed records by the community. Most of the content discussed in the security activity has a corresponding bug record in the bug repository, we call it security-related bug. Fig.1 shows the relationship between these activities and bug repository.

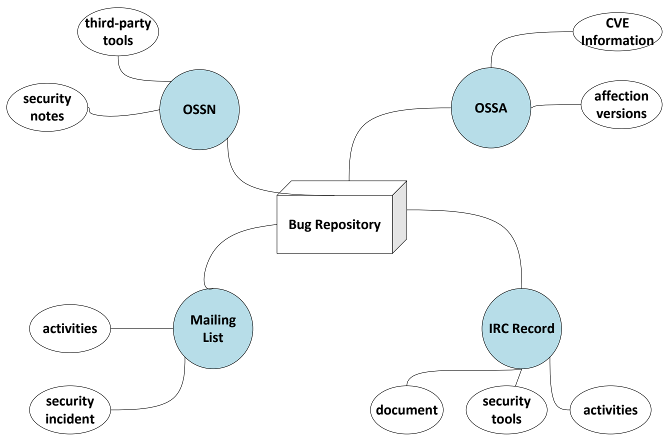


Fig. 1. The relationship between security activities and bug repository

From the introduction to the OpenStack Security Project, we can learn that the community is increasingly focusing on the security of OpenStack, and it has arranged for specific organizations to be responsible for OpenStack security-related activities. However, for this huge project with multiple releases, the security information updates do not have uniform push program. This poses a huge challenge for the security administrator to obtain specific versions of security information in a timely manner.

III. Approach

A. Overview

This paper presents a method for collecting security information from security-related activities. Under the guidance of this method, the OpenStack SIR is established as a unified portal for obtaining OpenStack security information. SIR contains security-related bugs, security patches, suggestions, and so on. The security-related bug repository is a refactor of the OpenStack community bug repository, which extends the bug's security attributes.

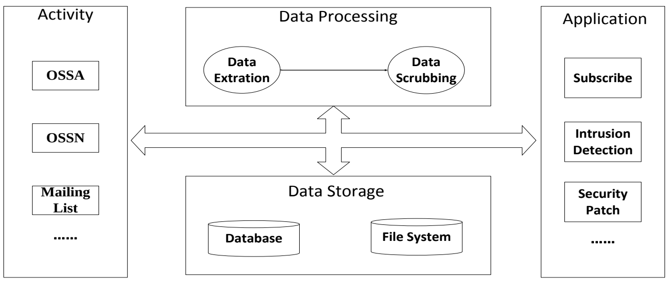


Fig. 2. Scheme of extracting security information in OpenStack Community

Fig.2 shows the scheme of establishing OpenStack SIR, the original data is extracted from security activities. We concentrates on Advisory and communication information exist in the community, including OSSA, OSSN, Mailing List and IRC Record. The safety information in these activities is updated quickly, and the content is edited by security specialists, including the latest security developments in the community. As for Guidance, it can be easily obtained from the Internet, and has slow update frequency, so there is no need to merge it into SIR.

After extracting the original data, the data processing is triggered to unify the formats of information through data extraction and data scrubbing. This paper employs a combination of database and file system to store SIR, in which the metadata is stored in the database, and the larger size information is stored in the form of file on disk. Various applications can be carried out based on SIR, such as message subscriptions, intrusion detection, security patch updates, and so on.

B. Data Extraction

Different security activities in the OpenStack community have different concerns. Therefore, if we want to obtain security information, firstly we must identify the primary fields that are valuable in each security activity. Security information in security activities is generally semi-structured data or even unstructured data.

In addition, each data source is different in format. It means that we need to use a specific data extraction scheme to obtain information from the security activity. OSSA and OSSN exist in the form of html, so security information can be extracted from the tag of the html page. The discussion on security-related bug exists in the mailing list and IRC records in MIME format and TXT format. IRC Records utilize regular expressions for keyword matching to extract the URL corresponding to the security issues discussed. As for mailing list, some key messages in accordance with the MIME standard will be extracted, including mail title, date and text. Table I shows details what we are looking for in each security activity.

C. Data Scrubbing and Archive

Data obtained from four data sources need to be preprocessed and unified. Among the four data sources, OSSA is of the most valuable security information. When processing OSSA raw data, it's important to pay attention to the complementary of information. For example, the threat type in CVE can supplement bug repository. Meanwhile, the CVE id, threat level and other information in CVE repository are recorded. For OSSN, we need to archive security problems according to the version. In IRC chat records, we focus on the bug, blueprint link and the associated code file path. In the mailing list, official participants often discuss on a specific security-related bug for many times, so some information maybe repetitive. This paper gives two methods to remove duplication.

