ENHANCING SECURITY IN CLOUD COMPUTING THROUGH VIRTUAL MACHINE PLACEMENT

by

JIN HAN, M.Sc.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

COMMITTEE MEMBERS:
Ravi Sandhu, Ph.D., Co-Chair
Meng Yu, Ph.D. , Co-Chair
Jianwei Niu, Ph.D.
Jianhua Ruan, Ph.D.
Matthew Gibson, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
December 2019

DEDICATION

This proposal is dedicated to my beautiful wife Dada and lovely son Samuel, my parents and my other families in China for their ever-lasting love and support.

ACKNOWLEDGEMENTS

It has been a wonderful journey with a great learning experience to achieve this point of my doctoral program, on many different levels. So many people have had a great impact and deserved to be sincerely thanked.

First of all, I want to express my sincere gratitude to my beautiful wife Qianqian. I could not imagine the finish of my program without her fully support and sacrifice.

Foremost, I want to express my sincere gratitude toward my advisors, Dr. Ravi Sandhu and Dr. Meng Yu, for the continuous support of my Ph.D. study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study. I am always in debt to their valuable teaching. I appreciate the Department of Computer Science at the University of Texas at San Antonio for their generous financial support during my study.

I am sincerely and heartily grateful to Dr. Jianwei Niu, Dr. Jianhua Ruan and Dr. Matthew Gibson for taking the time to be my committee members. I am honored to have you all. Also, my sincere thanks go to Dr. Wanyu Zang at Texas A&M University at San Antonio, Dr. Songqing Chen at George Mason University for their writing assistance and proofreading.

I would also like to thank my UTSA friends for their support and for making my life more wonderful: Hongyu Liu, Jingpeng Zhou, Zhen Gao, Naiwei Liu, and Sam Silvestro. I am so honored that we could study together and I will always remember every pleasant and tough moment we have been through together.

Last but not least, I would like to show gratitude to my parents, my father Jianmao Han and my mother Yiping Wen, for all their support, love and encouragement in my whole life.

Finally, thanks to ARO grant W911NF-15-1-026 and NSF CNS-1634441 for supporting this research and highly appreciate the support provided by Samsung SARC Research Center at Austin.

This Masters Thesis/Recital Document or Doctoral Dissertation was produced in accordance with guidelines which permit the inclusion as part of the Masters Thesis/Recital Document or Doc-

toral Dissertation the text of an original paper, or papers, submitted for publication. The Masters Thesis/Recital Document or Doctoral Dissertation must still conform to all other requirements explained in the Guide for the Preparation of a Masters Thesis/Recital Document or Doctoral Dissertation at The University of Texas at San Antonio. It must include a comprehensive abstract, a full introduction and literature review, and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this Masters Thesis/Recital Document or Doctoral Dissertation to include as chapters authentic copies of papers already published, provided these meet type size, margin, and legibility requirements. In such cases, connecting texts, which provide logical bridges between different manuscripts, are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the students contribution to the work and acknowledging the contribution of the other author(s). The signatures of the Supervising Committee which precede all other material in the Masters Thesis/Recital Document or Doctoral Dissertation attest to the accuracy of this statement.

December 2019

ENHANCING SECURITY IN CLOUD COMPUTING THROUGH VIRTUAL MACHINE PLACEMENT

Jin Han, Ph.D.
The University of Texas at San Antonio, 2019

Supervising Professors: Ravi Sandhu, Ph.D. and Meng Yu, Ph.D.

Cloud computing, while becoming more and more popular as a dominant computing platform, introduces new security challenges. Considering the benefits provided by the infrastructure as a Service(IaaS) model, such as reduced cost for customer and better resource utilization for service provider, it attracted more attention from attacker. In recent years, the attacks are specifically designed to co-locate with target virtual machines in the cloud. Thus, when virtual machines are deployed in a cloud environment, virtual machine placement strategies can significantly affect the overall security risks of the entire cloud. A virtual machine placement without considering security risks may put the customers, or even the entire cloud, in danger.

This dissertation addresses the problem of risk evaluation of a cloud and put the co-residency risk into consideration. The current state-of-art VM allocation policies were reviewed and existing multi-objective optimization algorithms were studied. We believe that existing solutions of Virtual Machine Placement(VMP) lack consideration of security issues caused by co-residency. Proper security metrics should be proposed to guide the new solution of VMP. To address the issue, We proposed a Secured Multi-Objective Optimization based virtual machine Placement algorithm (SMOOP) to seek an improved solution to reduce overall security risks of a cloud. The goal is accomplished in three steps. First, we proposed a set of security metrics to quantify the security risks of a cloud, using more comprehensive security evaluation based on network, host and VM aspects. Second, we designed and implemented SMOOP that utilizes our security metrics to improve the security level of the cloud, while also considering workload balance, resource utilization on CPU, memory, disk, and network traffic. The optimization preference be adjusted by users' own configuration. Third, we extended our work by studying state-of-art VM allocation policy,

and integrate them into our system design to defeat co-residency attacks. Our evaluation results show that the security level of clouds can be effectively improved through our proposed algorithm with affordable overhead, while other objects were also satisfied at the same time.

To further enhance the adaptability of our proposed security metrics, we focused on developing a fine-grained model to better quantify the risk level caused by co-residency. Based on a large scale dataset collected from Microsoft Azure Platform, we profiled the behavior pattern of normal service subscriber based on our proposed feature metrics. Service subscribers are clustered into multiple categories. After the baseline is built up based on the normal behavior pattern, the derivation can be evaluated for each category and the high-risk group can be labeled accordingly. With the labeled data sets, a classification component and a quantification component are constructed are used to dynamically quantify the co-residency risk level for a specific VM. Based on our experimental result, our model demonstrated robustness to new data and the accuracy was verified by examination of F Measuring Matrix.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Cloud computing is the basis of many services in our daily life, such as email services, Internet of Things (IoT) and file sharing services, etc. In an Infrastructure as a Service (IaaS) cloud like Amazon EC2 [8], many virtual machines (VMs) share a physical server. The placement of virtual machines can have different strategies, leading to different computing performance, energy consumption, and resource utilization. Therefore, given different resource constraints, how to achieve multiple objectives is a very important problem in cloud computing. Such a problem has attracted extensive attention recently [24, 70, 80].

With resource and other constraints, the virtual machine placement (VMP) problem is essentially a multiple-objective optimization problem. Phan et al. [80] used an Evolutionary Multi-Objective Optimization (EMOA) algorithm to build Green Clouds when considering energy consumption, cooling energy consumption and user-to-service distance in VMP optimization. Xu and Fortes [108] proposed a genetic algorithm with fuzzy multi-objective evaluation to minimize the total resource wastage, power consumption and thermal dissipation costs in VMs placement. Shigeta et al. [91] suggested to assign different weights to multi-objectives on cost and performance and built a cost evaluation plug-in module to search for the optimal VMs placement. Ian Kash et al. [65] proposed a demand vectors from a sparse demand matrix to better manage resource allocation in the cloud systems. Some other research focus on minimizing the overall network cost while considering large communication requirements [7, 74], or applying the constraint programming (CP) engine to optimize VMP [6, 30]. While above multi-objective optimization placement schemes greatly improve the overall performance of the cloud, the security risk of the entire cloud environment was not considered as an objective or at most considered as one constraint in the initialization phase.

At the same time, there are new types of attacks targeting at the cloud infrastructure. For example, some attacks, such as those discussed in [33, 34, 50, 86] , exploit the vulnerabilities of hypervisor (or Virtual Machine Monitor, VMM), e.g., Xen [15] or KVM [38]. Once the attacker

compromises the hypervisor, the attacker can take over all the VMs running on it. In [84] (HYG attack), the initial stage of the attack is to locate a target VM. Upon success, the attacker will try to launch a VM on the same physical server. It is a placement based attack and the success of the attack depends on the placement strategies of the cloud, or the configuration policy of the cloud. Apparently, collocating with vulnerable virtual machines, or "bad neighbors", on the same physical server does increase the security risks to cloud users. Thus, the security risk exposed to the user depends not only on how secure the VM itself is, such as the operating system and applications running inside, but also the Virtual Machine Monitor (VMM or Hypervisor), running underlying the VMs, and other VMs coresident on the same node.

1.1 Motivation

We believe that security should be considered as one key element, the same as energy and performance, in VM placement. In the previous work [70], we proposed a VMP scheme based on the security risk of each VM. However, the security analysis of our previous work mainly focused on dependency relations. Yuchi and Shetty [113] extended our previous work to the VM placement initialization. Yu et al. [112] proposed isolation rules to formulate the VMP behavior based on the Chinese wall policy. Unfortunately, this work mainly focuses on improving security and overlooks other objectives, such as energy saving and resource utilization. Besides, the security measurements in this work mainly consider the vulnerabilities of VMs or hypervisor, or security regulations, without considering the security assessment of a VMP.

When comparing different VMP schemes, the security metrics can only be evaluated after a placement is specified. For example, a specific placement scheme has an unique attack path exposed by co-residence that may disappear in a different placement. Therefore, there is no generic function to map a placement scheme into a security assessment value. We cannot simply apply any existing evolutionary multi-objective optimization algorithm (EMOA) to solve our problem directly. Furthermore, the low efficiency and the complicated security assessment require us to design our own crossover and mutation procedures in the the EMOA algorithm.

In the past a few years, co-residence attacks have attracted a lot of attention. The straightforward solution to such attacks is to eliminate the side channel directly [12, 89, 107]. However, this solution requires the modification of the hardware or the cloud platforms, which is not practical. Most recent work [14, 55, 56] tried to use the VM allocation policy to make the co-residence infeasible to achieve. Motivated by their designs, we also aim to integrate these objectives into our design in order to mitigate co-resident attacks.

1.2 Problem and Thesis Statement

Virtual Machine Placement acts as a key role in the security field of Cloud Computing. Several risks should be evaluated by a comprehensive set of security metrics. For each security aspect, the method to quantify it should be well studied and the accuracy level should be high. At the same time, for practical usage, multiple objectives should also be met, such as work load balance, resource utilization, and network traffic.

1.2.1 Scope and Assumptions

In this dissertation, in addition to the security risks caused by VM and Hypervisor themselves' vulnerabilities, we also considered the risks caused by network communication and co-residency based attacks, such as cross-VM side channel attacks. Meanwhile, we assume that the attackers are capable of utilizing vulnerabilities in both VMs and virtual machine monitors (VMMs, or hypervisor) of the clouds.

We have the following assumptions for the cloud: ① the cloud service management, placement related software components, and the migration process are all trusted; ② for simplicity, migration of a VM will result in affordable cost in terms of service interruption and consume the same amount of resources; ③ the cloud service provider has enough CPU, network bandwidth, and other resources to perform arbitrary migration of VMs; and ④ the cloud service provider has sufficient resources as the reward, e.g., extra memory or CPUs, to motivate VM migration. The above assumptions ensure that change of VM placement is both acceptable and affordable for cloud

service provider and service subscriber.

1.2.2 Terminology

In this dissertation, we used multiple terms and terminologies, which could be review in Table 1.1

Table 1.1: List of Abbreviations

Abbreviation hline Abbreviation	Full Term Full Term
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CSP	Cloud Service Provider
DBSCAN	Density-based Spatial Clustering of Applications with Noise
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
IaaS	Infrastructure as a Service
KNN	K-nearest Neighbors
ML	Machine Learning
NIST	National Institute of Standards and Technology
SS	Service Subscriber
VM	Virtual Machine
PM	Physical Machine

1.3 Key Contributions

In this dissertation, we proposed a new security measurement for the cloud, and a new VMP approach to provide better intrusion resilience, workload balance, resource utilization, and network performance. In the proposed security assessment, we consider the security risks not only on VMs and the hypervisor, but also from the co-resident VMs and the network connections that will change with the VM placement. Based on the proposed security measurement, we design an evolutionary multi-objectives optimization algorithm, named as Secured Multi-Objective Optimization based virtual machine Placement algorithm (SMOOP), to seek an improved solutions on specific target objective or balancing on multiple objectives on security, resource utilization, network traffic and workload.

Our proposed scheme features an innovative combination of the following contributions.

- We conduct security assessment of the cloud from four aspects: networking, co-residence, hypervisor vulnerabilities, and VM vulnerabilities. The proposed security risk assessment is placement specific and crosses multiple dimensions. We provide detailed metrics and an approach to measure the security of the cloud in the case study and experiments.