1. Removing mails whose header has "RE", which is generally a reply to a security bug before, so these mails do not need to be analyzed, but the number of 'RE' is recorded as supplement of heat indicator.

2. Setting bug id as the primary key to avoid repeated storage of security-related bug.

This paper takes into account the relationship between four data sources and bug repository to summary the security-related bugs' attributes which need to be archived. As shown in Table II, a security-related bug consists of three parts, general bug information, security activity information and extended security information.

TABLE II. Security-related bug information

Name	Definition	Content
X	General bug information	id, title, bug description, discussion, importance, component, review url
A	Security activity information	activity type, date, description, reference
E	Extended security information	vulnerability information, vulnerability type, reproduce description, CVE

We can use the following expression to describe a specific security-related bug.

$$S = \{(\alpha, \beta, \gamma) \mid \alpha \in X, \beta \in A, \gamma \in E\}$$

TABLE I. PRIMARY FIELDS IN THE SECURITY ACTIVITY

Activity	Field	example
OSSA	id	OSSA-2015-017
	date	2015-09-01
	title	Nova may fail to delete images in resize state
	description	George Shuklin from Webzilla LTD and Tushar Patil from NTT DATA, Inc independently reported a vulnerability in Nova resize state…
	reference	https://security.openstack.org/ossa/OSSA-2015-017.html
	affected_scope	Nova: 2014.2 versions through 2014.2.3, and 2015.1 versions through 2015.1.1
	cve_ref	http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3280
	bug_ref	https://launchpad.net/bugs/1392527
	patch	https://review.openstack.org/219301 (Juno) https://review.openstack.org/219300 (Kilo) https://review.openstack.org/219299 (Liberty)
OSSN	id	OSSN-0063
	title	Nova and Cinder key manager for Barbican misuses cached credentials
	date	2016-6-9
	summary	During the Icehouse release the Cinder and Nova projects added a feature that supports…
	affected_scope	Cinder: Icehouse, Juno, Kilo, Liberty Nova: Juno, Kilo, Liberty
	discussion	The Barbican key manager is a feature that is part of Nova and Cinder to allow those projects to create and retrieve keys in Barbican. The key manager includes a cache function that allows for a copy_key() operation to work while only validating the token once with Keystone……
	recommended actions	Users wishing to use the Barbican key manager to provided keys for volume encryption with Nova and Cinder should ensure they are using a patched version…
	bug-ref	https://bugs.launchpad.net/glance/+bug/1523646
	patch-ref	Nova patch for Mitaka : https://review.openstack.org/254358/ Nova patch for stable/liberty: https://review.openstack.org/288490 Cinder patch for Mitaka : https://review.openstack.org/254357/ Cinder patch for stable/liberty: https://review.openstack.org/266678 Cinder patch for stable/kilo: https://review.openstack.org/266680
Mailing List	title	[Openstack-security] [Bug 1575909] Fix proposed to horizon (stable/newton)
	date	2017-3-3
	review-ref	https://review.openstack.org/440734
	branch	stable/newton
	status	fix released
	bug-id	1575909
IRC Record	date	2017-01-05-16.59
	reference	http://eavesdrop.openstack.org/meetings/security/2017/security.2017-01-05-16.59.log.html
	summary	* Elections (hyakuhei, 17:28:49) * LINK: https://governance.openstack.org/election/#how-to-submit-your-candidacy (sigmavirus, 17:30:59) * PTL Nominations are open from 18 Jan 2017 23:59 UTC until 29 Jan 2017 23:45 UTC (sigmavirus, 17:31:40) * LINK: https://releases.openstack.org/ocata/schedule.html#pike-ptls-self-nomination (hyakuhei, 17:32:19)

In the process of archiving, it's difficult to deal with bug description and the review URL of the patch. Bug description is submitted by bug founder so that the community can have a comprehensive understanding of the bug. The bug description and discussion vary widely in length, and some even have several supplementary attachments. So the combination of database and file system is necessary.

Another challenge is to extract the review URL from the multi-layer discussion con-tent. Fixing bug requires discussion and revise for several times. Therefore, there will be a lot of discussion information and multiple submitted re-view URL on bug home page. The paper analyses lots of review URLs to distinguish which one is the final adoption of the community. We find that these review URLs has two features different from others. One is the URL publisher, i.e. the community. The other is the context of the URL, which has specific phrases such as "fix" or "changed in".