- We consider security as one objective in VMP strategies, with other objectives and constraints at the same time. To the best of our knowledge, this is the first work that includes a placement specific security assessment in the context of multi-objective optimization based VMP.

- We propose a highly scalable scheme, SMOOP with five placement strategies, to seek the improved placement to balance multiple objectives. Users could adjust the weight list according to their own preferences. The experimental results confirm the effectiveness of our strategies and SMOOP can provide an overall improved security of the cloud with reasonable overhead.

- We study the latest VM allocation policies and integrate them into our design to defeat co-residence attacks. Thus, our algorithm is compatible with up-to-date VM allocation policies. Besides, our placement strategies can be configured by the users to fit for different business needs.

- Based on the large scale Microsoft Azure dataset, we created a profile for the behavior patterns of normal service subscriber within our proposed feature metrics. Service Subscribers were clustered into several categories. Inactive (Normal), Period Active, and Extreme Active labels were assigned accordingly. With the labeled data sets, a classification component, and a quantification component were constructed and used to dynamically quantify the co-residency risk level for a specific VM. Based on our experimental result, our model demonstrated robustness to new data sets and the accuracy had been verified by examination of F Measuring Matrix.

1.4 Related Publication

Security Metrics to valuation of a cloud in Chapter 3 and Security-aware Multi-Objective Optimization based virtual machine Placement algorithm (SMOOP) proposed in Chapter 4 were accept for publication in The 31st Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec 2017).

- Jin Han, Wanyu Zang, Songqing Chen, and Meng Yu. Reducing security risks of clouds through virtual machine placement. In Giovanni Livraga and Sencun Zhu, editors, Data and Applications Security and Privacy (Dbsec 2017), pages 275-292, Cham, 2017. Springer International Publishing.

The current state-of-art VM allocation policies were integrated into SMOOP in Chapter 5 were accepted for publication in Journal of Computer Security 2018.

- Jin Han, Wanyu Zang, Li Liu, Songqing Chen, and Meng Yu. Risk-aware multi-objectives optimized virtual machine placement in the cloud. Journal of Computer Security, Vol 26, Issue 5, pages 1-24 2018. IOS Press Publishing.

1.5 Organization of the Dissertation

In this chapter, the motivation behind this work, the problem statement, the objectives, and key contributions are explained. The rest of the dissertation is organized as follows. In Chapter 2, a brief background on the vulnerability of VM was discussed. We also talked about the current state-of-art related work on Virtual Machine Placement. Clustering Algorithm and Machine Learning were also introduced for proper addressing our problem. In Chapter 3, we proposed a practical security metric to quantify a comprehensive security risk evaluation of cloud environments from co-residency, network, host and VM aspects. Chapter 4 describes the formulation of VMP optimization problem and objectives we want to accomplish. Secured Multi-Objective Optimization based virtual machine Placement algorithm (SMOOP) was proposed and evaluated. Chapter 5 described how to integrate the state-of-art VM allocation policies into SMOOP. In Chapter 6, based

on a large scale Microsoft Azure dataset, Service Subscribers were clustered into several categories and Inactive(Normal), Period Active, Extreme Active labels were assigned accordingly. A fine-tuned deep neural network was used to quantify the co-residency risk rate and its robustness was verified. Transfer training for new data was tested and the accuracy rate was verified by examination of F Measuring Matrix. The last Chapter 7 is the conclusion of our work and the discussion of our future work.

CHAPTER 2: BACKGROUND AND RELATED WORKS

2.1 Vulnerability

A vulnerability is a weakness in a product that could allow an attacker to compromise the integrity, availability, or confidentiality of that product. [9, 48]

In computer security, it can be defined as a weakening or the removal of certain resistance strength of the system by a malicious code. Is the probability that an asset will be unable to resist the actions of a threat agent. It can also be defined as the inability of a system or a unit to withstand the effects of a threat agent. Some Common types of vulnerabilities are listed as the following [93]:

- Remote code execution: Allows an attacker to run arbitrary, system level code on a vulnerable server and retrieve any desired information contained therein. Improper coding errors lead to this vulnerability.

- SQL injection: This technique allows an attacker to retrieve crucial information from a Web server's database. The impact of this attack can vary from basic information disclosure to remote code execution and total system compromise

- Format string vulnerabilities: This vulnerability results from the use of unfiltered user input as the format string parameter in certain Perl or C functions that perform formatting, such as C's printf(). This vulnerability attack falls into three general categories: denial of service, reading and writing.

- Cross Site Scripting (XSS): The success of this attack requires the victim to execute a malicious URL which may be crafted in such a manner to appear to be legitimate at first look

- Username enumeration: Username enumeration is a type of attack where the backend validation script tells the attacker if the supplied username is correct or not

What is a Vulnerability scan? A vulnerability scan is a script-tool with various kinds of plugins that can detect different kinds of vulnerabilities automatically. Each plugin has the potential of detecting a particular kind of vulnerability. [3]

When performing a vulnerability scan, choosing a scanner with more knowledge base gives the security analyst an edge and wider scope. This is because a vulnerability scanner with more knowledge base will perform better and can detect more vulnerabilities in a network scan than one with limited knowledge base.

2.2 Security Risks in Cloud Computing

Cloud Computing is a type of Internet based computing that provides shared computer processing resources and data to computers and other devices on demand. It also is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., computer networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort [1]. There are two major type of Cloud Computing: 1. Public Clouds provide service to any paying customer, 2. Pravite Clouds are accessible only to authorized users.

In literature, many definitions of cloud computing could be found [43, 99], but in this dissertation, Cloud Computing was defined as below:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models" [78]

All cloud computing are built upon some same conceptual framework of remote infrastructure powered by physical servers. On-demand service is provided to service subscriber in below categories:

- HaaS: Hardware as a Service. Service subscriber get access to bare-bones hardware ma-

chines, do whatever they want with them.

- IaaS: Infrastructure as a Service. Service subscriber get access to flexible computing and storage infrastructure. Virtualization is one way of achieving this and it is often said to subsume HaaS.

- PaaS: Platform as a Service. Service subscriber get access to flexible computing and storage infrastructure, coupled with a software platform (often tightly coupled).

- SaaS: Software as a Service. Service subscriber get access to software services, when they need this service. It is Often said to subsume SOA (Service Oriented Architectures).

Nowadays, Customers are attracted by Cloud Computing services by its essential characteristics, such as on-demand service, pay-as-you-go pricing, easily controlled resource utilization. With the increased popularity of Cloud Computing, it also attracted more attackers at the same time. To better understand the security factor of Cloud Computing, threats and vulnerabilities in cloud need to be well studied. Due to current cloud properties, vulnerabilities in cloud computing were discussed. [36, 46, 48, 49, 62, 98]

2.2.1 Co-residency Attack

Since multiple Virtual Machines could be deployed into a same physical machine, a new threat is introduced as co-residency risk, such as cross-VM attack [84] which allows for shared memory attack or side-channel attack. There is another threat generated by this risk type is VM image manipulation [47], where the victim's VM image could be used by attack to create some new VMs for attacking.

Logical isolation between VMs deployed into the same server was provided by Virtualization techniques [16]. Under the control of hypervisor, one VM could only access to its own assigned partition of hardware resource, such as computing power or data storage. One VM should never be affected or compromised by another VM's behavior. However, in real world, many stealthy attack could happen. For example, cache utilization rate would determine the execution time of

cache read operations [85]. As a result, various types of side channels between attacker's VMs and victim's VMs could be built and sensitive information could be extracted [59, 85, 111, 116], which also was called as co-resident attack.

Resource sharing in cloud computing raises a threat of Cache-Based Side Channel Attack (CSCA). It is proposed to detect and prevent guest virtual machines from CSCA. Cache miss patterns were analyzed in this solution to detect side channel attack. CSCA is divided into two types and those are time-driven cache attacks and trace-driven cache attacks. It is based on a cloud setting with two VMs installed on the same physical machine using a bare-metal hypervisor sharing a highest-level cache. One of the most influential one is to extract private keys [117]. Author successfully constructed such a side-channel requires overcoming challenges including core migration, numerous sources of channel noise, and the difficulty of preempting the victim with sufficient frequency to extract fine-grained information from it, and demonstrates the attack in a lab setting by extracting an ElGamal decryption key from a victim using the most recent version of the libgcrypt cryptographic library. Since the Last Lever Cache was used as the side channel for attack, Yinqian Zhang et al. [57] describer a Home Alone system that makes tenant verification of exclusive utilization of VMs over physical machine. The primary concept behind Home Alone is to reverse the essential application of side channels. Indeed of making the best advantage of side channel to be vector of any attack, Home Alone utilizes a side-channel (in Last Level Cache) as a new protective detection tool. By examining the utilization of cache while the period in which friendly VMs collaborate to eliminate segments of cache, the tenant with home Alone can identify co-resident activity of foe VM.

To defend this particular type attack, lots of other research effort was devoted into different directions. One of them is to enforce the isolation of virtual resources. Khalid Bijon et al. [23] proposed a formal Trusted Virtual Datacenter (TVD) management model to manage strong isolation among virtual resources and also develop an authorization model for the cloud administrative-user privilege management in the system. In their following works [21, 22], they presented attribute-based constraints specification and enforcement as a mechanism to mitigate such co-resident risks.

Conflicting attribute values are specified by the tenant or by the cloud IaaS system as appropriate. Based on conflict specifications, author developed a conflict-free virtual machine scheduling framework. Our proposed framework is also inspired by their work and focused on the scheduling strategy of deployment of Virtual Machines.

Some of existing works focused on the analysis of the potential data traffic information. Adam Bates et al. [82] depicts about Co-resident watermarking, traffic analysis attack, which facilitates dangerous co-resident VM to introduce watermark signature over the internet flow of target instance. The final examination illustrates that there is a careful hardware design to be utilized in the cloud environment. Similarly, there has been many techniques developed for the security of cloud computing.

Another interesting direction is based on game theory. Yi Han et al. [56] proposed a mechanism to identify the co-resident attacker's behaviors in the absence of any defense. Overall speaking, the attacker has to start a much larger number of VMs than the normal user to achieve the co-residency with their target. In their subsequent work [54], they proposed a semi-supervised learning based defense strategy to increase the attack cost. Followed with his lead, multiple works were done [57, 83, 106] based on Game Theory Model. C. zhang et al. [114] proposed an information Model to increase user's value and improve user experience by reducing job failures. In those works, the two-player security game will be utilized. One major defect in this work is that the effect of the game was determined by the accuracy rate of the classification of the attacker. In their experiment, the clustering work was either done with a simulated or small size dataset, which had a very limited chance to contain real attackers. To conquer this weakness, our proposed framework is based on real-world large scale dataset, and it also could dynamically adapt to keep-evolving environment.

2.2.2 Virtual Machine Placement

As cloud computing becomes more popular, virtual machine placement (VMP) has become one of the most critical problems in clouds. Virtual machine placement is a process of mapping a group of virtual machines (VMs) onto a set of physical machines (PMs). VM placement schemes

can be classified as dynamic and static: [71]

- Static VM placement: in which the mapping of the VMs is fixed throughout the lifetime of the VM and it will not recomputed for a long period of time.
- Dynamic VM placement: in which the initial placement is allowed to change due to some changes in the system load or etc.

Several studies have observed the similarity between the VMP problem and the bin-packing problem [73]. There are some simple but effective heuristics for this problem, such as First Fit, Best Fit, and Worst Fit. Based on these policies, First Fit Decreasing (FFD) could improve the approximation ratio if the objects are first sorted in decreasing order of their weights. If OPT denotes the optimal number of bins, then FFD is guaranteed to use no more than 11/9 OPT + 6/9 bins [40, 42].

Saeed et al. [6] presented a security-aware approach for resource allocation in clouds which allows for effective enforcement of defense-in-depth for cloud VMs. Ravi et al. [63] proposed a framework allowed providers to impose restrictions on the allocations to be made to their hosts and users to express constraints on the placement of their virtual machines (VMs). Both of them tried to enhance the security level by modeling the cloud provider's constraints or customer's requirements as a constraint satisfaction problem (CSP). However, the placement generated by these method can only satisfy the input constraints, rather than being an optimal placement to meet multiple objectives.

Some other research utilized isolation rules in the VMP. Afoulki et al. [5] proposed a VMP algorithm which improves the security of clouds by performing isolation between users. Each user can submit a list of adversary users with whom it does not want to share a physical machine. Yu et al. [112] also proposed isolation rules to formulate the VMs placement behavior based on Chinese wall policies. Unfortunately, both of them do not take into account of the user's security preferences.

As time passed, new types of attacks targeting the cloud infrastructure appeared. For example, some attacks, such as those discussed in [33,34,50,86], exploit vulnerabilities of the hypervisor (or

Virtual Machine Monitor, VMM), e.g., Xen [15] or KVM [38]. Once an attacker compromises the hypervisor, he or she can take over all the VMs running on it. In [84] (the HYG attack), the initial stage of the attack is to locate a target VM. Upon success, the attacker will try to launch a VM on the same physical server. It is a placement based attack and the success of the attack depends on the placement strategies of the cloud, or the configuration policy of the cloud. Apparently, collocating with vulnerable virtual machines, or "bad neighbors", on the same physical server does increase the security risks of the cloud users. Thus, a lot of research on cloud computing has set the goal to improve the security level of data centers [5] . At the same time, some existing research on the co-residence based attacks, e.g., side channel attacks, demonstrates the real threat to the normal users if they are co-located with a vulnerable or malicious VM [84, 109, 110, 118]. Thus, the security aware VMP has been investigated as a practical solution to mitigate such attacks [6, 70, 88].

Our previous work [70] proposed a VM placement scheme based on security risk of each VM, and Yuchi and Shettey [113] extended it to the VM placement initialization. Both of them mainly focused on the dependency relations. Yuchi and Shettey's method also oversimplified the problem and did not reflect the potential risk caused by co-resident VMs, whose importance was discussed in [101, 115]. In [115], the author studied the characteristics of different PaaS cloud and the co-resident threat in placement policies. They implemented a memory-bus based covert-channel detection for co-residence and presented an efficient launch strategy. Their experiment concluded the risk caused by co-residency was real in popular PaaS clouds. Previously, we have investigated to periodically migrate VMs based on the game theory, making it much harder for the adversaries to locate the target VMs in terms of survivability measurement [120]. But we did not consider the risk caused by the co-resident VMs in the same physical machine.

Our work in this dissertation differs from the existing research mainly in two aspects. First, existing work simplifies the security consideration in the placement. They mainly consider the security constraints or regulations, or vulnerabilities of the VMs or hypervisor in the placement. They often overlook co-residence attacks, which is a key factor in VM placement. In our security-aware VMP, we comprehensively consider security assessment associated with placement, including the

security risks in the network connection, co-residence, VMs and hypervisor. Second, existing work often emphasizes on security while overlooking other performance factors. We propose an optimal solution satisfying multiple objectives on security, resource utilization, and network traffic.

2.3 Machine Learning

As a sub-field of computer science, Machine Learning evolved from the study of pattern recognition and computational learning theory in artificial intelligence. The goal of machine learning generally is to understand the structure of devoted training data and fit that data into mathematical models that can be understood and utilized by people. [97] Although it is a sub-field of computer science, it differs from traditional computational approaches. Machine Learning algorithms would train the models by inputs data and statistical analysis would be applied to output specific designed values that would fall into targeted ranges. Also, automatic decision-making processes could be achieved by built algorithm models based on inputs data. At the same time, Machine Learning is a continuously developing field and attracted more and more research efforts.

2.3.1 Categories of Machine Learning

In Machine Learning, learning tasks usually could be classified into several broad categories. These categories could be based on the input training data type or the target result type of learning models. Three of the most widely adopted Machine Learning methods are supervised learning which would train the model based on input and output data which are already labeled manually, unsupervised learning will let the model find the input data structure automatically without any manual label effort, and Reinforcement Learning will adjust the model by rewarding and punishing the model in a particular situation.

- Supervised learning: The input and output data provided for the training model will be labeled. A map function will be simulated by the trained model, which would generate output labels or regression for each specific input data. Common algorithms in supervised learning include Logistic Regression, Random Forest, Support Vector Machines, Naive Bayesian

model, and Artificial Neural Networks. Supervised learning is sometimes referred to as inductive learning because it goes from specific examples to more generalized rules. Also, it is typically classified into two major categories: regression and classification. In both categories, the specific relationship between provided labeled input and output data should be confirmed by the trained model and it could be used to generated correct output for future input. Noisy, mislabeled data in the input training data would significantly reduce the effectiveness of the model. At the same time, model complexity would be determined by the complexity of mapping relationships attempting learned.

- Unsupervised learning: Most of the tasks finished by unsupervised learning are clustering, representation learning, and density learning. The inherent structure of training input data should be learned without explicitly labeled. Unlike supervised learning, the performance of the model could not be compared in particular ways, since the training input and output are not labeled. The common algorithms in unsupervised learning include k-means clustering, principal component analysis, and auto-encoders. Also, since unsupervised learning can automatically identify the inherent structure of data, it is a very effective tool in exploratory analysis tasks. It also could provide some initial insights used to test hypotheses, when the job is difficult or impractical finished by a human being.

- Reinforcement Learning: Within reinforcement learning, the model is trained to make specific decisions. Generally speaking, the model will be exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task.

2.3.2 Machine Learning Algorithms

Nowadays, there are so many algorithms available for Machine Learning Models to choose. Those algorithms can be applied to almost any data problems, and selecting the proper algorithm become the most important part of any particular machine learning task. Commonly used machine learning algorithms include Liner Regression, Logistic Regression, Support Vector Machine, K-Nearest Neighbor, Random Forest and the one we used, Artificial Neural Network.

Clustering

Clustering is an unsupervised technique to group similar data samples into clusters. There are several techniques [20, 104] for clustering the input data. K-means clustering algorithm is one of the most popular clustering algorithms due to its performance and simplicity. It groups data samples based on their feature values into k clusters. Data samples that belongs to the same cluster have similar feature values. Knowing the best k value remains a challenge, although there are some proposed approaches [79].

Density-based spatial clustering of applications with noise (DBSCAN) [72] is a density based clustering models group the points that are closely packed together (points with many nearby neighbors), while marking (as outliers) points that lie in low-density regions. K-nearest neighbors (K-NN) is an instance-based learning (sometimes referred to as lazy learning) algorithm. It is used in classification problems where the classification of points is determined by the k nearest neighbors of that data point.

Support Vector Machine

Support Vector Machine(SVM) is a classification technique where given labeled training data (supervised learning), and the objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that could distinctly classifies the data points [44]. There are many possible hyperplanes available to separate the data points into two classes, and SVM is aimed to find the plane with maximum margin, i.e the maximum distance between data points of

both classes. In practice, explicitly labeled data are not provided most of the time, which make supervised learning not possible. In that case, an unsupervised learning approach can be used. Support vector clustering [18] attempts to find natural clustering of the data to groups and map new data to these groups.

Support Vector Machine are supervised learning algorithms which have been used increasingly for classification task. One of the reason is that SVMs performed effectively on high dimensional data [26] and they could be trained very quickly and effectively compared with Multi Layer Perceptrons (MLPs) [77, 95]. Most of times, SVM algorithms would act as binary classifiers (e.g., normal and anomalous data), but there are also other SVM algorithms, which have been proposed, support multi-class learning [32,41]. In Yi Han et al. [56] proposed framework, they used the semi-supervised learning algorithm, Deterministic Annealing Semisupervised SVM (DAS3VM) [92], to classify normal users and high risk users in the cloud.

Artificial Neural Network

Artificial Neural Network (ANN) are computing systems that are inspired by biological neural network and are based on a collection of connected units or nodes called artificial neurons [105]. An artificial neurons could receives a signal then processes it, and signal neurons connected to it. Artificial Neural Networks would have multiple neuron layers and each layer passes its output as an input to the next connected layer. ANNs have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, medical diagnosis and even in activities that have traditionally been considered as reserved to humans, like painting [45]. Self Organizing Map (SOM) [68] is an unsupervised model that is very commonly applied to anomaly detection. Although it falls in the category of ANNs, it is also considered to be a clustering technique, which produce lower dimensional (typically two-dimensional) representations of multi-dimensional data, which is also called a map.

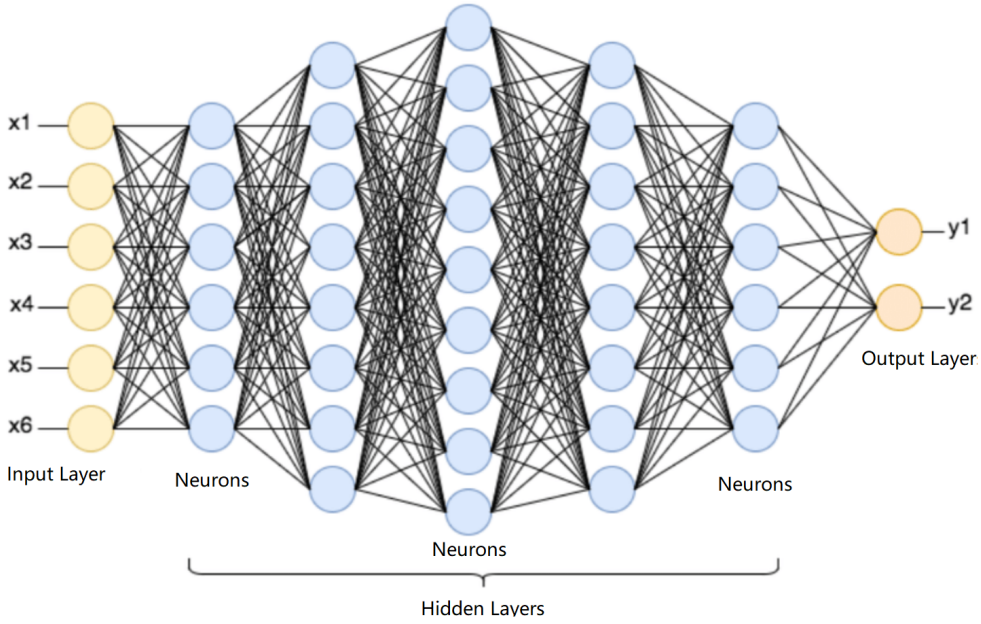

Figure 2.1: Deep Neural Network Architecture

Deep Neural Networks

Deep learning has been a challenge to define for many because it has changed forms slowly over the past decade. One useful definition specifies that deep learning deals with a neural network with more than two layers. Figure 2.1 described an abstract of a forward feeding deep neural network (DNN) architecture. In Deep Neural Network, the leftmost layer is usualy the input layer, and the rightmost layer is the output layer. The middle layers of nodes are called the hidden layers. Neurons in each hidden layer are the most important and fundamental component in DNNs. Neuron is a computational unit that takes an input from previous hidden layer and generate an output to its connected subsequent hidden layer. No matter how many hidden layers the network has, the network will behave like a single-layer perceptron.

Input Layer

In Figure 2.1, the Input layer is located at the left most. Most of the time, for typical classification task, Deep Neural Network is an appropriate tool. In that case,the input layer will take an input vector which represents the object to be classified and is passed to the connected hidden

layers for further processing. So no computation is done here within the Input Layer.

Hidden Layers

Hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or information) from the input layer to the following layer (another hidden layer or to the output layer). Typically, a Deep Neural Network consists of multiple hidden layers. The advantage of having multiple layers is that they add levels of abstraction that cannot be as simply contained within a single layer.

Output Layer

As the opposite of Input Layer, the Output layer is located at the right most of Figure 2.1. Depending on the problem, the output layer deals with the transformed vectors sent by the last of the hidden layers. For the task of classification job, the output layer with softmax activation transforms the output received from the last full connected layer to a probability distribution representing the estimated probabilities that an object belongs to each class.

Tensorflow Framework

In our experiment, we used the Tensorflow framework to finish the construction of our Deep Neural Network Model. TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer: whereas in previous parameter server designs the management of shared state is built into the system, TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural

networks [4]. Based on the tensorflow framework, we could quickly construct a model mixed with CNN submodule to handle a detail feature diagram of each service subscriber and regular DNN submodule to handle abstract feature information which was calculated based on our proposed feature metrics. GPU acceleration was applied in our experiment, which dramatically shortens our experiment time. With this benefit, we could finish further investigation regarding the optimization of deep learning model structure.