Fig.3 shows the whole picture of SIR. SIR stores security information from the community in two dimensions. IRC Records and Mailing List are archived in time dimension, OSSN and OSSA are archived in version dimension. As the core of SIR, the security-related bug associates with all security activities and related attachments are stored in the file system.

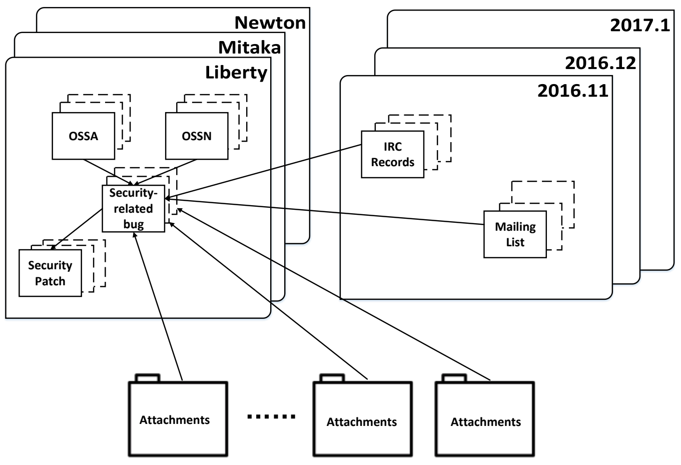


Fig. 3. Whole Picture of SIR

IV. Application

SIR can be widely utilized to improve the security of OpenStack cluster, such as security event subscription service, security patches update, dangerous behavior recognition and so on. This paper explores its effort on dangerous behavior recognition. We implement a security information as service component, Miper. And a virtual machine security migration scheduling algorithm is also implemented based on the extracted feature sequences of dangerous operation sequences. The goal of Miper is to increase the difficulty of expanding the damage scope via virtual machines migration without significantly decreasing the migration efficiency.

A. Framework

As a standard component of OpenStack, Miper can seamlessly interface with other standard components in OpenStack. The overall architecture of this component is shown in Fig 4.

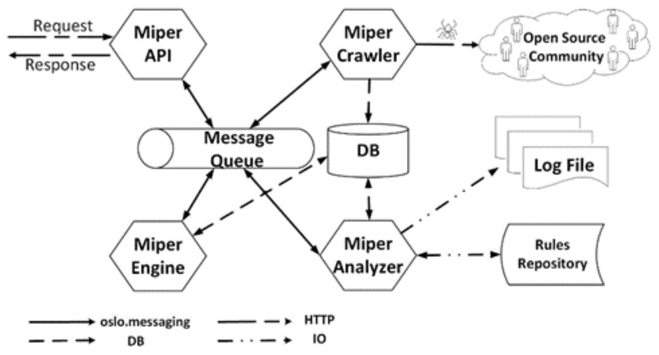


Fig. 4. Framework of Miper

Miper component is constituted of Miper API, Miper Crawler, Miper Engine and Miper Analyzer. Like other OpenStack Component, Miper API receives Restful-style requests and forwards them according to pre-defined routing rules.

Miper Crawler is responsible for regularly crawling security-related information in the open-source community according to section 3. Miper Crawler also extracts security-related bugs from bug repository, such as the bug with security tag. In fact, SIR is initialized and constantly updated by Miper Crawler.

Miper Analyzer aims at recognizing whether the user in cloud performed dangerous behaviors recently. Miper Analyzer is composed by two task flows: users' behaviors collector and dangerous operations' extraction. Prior task flow is to extract the user's recent operation requests from the system log, translate the requests into concrete actions to match the dangerous behavior characteristics in the rules repository and store the matching results in the database. The second task flow is to extract the features of dangerous operation sequences from SIR.

Miper Engine is the brain of the entire component to implement all the application logic. In this paper, Miper engine is responsible for making decisions about migration requests and includes the security migration scheduling algorithm.

B. Dangerous Operation Sequence Extraction

We can easily access complete security-related bugs once the SIR is set up. Most security-related bugs' description field contains instructional steps to reproduce the bug, and some also attach automatic scripts. These reproducing steps are defined as dangerous operations. Therefore, features can be extracted from the reproduction description to establish OpenStack's dangerous operation library. The dangerous operations can be divided into three categories according to the steps needed for reproducing:

- Non-step dangerous operations. Operations happened due to configuration errors, and no sequence of operations can be extracted.
- Single-step dangerous operation. The operation corresponds to single operating characteristic, usually shown in traditional dangerous behavior.
- Multi-step dangerous operation. Such operation corresponds to a series of mutually dependent operational sequences.