CHAPTER 3: MULTI-DIMENSION SECURITY METRICS FOR CLOUD

This chapter was published as Reducing security risks of clouds through virtual machine placement in The 31st Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec 2017). Prof. Meng Yu and Wanyu Zang provided their fully support on writing and Prof. Songqing Chen helped on proof-reading.

In a cloud, an attacker can attack a VM through different attack paths. They can attack a VM through the vulnerabilities (in the operating system, or applications) carried by the VM, the co-resident VMs, the host VMM, or VMs on different physical machines having network connections. Therefore, we cannot simply use the vulnerabilities of VMs, or the vulnerabilities of the hypervisor to evaluate the security risk of an entire cloud. We need a comprehensive approach to measure the security risks of a specific placement scheme.

For this purpose, we propose a four dimensional security risk evaluation model, as shown in Figure 3.1, to assess the security risk of a cloud. The new evaluation model covers all possible attack paths in a cloud. Four different types of security risks are described as follows.

- VM risk (R_1 in the figure): the risk/vulnerability carried by a VM itself. If a VM has more vulnerabilities than others, it is more likely to be compromised first. Vulnerable VMs can be used as stepping stones to attack co-resident VMs and underlying hypervisor to gain more privileges.

- VMM/hypervisor risk (R_2): the risk/vulnerability carried by a VMM/Hypervisor. An adversary may gain the administrative privileges via the vulnerabilities in a hypervisor or the control VM. Such vulnerabilities will enable the adversary to compromise all guest VMs on the hypervisor.

- Co-residency risk (R_3): the risk caused by the VMs co-resident on the same hypervisor. Assume that, in the figure, VM1 (an attacker VM) and VM2 (a normal user VM) share the same CPU core or are located on the same physical machine, the attacker will be able to steal the user's private information, such as the cryptographic key, via side-channel attacks.

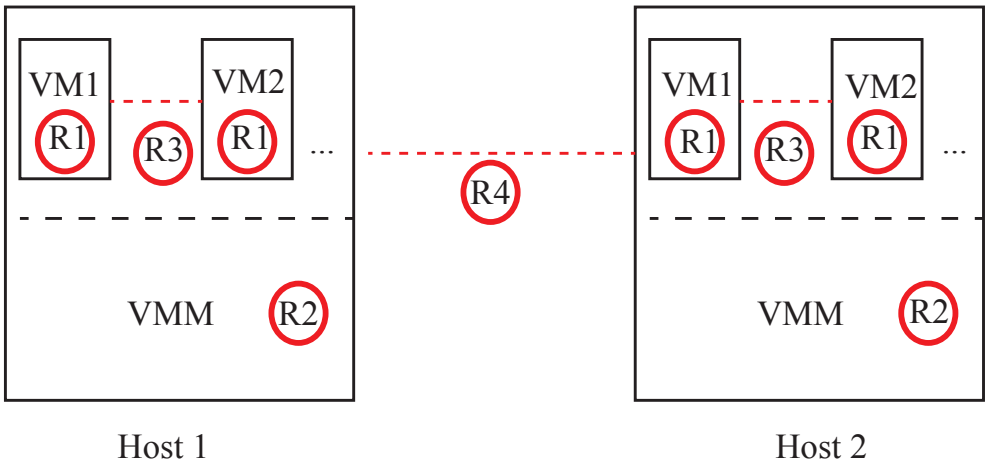

Figure 3.1: Security Risk Metrics

- Network risk (R_4): the security risk of a VM caused by the network connections. For example, VM1, located in host 1, provides web services, and the VM2, located in host 2, contains a database server. The attacker may compromise the database server through accessing the web server, e.g., SQL injection.

3.1 Explanation of the Security Metrics

Using the proposed security risk assessment model, we can assign or calculate the values of each type of the risks based on specific hardware, software, and network configuration. In this section, we provide an example to show how to quantify the values of each types of security risks, and also how to calculate the overall security risk of the entire cloud. In the example, we assume we have N VMs and M physical machines.

- R_1: CVSS (Common Vulnerability Scoring System) is a popular tool to measure the vulnerabilities of software or hardware [70]. We can use vulnerability scanner tools, such as Nessus and Qualys, to generate the vulnerability list for every VM. Based on the list, we can score each VM's risk carried by itself. One way to do that is to simply use the worst vulnerability's score, with the assumption that the vulnerable level of a VM is not higher than the worst vulnerability of that VM, although there certainly exist more refined approaches.

Also, it could be calculated by other security metrics proposed for physical hosts [30].

The CVSS score uses an interval scale of (0, 10) to measure the severity of vulnerabilities. If different tools are used to score vulnerabilities, we can choose CVSS as our baseline to normalize all vulnerability scores. For example, if a different tool scores 7 for a vulnerability which is scored 8 in CVSS, then the correlated score relationship is 7:8 between the tool and CVSS. At the same time, for some vulnerability score tools, such as Nexpose, CVSS is considered as a major factor in the score calculation model.

For a VM v_i, its VM risk R_1 is $VM^i_{R_1} = SCORE_i/10$, so its range would be limited in a scale of (0, 1). Note that R_1 is not affected by a specific placement.

- R_2: The risk level of a hypervisor is determined by two factors: its own vulnerability and the VMs running on it. For hypervisor's own vulnerability, we can use scanner tools to generate the vulnerability list and also use the most severe one to obtain the $SCORE_{hypervisor}$ from CVSS. We use $Risk_{hypervisor}$ = $SCORE_{hypervisor}/10$ to indicate the vulnerability of the hypervisor so that the value is in 0 to 1, similar to R_1.

  There are different ways to calculate how the guest VMs can affect the security of hypervisors. In this paper, we mainly consider the VM with the highest risk since this may be the most vulnerable attacking surface to the hypervisor. Assume VM v_i is on the host K, its hypervisor risk R_2 is calculated as: R^i_2 = $Risk_{hypervisor}(1 + \max(R^j_1 p_{jK}))$, where $j = 1$ to N, and $p_{jK} = 1$ if VM v_j is placed on host K as well. Different from R_1, R_2 is affected by different VM placements.

- R_3: An Attacker could deploy co-resident attack to its target if his malicious VMs are collocated on the same physical Host. There are extensive research demonstrate the attacker VM can steal other co-resident VMs' valuable information. In some scenario, an attacker may compromised a VM through some vulnerabilities. Then she or he can compromise, with enough time, other co-resident VMs eventually. In that case, the risk rate of a co-resident VM will be affected by its network connected parties. This will be a dead circle until either one is determined and we choose to set the co-residence risk first. For a VM v_i on the host

K, its co-residence risk R_3 is calculated as: $R_3^i = 1 - \prod_{j=1}^{N}(1 - R_1^j p_{jK})$, where $p_{jK} = 1$ if VM v_j is placed on the physical machine K. Similar to R_2, R_3 will change if the VM placement changes. In our subsequent work, we also proposed a machine learning based framework to quantify this type Risk Rate more accurately. We believe this type risk rate should be mainly determined by the owner of VMs, the service subscriber.

- R_4: If an attacker compromised a VM, he/she is able to compromise (with enough time) all other VMs with network connections to the compromised VM. Considering the multiple layer connection in the network, given a VM, the depth first algorithm can be applied to generate all possible attack paths. However, when a loop exists, the start point of risk calculation can be hard to be determined. We aim to identify the correlated risk level between two VMs. So we design our algorithm aiming to improve efficiency, while maintaining the correct correlations. We only consider the risk caused by direct network connections in the paper. Thus, for a VM v_i, its network risk R_4 is $R_4^i = 1 - \prod_{j=1}^{N}(1 - R_1^j)$, where v_j is a VM sending packets to v_i directly, and v_i and v_j are not on the same host. R_4 changes with different VM placements as well.

With all types of risks defined as above, we define the security risk, R^i of a VM v_i as the following.

$$R^i = 1 - (1 - R_1^i)(1 - R_2^i)(1 - R_3^i)(1 - R_4^i) \tag{3.1}$$

Considering all simplicities applied above, our metrics would represent the correct risk level correlation between two VMs. For example, a VM with a risk rate of 70% is safer than a VM with a risk rate of 80%, but without knowing how much safer to be exact.

Assurance Level of a Physical Machine Once we have the security risk of each VM, we can further calculate the assurance level of a physical machine J using the following equation.

$$AL_J = 1 - \overline{(R^i)} \tag{3.2}$$

,where VM_i (R^i is corresponding R value for VM_i) is located in physical machine J.

In general, the higher the assurance level is, the more secure physical machine will be, and the harder it is to be compromised by the attacker. However, the network communication from physical machines with lower assurance level to physical machines with higher assurance level may open an attacking channel for the attacker. Therefore, we should minimize the traffic between the low assurance physical machines and the high assurance physical machines. Besides, it can also be used as related information for potential cascade vulnerability correction [25, 60, 87]. Note that our algorithm does not include the clearance level of information stored in each physical machine.

CHAPTER 4: SECURED MULTI-OBJECTIVE OPTIMIZED VIRTUAL MACHINE PLACEMENT IN CLOUD

This chapter was published as Reducing security risks of clouds through virtual machine placement in The 31st Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec 2017). Prof. Meng Yu and Wanyu Zang provided their fully support on writing and Prof. Songqing Chen helped on proof-reading.

In this chapter, we present a Secured Multi-Objective Optimization Virtual Machine Placement algorithm (SMOOP) to seek improved solutions on specific target objective or balancing on multiple objectives. The optimization direction would be determined by Service Provider's preference.

4.1 Objectives in VM placement

Assume that we have N VMs and M physical machines. There are three values to optimize: security risk (SR), resource wastage (RW) and network traffic (NT). Our goal is to find solutions to minimize these values.

Security Risk Minimizing the security risk of the entire cloud is our first objective. The security risk of a VM v_i is $R^i = 1-(1-R^i_1)(1-R^i_2)(1-R^i_3)(1-R^i_4)$. To evaluate the security risk of the entire cloud, we need to consider the security risk of all VMs in the cloud. In security assessment, we use the median value of all VMs' risk values as the risk level of the cloud. Thus, security risk is calculated as follows.

$$f_{SR} = \mathrm{median(R^i)} \tag{4.1}$$

where $i = 1$ to N. The reason to choose the median value is twofold. First, the median value is more robust than the mean value. Second, in our placement generation, a dangerous VM will be isolated from other VMs. Thus, VMs with high risk values are outliers in our system. It is the same for VMs with low risk values.

Resource Wastage Minimizing resource wastage, while complying with the constraints, is the second objective in VMP optimization. In this paper, we consider the wastage of multiple resources, including CPU, memory, and disk. In stead of using one value to measure the resource wastage, we use a vector to represent the resources wastage.

Assume the CPU, memory and disk capacity for a host J as $\langle CPU^J, MEM^J, DISK^J \rangle$. A VM v_i requests resources as $\langle cpu_i, mem_i, disk_i \rangle$, therefore, the CPU wastage of the host J is $W_J^c = (CPU^J - \sum_{i=1}^{N} cpu_i p_{iJ})/CPU^j$, where $p_{iJ} = 1$ if VM v_i is placed on host J, otherwise it is 0. The memory wastage of host J is $W_J^m = (MEM^J - \sum_{i=1}^{N} mem_i p_{iJ})/MEM^J$. The disk wastage of host J is $W_J^d = (DISK^J - \sum_{i=1}^{N} disk_i p_{iJ})/DISK^J$.

For a physical machine J, we choose the maximum value from $\{W_J^c, W_J^m, W_J^d\}$ to represent the resource wastage of host J. We would like to minimize the total amount of the resource wastage of the entire cloud.

$$f_{RW} = \sum_{J=1}^{M} \max(W_J^c, W_J^m, W_J^d) \tag{4.2}$$

while subject to the following capacity constraints in each host J:

$$\sum_{i=1}^{M} cpu_i \times p_{iJ} < CPU^J \tag{4.3}$$

$$\sum_{i=1}^{M} mem_i \times p_{iJ} < MEM^J \tag{4.4}$$

$$\sum_{i=1}^{M} disk_i \times p_{iJ} < DISK^J \tag{4.5}$$

Network Traffic The third optimization objective is to minimize the network traffic in cloud. One way to reduce the network traffic is to identify correlated VMs that exchange high volume of data with each other, and then put them on the same physical machine if possible. We use the

following equation to measure the network traffic from VM v_i to VMv_j:

$$T_{ij} = P_{ij}/t \tag{4.6}$$

where P_{ij} is the number of packets sent from VM v_i to VM v_j in time period of t. Therefore, the total network traffic in the time period of t is:

$$f_{NT} = \sum_{j=1}^{N} \sum_{i=1}^{N} T_{ij} g_{ij} \tag{4.7}$$

where $g_{ij} = 0$ if VM v_i and VM v_j are placed on the same host, otherwise it is 1.