The paper focuses on multi-step dangerous operations. It is difficult to identify by the traditional intrusion detection system [28], [29], [30], [31]. Because each step of these operations is normal operation. However, once these operations are executed as a specific sequence, damage on OpenStack cloud platform will occur.

A semi-automatic approach is utilized to extract features towards the multi-step dangerous operation. The approach includes two steps:

1. The first step is to identify whether the bug description has keywords "reproduce". If exists, the all content after this word will be extracted, and then filter the contents whose first line is digital. Contents acquired are the detailed steps for re-producing.
2. The second step is to extract characteristic of operation sequence. Because the current Natural Language Processing (NLP) technique is limited and this step re-quires experts' experience, it can only be done manually. The main technique is to express the characteristic of the operation sequence with the improved regular expression of this paper. The operation of the user is essentially the operation of the various resources in the cloud environment, so we can abstract each operation to a <resource, operation > tuple. For example, the event that a user created a virtual machine can be represented as <server, create>.

Here is a bug as an example, to illustrate how to extract the dangerous operating's characteristics from bug description. The No.1392527 bug describes a vulnerability that exists in nova VM resize operation. From the bug's description, we can get the following content about the reproduction:

```
Steps to reproduce:

1) Create a new instance, waiting until its status goes to ACTIVE state

2) Call resize API

3) Delete the instance immediately after the task state is "resize_migrated" or vm_state is "resized".

4) Repeat 1 through 3 in a loop
```

From the above description, we can know that when users alternately perform resize and delete operations, it will trigger this bug, which can result in DOS attacks. Thus, the dangerous operation sequence corresponding to the bug is "[<server, create> <server, resize> <server, delete>] *".

C. Security Migration Scheduling Algorithm based on security information

What secure migration scheduling algorithm (SMSA) wants to solve is to identify dangerous users who may attack the open-source cloud platform based on the security information in the community which may attack the open-source cloud platform, and to limit the migration requests for these dangerous users.

SMSA divides the hosts in the cloud into three regions: Normal Region, Dangerous Region, and Dangerous Migration Region. A normal user's virtual machine runs on the host in the normal region, while the host in the "dangerous region" holds the dangerous users' virtual machines. The dangerous user attack the cloud by exploiting the security information in the community to attack the cloud. Certainly, the monitoring level of dangerous region is higher than the normal region. When a user has recently executed a dangerous action and performed a migration, their virtual machines will be migrated to the dangerous migration region. We believe that such user migrates his virtual machines with the dangerous intent, so virtual machines in that region are not allowed to migrate. The task of the SMSA is to select the appropriate destination region where the virtual machine should be migrated. Specific scheduling rules are shown as follows:

SMSA periodically detects the user's behavior via comparing the user's recent operational records with the dangerous behavior repository. For a migration request issued by user in the "Normal Region", if this user has an ab-normal behavior recently, Miper will migrate it to the "Dangerous Region", otherwise it will be migrated to the "Normal Region". For a migration request issued by user in the "Dangerous Region", the virtual machine will be migrated to the "Dangerous Migration Region". For a migration request issued by user in the "Dangerous Migration Region", the migration request will be denied. If an administrator has to migrate because of hardware maintenance, the virtual machine can only be migrated inside its own region. The pseudo-code for scheduling is shown as follows.

Algorithm 1 Security Migration Schedule Algorithm

```
Input: user's information, migration request
Output: destination domain for migration
1:  migration ← TRUE
2:  behavior ← detect_user_recent_behavior(user.id)
3:  user_domain ← get_user_domain(user.id)
4:  if request.migration_flag == FORCE then
5:      dest_domain ← user_domain
6:      goto RETURN_RESPONSE
7:  end if
8:  if behavior == ABNORMAL then
9:      if user_domain == NORMAL_DOMAIN then
10:         dest_domain ← DANGEROUS_DOMAIN
11:     else if user_domain == DANGEROUS_DOMAIN then
12:         dest_domain ← DANGEROUS_MIGRATE_\
                          DOMAIN
13:     else
14:         user_domain ← DANGEROUS_MIGRATE_\
                          DOMAIN
15:     end if
16: else
17:     if user_domain == DANGEROUS_MIGRATE_\
                          DOMAIN then
18:         migration ← FALSE
19:         dest_domain ← user_domain
20:     end if
21: end if
```

```
22: RETURN_RESPONSE:
23:     response.migration $\leftarrow$ migration
24:     response.dest_domain $\leftarrow$ dest_domain
```

In order to obtain the user's migration request, we added two filters in Nova and Cinder. When the filter identifies the request as a migration request, this request will be forwarded to the Miper API. Then Miper API hands over the request to Miper Engine through RPC. When Miper Engine receives the migration request from the Miper API, it assigns the appropriate migration region according to SMSA.