Workload Balance The fourth optimization objective is to maintain the workload balance in the cloud. The importance of workload balance is twofold [61]. For cloud providers, evenly distributing VMs helps to decrease the probability of over-utilizing specific servers. The best scenario is that all physical machines are active and workloads are distributed evenly among them. However, that will be conflicting with the goal of energy consumption saving. Thus, in our algorithm, we focus on spreading workload evenly among active physical machines after the number of active physical machines is determined. For the customers, they may prefer that their VMs are not all located together on the server for better survivability and availability. So far, our algorithm does not consider this requirement, but it could be easily integrated into our algorithm according to the users' preferences.

In cloud computing, over-subscription is a common practice which allows the service providers to allocate more resources to users than the servers' actual capacity [17]. As the most critical resource, the over-subscription of CPU capacity is always enabled, which would make the RAM capacity a bottleneck. In this paper, we choose the RAM usage level as the criteria of workload balance measurement.

Assume the memory usage of the host J is $Mem^J = \sum_{i=1}^{N} Mem_i \times p_d iJ$, where $p_{iJ} = 1$ if VM v_i is placed on host J, otherwise it is 0. PM_a represents the number of active physical

machines in a placement. The average memory usage in all active physical machines will be:

$$Mem_m = \sum_{j=1}^{N} Mem^j / PM_a \tag{4.8}$$

The variance of the workload balance of all active physical machines can be calculated using the following equation.

$$f_{WB} = \sqrt{(\sum_{j=1}^{N} (Mem^j - Mem_m)^2) / PM_a} \tag{4.9}$$

Note that our system does not limit the number of objectives or constraints. The users can add more objectives or constraints, such as energy or migration cost, based on their preferences.

4.1.1 Prioritize The Objectives

In our current fitness function, we have four objectives, including minimizing the security risk, minimizing the resource wastage, minimizing network traffic and minimizing the variance of the workload balance. Our algorithm tries to provide a Pareto-optimal solution which can be as good as possible in every degree based on the four objectives. To enable users to prioritize the objectives according to their business preference, we can add weight factors into the fitness function.

$$f = w_1 f_{SR} + w_2 f_{RW} + w_3 f_{NT} + w_4 f_{WB} \tag{4.10}$$

, where w_i represent different weights and $\sum_{i=1}^{4} w_i = 1$. If we consider that the security is more important than the other objectives, we can assign larger weight on the security risk.

Currently, our algorithm can optimize and balance security, the utilization of CPU, memory and disk, the network traffic and workload. Our algorithm can be easily extended to support more objectives and constraints, such as energy consumption saving.

4.2 SMOOP Design

With the proposed security metric of VMs, we can quantify the risk level of a cloud. As a typical multi-objective optimization problem, the objectives may conflict with each other. For example, if we place more VMs on a physical server, it will be less secure due to co-residency problem. However, it can reduce the resource wastage and network traffic. It is impractical to always find the optimal solution minimizing all objectives. The evolutionary multi-objective algorithms (EMOA), such as NSGAII [37], are popular solutions to such multi-objective optimization problems. Using EMOA, we can obtain Pareto-optimal solutions balancing on the objectives of security, network traffic and resource utilization.

Challenges In the bin-packing problem, objects of different volumes must be packed into a finite number of bins of volume V in a way that minimizes the number of bins used. The Virtual Machine Placement (VMP) can be considered as a bin-packing problem, where each VM needs to be placed on a physical server once and only once, with multiple dimensions of volume constraint, such as CPU, Memory and Disk. As a regular bin-packing problem, VMP is also a $\mathcal{N}P$ hard problem [69]. The challenge is that the security metrics can only be evaluated after the placement is specified. For example, in a specific placement scheme, a unique attack path exposed by co-residence may disappear in a different placement. Therefore, there is no generic function mapping a placement scheme into a security assessment value. As a result, we cannot use any existing multi-objective programming solutions to solve our problem. Furthermore, we have complicated security strategies in each placement generation, and we have to design a new crossover and a new mutation procedure in the EMOA algorithm.

4.2.1 Secured Multi-Objective Optimization based VMP

In this section, we present our Secured Multi-Objective Optimization based virtual machine Placement (SMOOP). The algorithm is shown in Algorithm 4.1. Table 4.1 describes the variables used in the algorithm.

Table 4.1: Variable Definition

Variable	Description
V_N	Number of Virtual Machines
P_N	Number of Physical Machines
N_G	Number of iteration
N_IS	Number of placement in candidate pool
N_E	Number of Elite would be passed to next iteration
N_C	Times of Crossover operation in one iteration
N_M	Times of Mutation operation in one iteration
OFF_C	Offspring generated by Crossover Operation
OFF_M	Offspring generated by Mutation Operation

In practice, FFD (First-Fit with the possible fullest node) has been widely used in VMP. It can quickly provide a placement with consideration on resource utilization. Thus, we use it to generate a baseline for future comparison in the algorithm. As shown in Algorithm 4.1, SMOOP generates hundreds of placements and passes those with high fitness value to the next iteration. In each iteration, randomly chosen parents are applied to crossover and mutation operations. An elite choosing function is designed to improve efficiency. For each generated temporary placement in an iteration, we apply a multi-objective evaluation function to assign ranking values. The highly ranked placements are put into a candidate pool, and used as the parents for next iteration. The preference of multi-objective evaluation can be adjusted (described in Section 4.1.1) in our algorithm.

In the *initialization phase*, the consideration for the migration cost can be avoided. Our goal is to search for the best possible placement plan based on the multi-objectives requirement. The crossover operation is used to improve the overall efficiency. In the *re-optimization phase*, which is triggered by adding VMs or removing VMs, the migration cost is considered as an important factor in the mutation operation to limit the number of migrating VMs (in `switch()` function of Algorithm 4.3). Note that our goal is to improve the survivability of entire cloud, so we do not optimize our solution for a specific VM. Therefore, our approach may lower the security level of a specific VM, while the overall security level of the entire cloud can still be improved.

Algorithm 4.1 SMOOP

```
Ensure: Canditate = init() by Strategies
  for G = 1 → N_G do
      for i = 1 → N_E do
          Elite[i] = Elite_choosing(Canditate)
      end for
      for j = 1 → N_C do
          (X, Y) = Random_select(Canditate)
          Off_C[j] = Crossover(X, Y)
      end for
      for k = 1 → N_M do
          X = Random_select(Canditate)
          Off_M[k] = Mutation(X)
      end for
      Temp = fitness_sorting(Elite, Off_C, OFF_M)
      for i = 1 → N_G do
          Candidate[i] = temp[i]
      end for
  end for
```

4.2.2 Crossover And Mutation Operation

The crossover operation, shown in Algorithm 4.2, is one of the key elements in our algorithm. The main purpose of the crossover operation is to guarantee that there is always a chance to generate newly improved placement based on the existing placement in the current iteration.

Isolated Zones

Since the security is a key factor in the placement generation, we introduce isolated zones in our algorithm to accommodate different security demand. Physical machines with the highest hypervisor risk levels are put into isolated zones. The most dangerous VMs and VMs connected to them are placed into the isolated zones by priorities. The purpose of the isolated zones is to isolate the most dangerous VMs first and reduce the number of attack paths through network connections.

If all physical machines use the same copy of hypervisor, the vulnerabilities of all the hypervisors will also be the same. In such a situation, we have the following assumptions: 1. The possibility to compromise any physical machine through the hypervisor attack surface for a spe-

Algorithm 4.2 Crossover(X, Y)

```
Temp = Blank_Placement_Object
Rank_A = Rank(X)
Rank_B = Rank(Y)
for i = 1 → P_N do
    if Rank_A[i] > Preset_value then
        temp ← Rank_A[i]
    end if
    if Rank_B[i] > Preset_value then
        temp ← Rank_B[i]
    end if
end for
Remove_duplicate_VM(temp)
Ran = Gen_Random_list(VM)
for i = 1 → V_N do
    if Ran[i] is not in temp then
        temp ← Ran[i] by Strategies
    end if
end for
Return Temp
```

cific VM is the same. 2. If the communication bandwidth between two VMs is larger than zero, the possibility to compromise one VM through another VM will be non-zero.

We propose five security related strategies to reduce security risk during each placement generation.

- Placement strategy I: Deploying a VM into a physical machine which has network connections with it.
- Placement strategy II: The high risk VMs should be deployed into the isolated zones.
- Placement strategy III: The low risk VM without any connection with VMs in isolated zones should be deployed into low risk physical machines. Strategy II and III generate physical machines that contain only low risk VMs and have no network connections with high risk VMs in isolated zones.
- Placement strategy IV: The marked lowest and highest hypervisor risk physical machines should have a higher probability to be kept during crossover operation. This is based on our

strategy II and III.

- Placement strategy V: If a VM on one physical machine has connection with a VM on a different physical machine, we should migrate one of them to the same physical machine.

We use *Placement strategy I* as an example to explain how the placement strategy can reduce the security risk. The motivation of this strategy is to reduce R_4 caused by network connections.

Assume that physical machine PM_I already has a set of VMs, $S_i = (v_a, \ldots, v_i)$ and PM_J has a set of VMs, $S_j = (v_b, \ldots, v_j)$. There is no network connections between S_i and S_j. When a new VM v_n is to be placed and it has network connections with at least one VM in PM_I. If v_n is placed into PM_I, S_j on PM_J will not be affected by attacks through network connections. It will only affect S_i on physical machine PM_I. For any VM $v_i \in S_i$, R_3^i will be updated by adding the new VM v_n. Assume that the current co-residency risk of v_i is Old(R_3^i), then we have

$$New(R_3^i) = Old(R_3^i) + (1 - Old(R_3^i))R_1^n \tag{4.11}$$

where R_1^n is the risk level of v_n.

If v_n is placed into a physical machine PM_J, not only will R_3 be updated, but R_4 introduced by network connections will also be increased by $\sum_{v=c}^{K} T_{nv}$ (All traffic through VM v_n to connected VMs on physical machine PM_I).

According to the security metrics defined earlier, in this case, the co-residence risk of each VM $v_j \in S_j$ will be

$$New(R_3^j) = Old(R_3^j) + (1 - Old(R_3^j))R_1^n \tag{4.12}$$

Also, R_4 of VM v_j increases as well.

$$New(R_4^j) = Old(R_4^j) + (1 - Old(R_4^j))R_1^n \tag{4.13}$$

Therefore, following strategy 1, allocating VM v_n on to PM_I rather than PM_J can reduce security risk. Other placement strategies can reduce the security risk as well.

In our implementation, Strategies I, II, III are applied for VM deployment and Strategies IV and V are applied for crossover and mutation procedure. When a VM need to be deployed, <VM, PM> pairs would be generated, in which PM would be all available ones. Each pair has an associated "strategy fit set"; if a given VM placement fits a particular strategy, then the strategy number will appear in this set. Each strategy fit set will then be evaluated in order to decide a final deployment choice. To do so, the set will be tested to determine whether it satisfies each strategy, in order of weight, from highest priority to lowest priority. This process will progressively eliminate placement choices that do not satisfy a given strategy, successively shrinking the set of option. For example, given the strategy fit sets 1, 2, 1, and 2, 3, we examine these sets for membership of strategy 1. This eliminates set 2, 3, as it does not satisfy the highest priority strategy (strategy 1). We continue by testing the remaining sets for membership of strategy 2, and so on. This discards set 1, 2, as it does not satisfy the next-highest priority strategy (strategy 2). As a result, only set 1, 2 remains, indicating that its associated VM/PM deployment will be utilized. The selection rules (i.e., the system of weighting strategies) can be adjusted in real-time. For instance, after running for a period of time, it might be optimal for strategy 2 to be promoted to the highest priority; as a result, the weighting factors for strategies 1 and 2 would be swapped. Also, new strategies should be added, in response to the runtime environment. In this way, the system may evolve to fit real-world situations.

Mutation Operation

Mutation operations, shown in Algorithm 4.3, operate on a randomly-chosen temporary placement, trying to obtain an improved result. Its purpose is to keep evolving the existing placement with limited migration cost.