V. Evaluation

In this section, Miper is evaluated in three aspects. At first, we analyze the results of Miper crawler crawling and find out the rules hidden in the results. Next, we compare the damage to the cloud environment caused by dangerous users to the cloud environment with and without Miper protection and evaluate the security effect of secure migration scheduling algorithm. Finally, we evaluated the performance of Miper. Experiments show that Miper can satisfy the security objectives with the acceptable overload.

The experimental environment consists of 7 virtual machines on the VMware work-station 12, where we built an OpenStack cluster of Mitaka version including 1 controller node, 3 compute nodes, 2 block nodes, and an NFS server. To achieve live migration, virtual machine-related files on all compute nodes are stored in the NFS server's shared directory /var/lib/nova/instances/. Fig 5 shows our experimental environment.

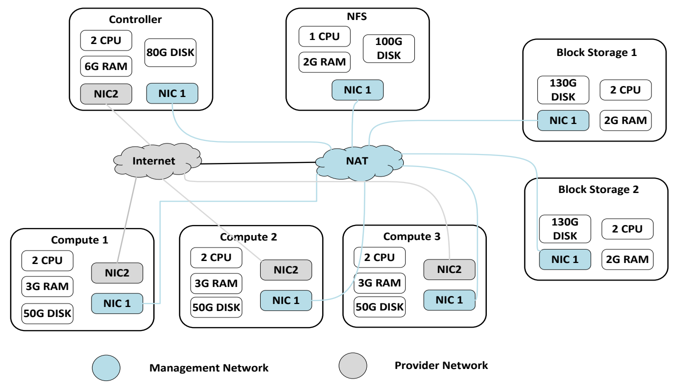


Fig. 5. Experiment environment diagram

A. *Security Information Repository Evaluation*

1) *Methods Effectiveness Evaluation.*

According to the information collection rules in this paper, we crawled all the security information from the inception of OpenStack community to August 2016, and crawled 578 security-related bugs. We analyze the relevant data in the process of constructing the SIR from the non-repetition rate, Review URL accuracy rate and illegal operation sequence feature extraction rate of the three indicators according to the relevant data in the process of constructing the SIR.

Non - repetition Rate. The non-repetition rate is the ratio of the number of security-related bugs that are ultimately stored to the sum of the security bugs extracted from all data sources. The number of bugs in the Security bugs repository is 578, while the number in OSSA, OSSN, Mailing List and Record are 139, 58, 335 and 149 respectively. The number of bugs whose tag is security is 36, and the sum of them is 717. So the non-repetition rate is 578/717 = 80.6%.

TABLE III. Result of Extration from OpenStack Community

Source	OSSA	OSSN	Mailing List	IRC Record	bug repository
Num	139	58	335	149	36

TABLE IV. Result of review URL extration

Condition	Accuracy URL	Accuracy rate
Feature 1	421	86.80%
Feature 1 and 2	461	95.10%

Review URL Accuracy Rate. We extract the official review URL through two features, and then test the accuracy in one feature and two feature extraction cases respectively. Experimental results show that among the 578 security-related bugs extracted, 93 bugs are still in the discussion state, and we cannot extract the official review URL from them. So only 485 bugs are extracted. In the use of only one feature of the conditions, the correct number is 421, and the accuracy rate is 86.8%. When increasing the second condition, the correct extraction number of the review URL number rose to 461, with the accuracy rate reaching 95.1%. These remaining 24 bugs cannot extract the correct review URL, because these bugs' dates are too early and the community has not formed standard review process at that moment.

Dangerous operation sequence feature extraction rate. In the process of extracting illegal operation sequence features from the bug description information, this paper records the number of bugs without operating characteristics, the number of bugs in a single operating feature, and the number of bugs in the operating sequence characteristics. The results show that the number of bugs without operation features is the highest, accounting for 47.1%, and the single operation feature is the second, accounting for 33.1%, and the operation sequence is 19.8%.

2) *Classification of security-related bugs.*

In order to better apply the security information in the security information repository, we classify the security related bugs in the security information repository. Security-related bugs can be categorized from multiple dimensions. We divide them into the following five types according to the reason for vulnerability: design error, configuration error, input error, logic error, memory damage [32].

The overall classification is shown in Fig 6. We can see that design errors occur most often in OpenStack. As a fast-growing open-source project, OpenStack is de-signed to mainly consider availability and reliability at the outset, with insufficient security considerations. The second most problem is the Configuration Error. In the OpenStack platform, each component has multiple configuration files with dazzling configuration options.

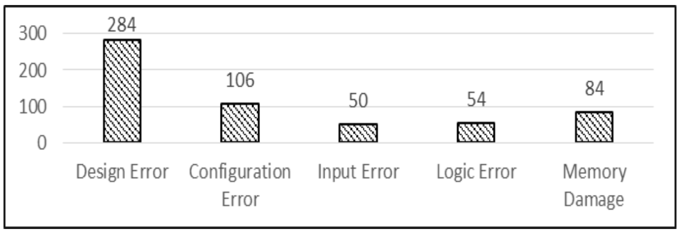


Fig. 6. The categories of the vulnerability's formation case

B. Security of Migration Scheduling Algorithm

In this section, we use a concrete example to prove that Miper can effectively ensure the security of virtual machine migration process in the cloud, and cannot be abused by dangerous users. The OSSA in the OpenStack community numbered 2015-017 report vulnerabilities that exist in virtual machine resize process. If an authenticated user deletes an instance when it is in resize state, the original instance will not be deleted from the compute node it was running on. An attacker can use this to launch a denial of service attack. We extracted the features of operation sequence that could trigger this security-related bug and stored it into the feature library in advance.

In the experimental cloud environment, we wrote an attack script that uses novaclient to repeat create, resize and delete operations in 5 minutes. Initially, we did not start the Miper service. When the script run over, we find that it creates a total of 15 virtual machines and executed the resize and delete operations sequence. There are disk files both in source host and destination host for each virtual machine, which means that the process produces 15 redundant virtual machine disk file.

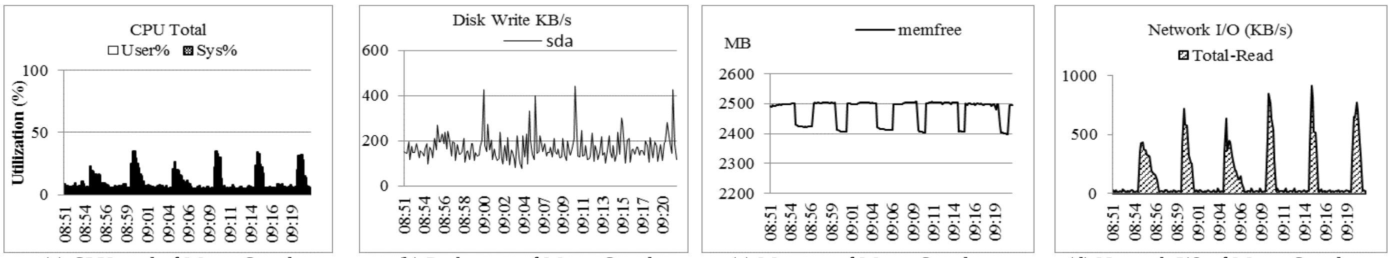
Before starting the Miper service to evaluate its security, we used the Availability Zone in OpenStack to divide Compute nodes into three isolation regions. Then we boot the Miper service and execute the script. As a result, we found that the script still creates 15 virtual machines, but only 2 redundant disk files are found on the migrated source and destination hosts. And the user's virtual machines all exist in the dangerous migration region, leading to that he user cannot perform resize and other migration-related operations any more.

C. Performance

We further analyzed the performance of the Miper in terms of its consumption of physical resources and its impacts on the cloud user's experience. We take into ac-count two performance issues: the overhead of the Miper Crawler task that runs periodically, and the overhead of the Miper component on the system.

1) Miper Crawler Performance

The Miper Crawler service regularly crawls security information from the Open-Stack community and is the most resource-intensive service in Miper. In order to evaluate the performance of the Miper Crawler service, we set up Miper Crawler service crawling community information every 5 minutes, and monitor the status of system in half an hour, mainly including CPU utilization, memory usage, disk IO and network throughput to reflect the Miper Crawler's overload on the system. Fig 7 shows our experimental results.



(a) CPU total of Miper Crawler (b) Disk write of Miper Crawler (c) Memory of Miper Crawler (d) Network I/O of Miper Crawler

Fig. 7. Overhead of Miper Crawler task

Experiments show that Miper Crawler has high occupancy rate on CPU, disk and network at run time. The high consumption on Disk and network is easily understood, because Miper Crawler needs to access the OpenStack community and store the crawled data into local disk. High CPU usage is due to multi-threading technology, which is used to improve the efficiency of information crawling, and data preprocessing also consumes CPU. Taking into account the actual environment, Miper Crawler is generally set to run once a day or even once a week, and the running time is generally at night, when the resource utilization is low. So the overhead of Miper Crawler is completely acceptable.