When a Pareto-optimal solution is generated, our algorithm checks the workload balance in every physical machine, migrating marked VMs among physical machines. In the `switch()` function of the algorithm, a VM can only be switched (migrated) to another physical machine to which it has network connections. The strategy is to guarantee that random evolving will not

Algorithm 4.3 Mutation(X)

```
Temp = Blank_Placement_Object
temp ← X
for i = 1 → Preset_Maximum_number do
    temp ← Switch(temp) by Strategy of Switching VM
end for
Return Temp
```

jeopardize the isolation of dangerous VMs and reduce unnecessary switching operation.

4.3 Evaluation

We implemented our solution in Java and conduct all experiment in a simulation environment. All input data are provided through configuration files. Multiple threads are used to improve the performance. We randomly generate a large number of VMs with different parameters to evaluate SMOOP. Every VM needs to be deployed into one physical machine, without considering the migration cost. In our evaluation, for each VM, we randomly assign the requirement of CPU, memory, and disk. The vulnerability score is assigned based on the uniform distribution. Following the same method, we configure the physical machine. The numbers of VMs and physical machines vary between experiments.

4.3.1 Computing Complexity

Assume that there are M physical servers and a total of N VMs. Our algorithm iterates for k times with candidate pool size of P in each iteration. In our case, N is far larger than other factors here. The value of our objectives for each Virtual Machine Placement (VMP) could be calculated in bounded by $O(MN^2)$. Assume that the fitness value of a VMP can be calculated in constant time with a fitness function, which is $O(P)$ for the entire candidate pool. We sort the candidate pool with complexity of $O(PlogP)$. The elite choosing operation will take constant time. The crossover and mutation operation in our algorithm is bounded by $O(MN^2)$. The overall combined complexity of our algorithm is $O(k(MN^2 + PMN^2 + P\log P))$, thus $O(kPMN^2)$. If we consider the cascade effect of network, the computing time of risk value for each VM would

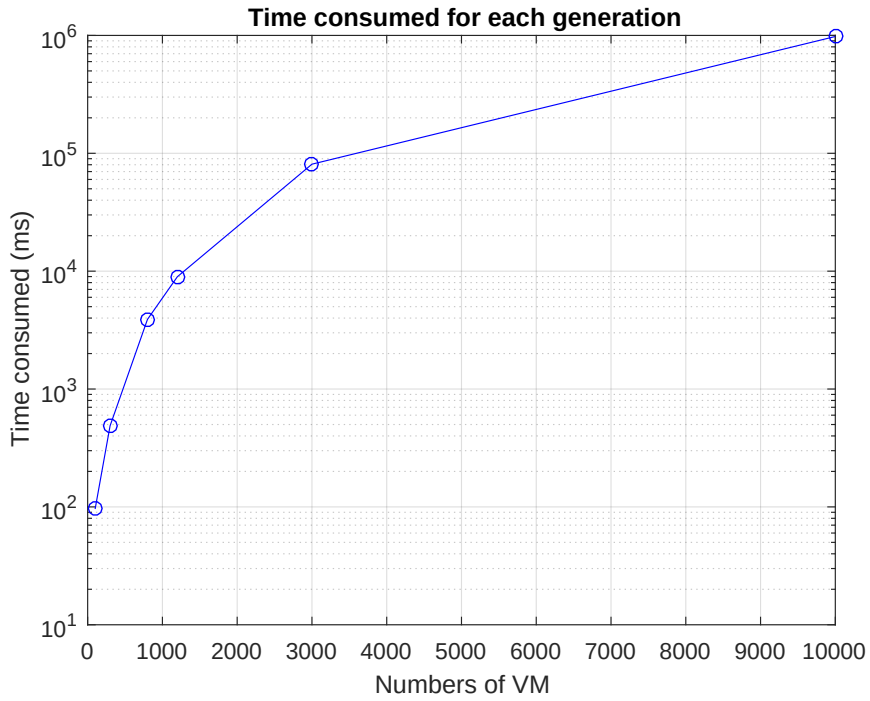

Figure 4.1: Scalability

be changed from $O(N^2)$ to $O(N^3)$, which is caused by the construction of all possible attack paths with a Depth-First search. Then the complexity of our algorithm will be bounded by $O(kPN^3)$. In order to improve efficiency, we simplified the computing of risk value, while maintaining correct correlations. Also, multi-threading is used in our implementation, and we used 8 threads to conduct elite selection, crossover and mutation operation simultaneously.

We test our implementation in a 8 core processor with 16GB memory. The overall performance of our algorithm is affected by the number of VMs, the number of physical machines, and the number of candidate placement generated in each generation. Figure 4.1 shows the computing time for each generation under the following setting: 1. 100 different placements are generated for each generation. 2. 270 operations are done in each generation. With 10000 VMs and 500 physical machines, each generation takes about 15-20 minutes. If we reduce the number of VMs to 3000, each generation takes around 2 minutes.

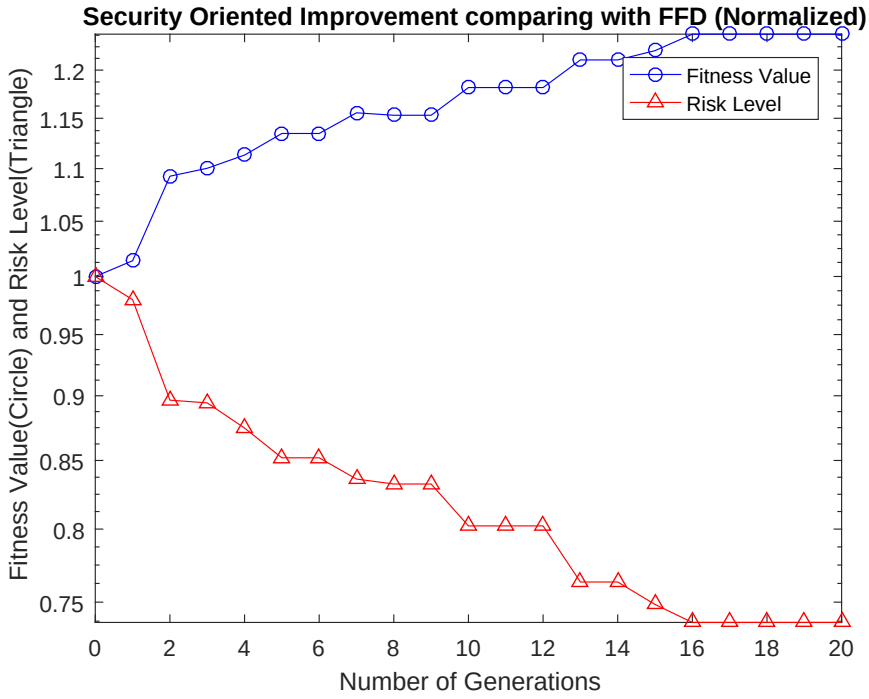

Figure 4.2: Comparing with random-FFD

4.3.2 Effectiveness in Risk Reduction

The security risk is a key consideration in VMP. To evaluate if our strategies can improve the security level of the entire cloud, we conduct the experiments considering the risk level as the only objective in the placement. Figure 4.2 shows the security risk with 800 VMs and 60 physical machines.

At the beginning of each simulation, we always generate 100 placement with the random-FFD algorithm and use the lowest risk level as the baseline reference. We collect the placement with the lowest risk level in each generation. Within 20 generations, the risk level of the entire cloud can be reduced by 25% to 30%.

As we discussed before, the network from low assurance level physical machines to higher ones would always be minimized to make the network more secure. Figure 4.3 shows the non-secure network traffic bandwidth is also dropped about 3% to 5% with other multiple objectives optimized.

Figure 4.4 shows the security risk with different number of VMs and physical machines in each

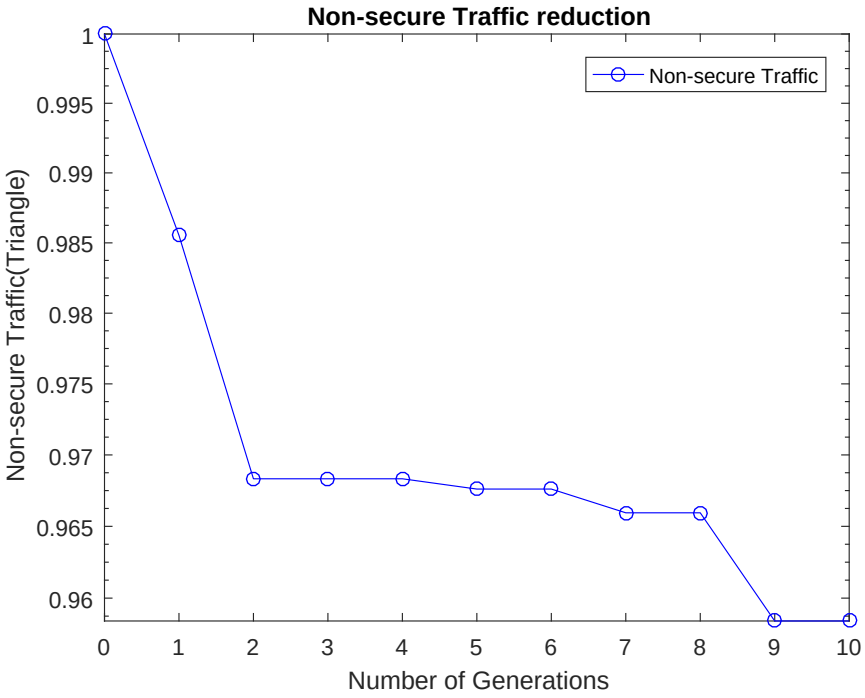

Figure 4.3: Non-secure Network Traffic

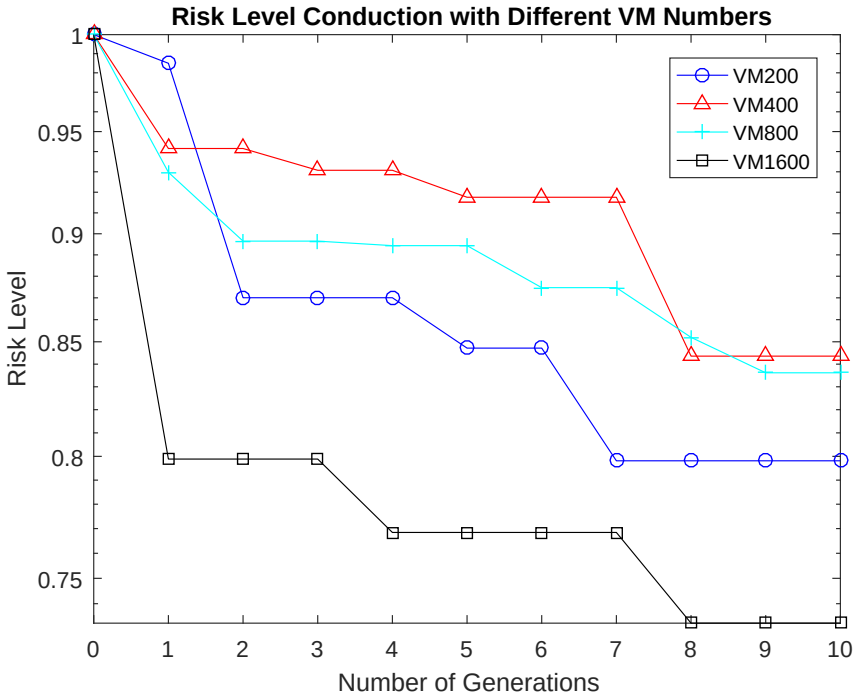

Figure 4.4: Security improvement with different number of VMs.

generation. Despite the increased number of the VMs, the median value of the risk level of VMs is stable within the range of 0.82 to 0.84. If we check the placement with the lowest risk level in the first generation, our algorithm improves with the increased number of the VMs. We repeat our experiment 20 times with different numbers of VMs and physical machines. The reduced risk level is from 5% (400 VMs and 20 physical machines) to 15% (6400 VMs and 400 physical machines) just in the first generation.

4.3.3 Effectiveness in Workload Balance

In our algorithm, we focus on spreading workload evenly among active physical machines once the number of active physical machines is determined.

Figure 4.5 shows experimental results with weight setting: (0.4 (risk level), 0.4 (Workload Balance), 0.1 (Resource Wastage), 0.1 (Network Traffic)) in an environment of 400 VMs and 60 physical machines. Per the setting of our experiment, the memory request of a VM ranges from 4GB to 40 GB and each physical machine could provide 256GB memory. Figure 4.6 demonstrates the usage of memory in active physical machines with the best fitness value. VMs are aggregated into 31 active physical machines and the usage of memory in those physical machines achieves 92% and higher, which ranges from 237GB to 255GB.

4.3.4 Effectiveness of Multi-Objective Optimization

In this evaluation, we consider multi-objectivs on risk level, resource wastage, and network traffic.

Figure 4.7 shows experimental results with weight setting (0.8 (risk level), 0.1 (Resource Wastage), 0.1 (Network Traffic)) in an environment of 800 VMs and 60 physical machines. The risk level has weight of 80%, resource wastage and network traffic have weight of 10% for each in the fitness function. We collect the placement with the best fitness value. The baseline is still the best placement chosen from 100 random-FFD placements. If a physical machine can hold hundreds of VMs, the placement generated by FFD will be using the minimum number of physical

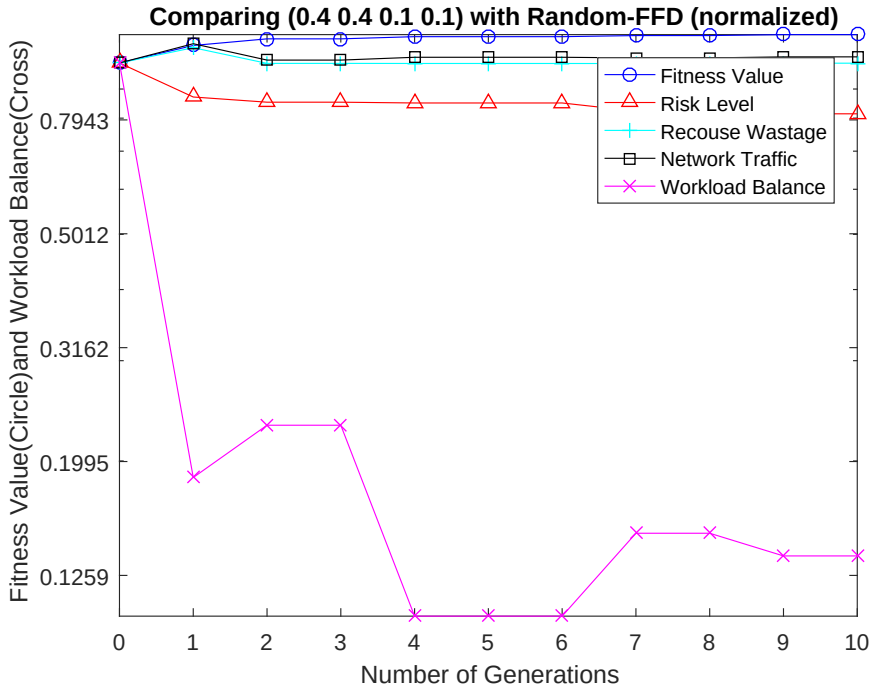

Figure 4.5: Multi-objective optimization with Workload balance

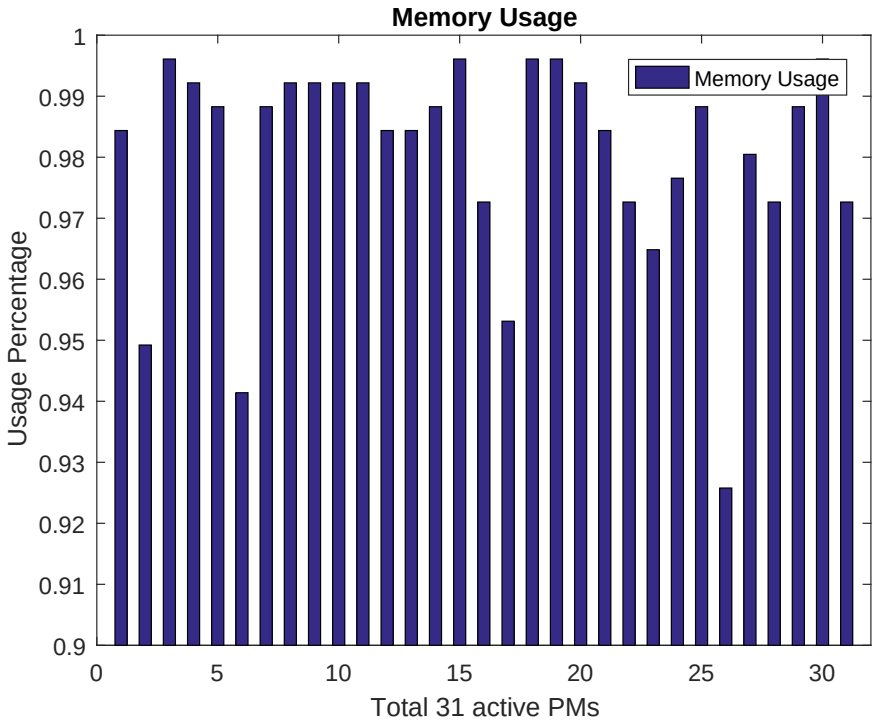

Figure 4.6: Memory usage in cloud after balanced

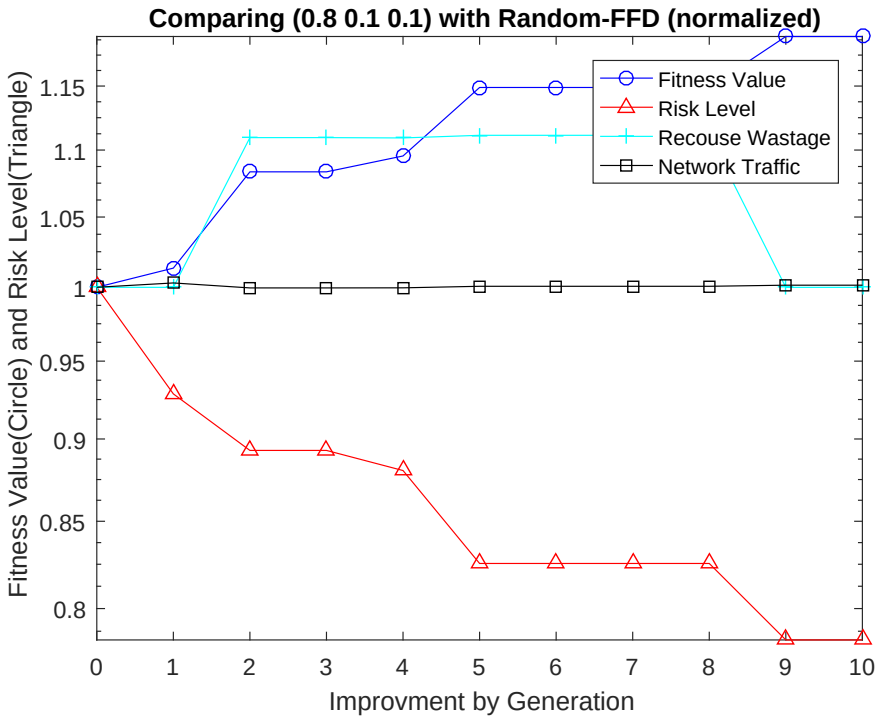

Figure 4.7: Multi-objective optimization

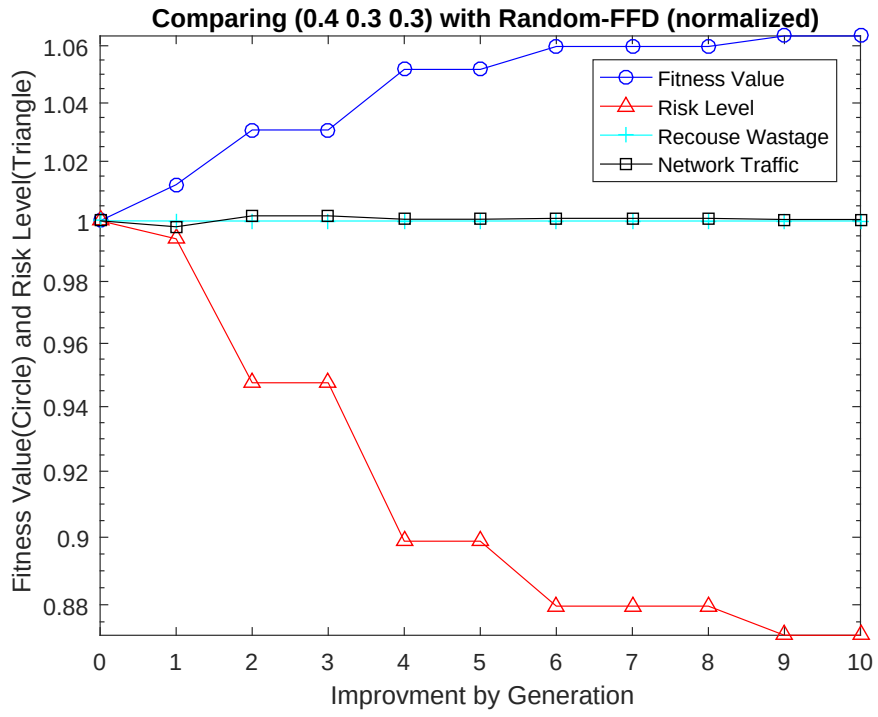

Figure 4.8: Multi-objective optimization 2

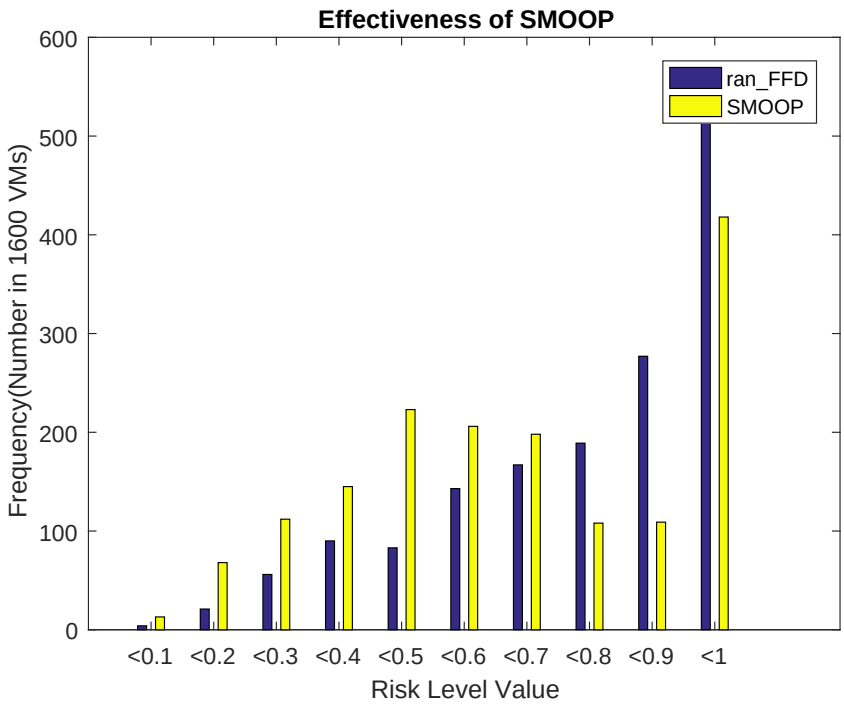

Figure 4.9: Comparison with Distribution in 1600 VMs and 120 PMs

machines. With setting of (0.8, 0.1, 0.1), the active number of physical machines and resource wastage are limited, with much improved security.

We also run the experiment with weight setting (0.4 (risk level), 0.3 (Resource Wastage), 0.3 (Network Traffic)) in an environment of 3000 VMs and 200 physical machines, and the results are shown in Figure 4.8. Since the resource wastage and network traffic have higher weights, the allowance of resource wastage was controlled and it also affects security improvement we can achieve. A cloud provider can always change the optimization preferences by changing the weights of different objectives.

4.3.5 Comparison with random-FFD algorithm

In the experiment, we use with 1600 VMs and 120 physical machines, we generate 100 placements with the random-FFD algorithm. We choose the placement with the lowest median value of risk level. After running our algorithm to reduce the risk level, we choose the best placement.