2) Miper Performance

We evaluated the overall performance of the Miper with the Miper Crawler in a silent state in this part. The experimental results are shown in Table V and Table VI. Like other components of OpenStack, Miper is a stateless application. Each service in Miper is able to start multiple processes at the same time to provide scalable processing power. In the case of each service process of 10, we can see that Miper overall CPU consumption is 10.1%, and the memory consumption is 10.9%.

TABLE V. MEMORY PERCENTAGE OF MIPER

Process Num	**Miper API**	**Miper Analyzer**	**Miper Crawler**	**Miper Engine**	**Total**
1	0.5	0.2	0.1	0.4	1.2
2	0.9	0.4	0.2	0.8	2.3
5	2.1	1.0	0.5	2.0	5.6
10	4.1	2.0	1.0	3.8	10.9

TABLE VI. CPU PERCENTAGE OF MIPER

Process Num	Miper API	Miper Analyzer	Miper Crawler	Miper Engine	Total
1	2.2	2.0	2.3	2.4	8.9
2	2.3	1.7	2.7	2.7	9.4
5	2.6	2.3	2.8	3.3	11.0
10	2.7	2.3	2.0	4.3	11.3

VI. RELATED WORK

There is little literature on OpenStack security information analysis. Torkura et al. [33] applied quantitative security assessment approach to cloud computing using OpenStack as a case study. They derived empirical data from NVD, OSVDB and OSSA. Torkura et al. [34] implemented CAVAS, a prototype that provides first steps to integration of VAaaS in OpenStack. We focus on identifying vulnerabilities that are specific to OpenStack core services and software and not those affecting other third party software. We leverage on information provided from public vulnerability information resources such as NVD and OSVDB. Tong et al. [35] we analyze and describe features of bug-induced failure logs from bug repository and Q&A websites, and then propose a general automatic approach to pinpoint logs of bug-induced failure from log files of open cloud platform.

VII. CONCLUSION AND FUTURE WORK

As far as we know, we are the first to consider of comprehensive analysis and aggregation of security information in the OpenStack community. Firstly, the original security data in each security activity is extracted, and then the security information in the activity is analyzed by the method of NLP to establish the OpenStack security information repository. Finally, a secure migration scheduling algorithm is proposed based on the dangerous operation sequence signature database in the security information repository, which prevents the dangerous users from using virtual machine migration to expand the scope of the damage. Experiments show that the security information repository has a positive effect on improving the utilization of security information in the OpenStack community.

In the future, we plan to apply more NLP techniques in the extraction of security information to improve extraction efficiency and automation. And we could combine it with the big data platform to improve data processing efficiency. There are also some works to do to improve the robustness of behavior recognition process. In this scenario, an attacker knowing about the recognition engine just needs to adapt her actions slightly can bypass behavior recognition.

ACKNOWLEDGMENT *(Heading 5)*

REFERENCES

[1] Almorsy M, Grundy J, Müller I. An analysis of the cloud computing security problem[J]. arXiv preprint arXiv:1609.01107, 2016.

[2] Modi C, Patel D, Borisaniya B, et al. A survey on security issues and solutions at different layers of Cloud computing[J]. The Journal of Supercomputing, 2013, 63(2): 561-592.

[3] Ahmed M, Hossain M A. Cloud computing and security issues in the cloud[J]. International Journal of Network Security & Its Applications, 2014, 6(1): 25.

[4] Gonzalez N, Miers C, Redigolo F, et al. A quantitative analysis of current security concerns and solutions for cloud computing[J]. Journal of Cloud Computing: Advances, Systems and Applications, 2012, 1(1): 11.

[5] Nelson G, Charles M, Fernando R, et al. MakanP., A quantitative analysis of current security concerns and solutions for cloud computing[J]. Journal of cloud computing: advanced, systems and applications, springer, 2014, 1(11).

[6] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: toward an open-source solution for cloud computing[J]. International Journal of Computer Applications, 2012, 55(3).

[7] Rosado T, Bernardino J. An overview of openstack architecture[C]//Proceedings of the 18th International Database Engineering & Applications Symposium. ACM, 2014: 366-367.

[8] Denton J. Learning OpenStack Networking (Neutron)[M]. Packt Publishing Ltd, 2014.

[9] Tang B, Sandhu R. Extending openstack access control with domain trust[C]//International Conference on Network and System Security. Springer International Publishing, 2014: 54-69.

[10] Bhatt S, Patwa F, Sandhu R. An Attribute-Based Access Control Extension for OpenStack and its Enforcement Utilizing the Policy Machine[C]//Collaboration and Internet Computing (CIC), 2016 IEEE 2nd International Conference on. IEEE, 2016: 37-45.

[11] Luo Y, Luo W, Puyang T, et al. OpenStack Security Modules: A Least-Invasive Access Control Framework for the Cloud[C]//Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on. IEEE, 2016: 51-58.

[12] Nguyen D, Park J, Sandhu R. Adopting provenance-based access control in OpenStack cloud IaaS[C]//International Conference on Network and System Security. Springer International Publishing, 2014: 15-27.

[13] Majumdar S, Jarraya Y, Madi T, et al. Proactive Verification of Security Compliance for Clouds Through Pre-computation: Application to OpenStack[C]//European Symposium on Research in Computer Security. Springer International Publishing, 2016: 47-66.

[14] Madi T, Majumdar S, Wang Y, et al. Auditing security compliance of the virtualized infrastructure in the cloud: application to OpenStack[C]//Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. ACM, 2016: 195-206.

[15] Majumdar S, Madi T, Wang Y, et al. Security compliance auditing of identity and access management in the cloud: application to OpenStack[C]//Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on. IEEE, 2015: 58-65.

[16] Ullah K W, Ahmed A S, Ylitalo J. Towards building an automated security compliance tool for the cloud[C]//Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on. IEEE, 2013: 1587-1593.

[17] Torkura K A, Meinel C. Towards Vulnerability Assessment as a Service in OpenStack Clouds[C]//Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on. IEEE, 2016: 1-8.

[18] Massonet P, Dupont S, Michot A, et al. Enforcement of global security policies in federated cloud networks with virtual network functions[C]//Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on. IEEE, 2016: 81-84.

[19] Patel P, Tiwari V, Abhishek M K. SDN and NFV integration in openstack cloud to improve network services and security[C]//Advanced Communication Control and Computing Technologies (ICACCCT), 2016 International Conference on. IEEE, 2016: 655-660.

[20] Battula L R. Network security function virtualization (NSFV) towards cloud computing with NFV over Openflow infrastructure: Challenges and novel approaches[C]//Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on. IEEE, 2014: 1622-1628.

[21] Zhang T, Lee R B. Monitoring and Attestation of Virtual Machine Security Health in Cloud Computing[J]. IEEE Micro, 2016, 36(5): 28-37.

[22] Corradi A, Fanelli M, Foschini L. VM consolidation: A real case based on OpenStack Cloud[J]. Future Generation Computer Systems, 2014, 32: 118-127.

[23] Tong J, Ying L, Xiaoyong Y, et al. Characterizing and Predicting Bug Assignment in OpenStack[C]//Trustworthy Systems and Their Applications (TSA), 2015 Second International Conference on. IEEE, 2015: 16-23.

[24] Robles G, González-Barahona J M, Cervigón C, et al. Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014: 222-231.

[25] Kamboj R, Arya A. OpenStack: open source cloud computing IaaS platform[J]. International Journal of Advanced Research in Computer Science and Software Engineering, 2014, 4(5).

[26] https://www.openstack.org/zh_CN/marketplace/public-clouds/

[27] OpenStack, Openstack security, Online, Openstack Security. [Online]. Available: https://wiki.openstack.org/wiki/Security.

[28] Patel A, Taghavi M, Bakhtiyari K, et al. An intrusion detection and prevention system in cloud computing: A systematic review[J]. Journal of network and computer applica-tions, 2013, 36(1): 25-41.

[29] Liao H J, Lin C H R, Lin Y C, et al. Intrusion detection system: A comprehensive re-view[J]. Journal of Network and Computer Applications, 2013, 36(1): 16-24.

[30] Jyothsna V, Prasad V V R, Prasad K M. A review of anomaly based intrusion detection systems[J]. International Journal of Computer Applications, 2011, 28(7): 26-35.

[31] Vieira K, Schulter A, Westphall C, et al. Intrusion detection for grid and cloud compu-ting[J]. IT Professional Magazine, 2010, 12(4): 38.

[32] NSFOCUS.http://blog.nsfocus.net/.

[33] Torkura K A, Cheng F, Meinel C. Application of quantitative security metrics in cloud computing[C]//Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for. IEEE, 2015: 256-262.

[34] Torkura K A, Meinel C. Towards Vulnerability Assessment as a Service in OpenStack Clouds[C]//Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on. IEEE, 2016: 1-8.

[35] Tong J, Ying L, Hongyan T, et al. An Approach to Pinpointing Bug-Induced Failure in Logs of Open Cloud Platforms[C]//Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on. IEEE, 2016: 294-302.