As shown in Figure 4.9 and Figure 4.10, we can see that the risk level of the entire VM set has

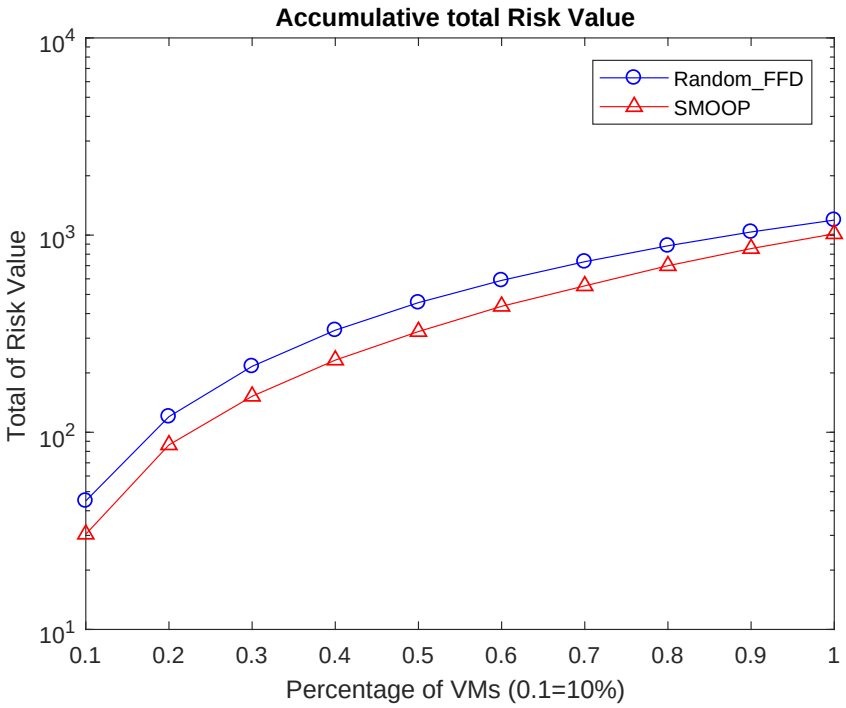

Figure 4.10: Accumulative Risk Value

been effectively reduced. In the figure, the X-axis is the risk level value of VMs. For example, <0.2 means that the risk level is between 10% and 20%. With 1600 VMs, the risk level under 50% is improved by 15% to 35%. The risk level above 80% dropped from 54% to 33%. The experimental results demonstrate our placement strategies can greatly improve the security level of the entire cloud.

Above experimental results demonstrate the effectiveness of our approach and the improvement compared to existing solutions. On the other hand, although security risk of cloud was quantified, but it is just a coarse-grained relative value. The effectiveness of median value is proved, but it might not be the best choice. At the same time, as we discussed, the priority of strategies could be adjusted in real-time and new strategy might need to be added, in response to the runtime environment. In this way, our system could keep evolving to fit real-world situations.

CHAPTER 5: COMPATIBLE WITH LATEST VIRTUAL MACHINE ALLOCATION POLICY

This chapter was published as a portion of Risk-aware multi-objectives optimized virtual machine placement in the cloud in Journal of Computer Security. Prof. Meng Yu and Wanyu Zang provided their fully support on writing and Prof. Songqing Chen helped on proof-reading.

The nature of our proposed SMOOP algorithm makes it more suitable to the VMP initialization and re-optimization procedure. When a new VM is deployed or re-activated, the new placement should be generated based on the current one to achieve the multi-objective optimization, while keeping the low migration cost in mind. Another major improvement we could make is that service subscriber's constraints should be taken into consideration. More importantly, we believe that Co-Residency Risk rate of a VM, R3 in our security metrics, should be mainly determined by its owner. We finished the improvement in two steps. In this chapter, we improved our SMOOP algorithm to better handle incremental deployment task by collaborating with latest Virtual Machine Allocation Policies.

5.1 Latest VM Allocation Policy

To the best of our knowledge, there are only two existing solutions using VM allocation policies to defeat the co-resident attacks. In [14], the authors proposed a Co-Location Resistant (CLR) algorithm. All servers are labelled either in open or closed state and can switch the state as needed. Open (closed) means the server can (cannot) receive more VMs. CLR would try to maintain a fixed amount (N_open) of servers staying open state, and allocates a new deployed VM to one of these servers randomly. If the allocation procedure failed, the server will be switched into the closed state, and another inactive server will be activated and marked as open to receive new deployed VM. Specifically, the larger N_{open} is, the more co-location resistant the algorithm becomes. Therefore, the best case scenario is that all servers are open. Han et al. [55, 56] proposed a Previous-Selected-Server-first (PSSF) allocation policy to increase the difficulties of co-locating

with target VMs and decrease the attack efficiency. The workload balance and power consumption are considered in the allocation as well. They defined and modeled some security metrics to assess the attack and compared the difficulty of achieving co-residence between their PSSF and CLR. They implemented the experiments on the simulation environment CloudSim [2,28]. Per our study, both allocation policies can be adapted into our mutation operation with minor modification.

5.2 Integrating with SMOOP

As we discussed above, in order to better handle the incremental deployment task, we modified our original SMOOP algorithm. As presented in Algorithm 5.1, the crossover operation is removed and the mutation operation is modified to adapt the latest VM allocation policy. The mutation operation will always operate based on the current placement, in order to limit the number of migrated VMs.

Algorithm 5.1 SMOOP-phase2

```
Ensure: Canditate ← Insert(Current, New_VM) with Pre-set VM allocation Policy
  for G = 1 → N_G do
      for i = 1 → N_E do
          Elite[i] = Elite_choosing(Canditate)
      end for
      for k = 1 → N_M do
          X = Random_select(Canditate)
          Off_M[k] = Mutation(X)
      end for
      Temp = fitness_sorting(Elite, Off_C, OFF_M)
      for i = 1 → N_G do
          Candidate[i] = temp[i]
      end for
  end for
```

Also, we considered two currently existing VM allocation policies in the mutation operation to defeat the co-residence attack. The CLR (Co-located Resistant) could be easily implemented into our mutation operation without major modification. The mutation operation is revised as algorithm 5.2.

Algorithm 5.2 Mutation(X)

Temp = $Blank\_Placement\_Object$
temp ← X
temp
for $i = 1 \rightarrow$ Preset_Maximum_number **do**
    temp ← Switch(temp) with Pre-set VM allocation Policy
end for
Return Temp

5.3 Evaluation

In our previous work, we did not consider VMs with its owner, Service Subscriber's information. All VMs were treated as the same. This setting simplifies the modeling, and it well fits for the situation of not-knowing the advisor's account. However, it makes PSSF (Previous-Selected-Server-First) hard to be adapted. To adapt PSSF, two preliminary settings should be supported. The VM deployment history for a particular user should be maintained and all VMs need to be grouped by their owner's account. Afterwards, PSSF can be integrated as other VM allocation policies in our mutation operation.

In our experiment, we generated simulated dataset to test our modified SMOOP algorithm. Since we have the service subscriber's information taken into consideration, subscriber's preference and constraints could be applied at the same time. Thus, we modified our security strategies as below:

- Placement strategy I: Deploying a VM into a physical machine which is fit for its owner's preference.
- Placement strategy II: Deploying a VM into a physical machine which is fit for co-resident VM's owner's constraints. Strategy I and II would have the top priority and make sure subscriber's preference and constraints would be satisfied.
- Placement strategy III: The high risk VMs should be deployed into the isolated zones.
- Placement strategy IV: The low risk VM without any connection with VMs in isolated zones should be deployed into low risk physical machines. Strategy III and IV generate physical

machines that contain only low risk VMs and have no network connections with high risk VMs in isolated zones.

- Placement strategy VI: If a VM on one physical machine has connection with a VM on a different physical machine, we should migrate one of them to the same physical machine. But the mitigation should not violate prioritized strategies.

Our experiment demonstrated the overhead to calculated a new placement which maintain the same security level is acceptable. Since our test data are all simulated, we would further extend our experiment with data collected from real world and the effectiveness of our security strategies could be better verified.

CHAPTER 6: QUANTIFY CO-RESIDENCY RISK THROUGH MACHINE LEARNING

For a cloud computing system, one of the most critical and fundamental components is the Virtual Machine (VM). Multi-tenant deployment of VMs significantly improved cloud resources utilization, which would benefit the cloud service provider. On the other hand, sharing resource among clients makes the cost of service more affordable and minimizes the maintenance overhead of the computing resources. Along with these advantages, Cloud computing acts as more and more important roles in our daily life, and also attract more attention from attackers. Since VMs could be exposed to various types of security threats, lots of research efforts were also made into this particular field.

Logical isolation between VMs deployed on the same server is provided by Virtualization techniques [16]. Under the control of the hypervisor, one VM can only access its own assigned partition of hardware resources, such as computing power or data storage. One VM should never be affected or compromised by another VM's activities. However, in the real world, many stealthy attacks can happen. For example, cache utilization can determine the execution time of cache read operations [85]. As the result, various types of side channels between attacker's VMs and victim's VMs could be built and sensitive information could be extracted [59, 85, 111, 116], which also was called the co-resident attack. One of the most influential co-resident attacks is to extract private keys [117]. To successfully construct a side-channel requires overcoming challenges including core migration, numerous sources of channel noise, and the difficulty of preempting the victim with sufficient frequency to extract fine-grained information from it. The authors [117] demonstrated the attack in a lab setting by extracting an ElGamal decryption key from a victim using the most recent version of the libgcrypt cryptographic library.

To address the problem, a lot of research work has been done to defend against co-residency attack. Most of them [13, 64, 66, 89, 96, 100, 102, 107, 119] focused on preventing the construction of side channels. One major defect of those methods is that normally substantial changes need

to be made to existing systems which could be expensive to a cloud provider. Other researchers investigated the problem from different perspectives. Sundarewaran, et al. [94] proposed a defense mechanism to identify abnormalities in CPU and RAM utilization, cache miss activity. Yi Han et al. [56] proposed a mechanism to identify the co-resident attacker's behaviors in the absence of any defense. Generally, an attacker has to start a much larger number of VMs than normal service subscribers to achieve co-residency with their target. In later work [54], they proposed a semi-supervised learning based defense strategy to increase the attack cost. Followed the work, more research was done [57, 83] based on Game Theories. In those works, a two-player security game was utilized. One major defect in this work is that the effect of a game was determined by the accuracy of the classification of the attacker. In their experiments, the clustering was either done with a simulated or small size dataset, which had a very limited chance to contain actual attackers. To improve the whole security level of the cloud system, we proposed a four-dimensional security risk evaluation model to cover all possible attack paths in a Cloud [51]. In our previous work, the co-resident risk was also taken into consideration and a coarse-grained method was proposed to quantify the co-residency risk, which is not sufficient.

To better understand and predict workloads for improved resource management of large cloud platforms, Microsoft made an open VM trace dataset which contained over 25,000 service subscriber account with 4.6 million VMs and 3.1 billion utilization readings record on 2019 [31]. A. George et al. [10] provided another four new traces from two private and two high-performance computing clusters to conquer the over-fit issue with original dataset when evaluating the generality of new research. This large-scale dataset was widely used in the workload related research field. C Ignacio et al. proposed an adaptive resource management for Virtual Machines and verified their work through it [29]. In Rong shi's work [90], they used this dataset to extrapolates the workload to a bottleneck mode. At the same time, we believed these new real-world large scale VMs trace datasets provided a great opportunity to profile normal service subscriber's behavior. In our work, we first applied these VMs trace datasets to quantify the risk rate of the whole cloud system. Thus, we make the following contributions in this chapter:

- we proposed an effective feature metrics to handle the clustering task on real world large-scale dataset.

- Based on the large scale Microsoft Azure dataset, we profile the behavior pattern of normal service subscriber within our proposed feature metrics. Service Subscribers were clustered into several categories. Inactive (Normal), Period Active, and Extreme Active labels were assigned accordingly to the dataset.

- We proposed a framework to dynamically quantify the co-residency risk level for a specific VM. Based on our experimental results, our model demonstrated a robustness to new datasets and the accuracy was verified by examination of F Measuring Matrix.

6.1 Problem Formulation

In our previous work [52], we proposed a coarse-grained metric to quantify the risk level of the whole Cloud. In our metrics, the co-residence risk acts as the most important portion of our work. In our previous work, we could only use some simulated data and calculated the co-resident risk level based on the vulnerabilities of each VM, which is insufficient. In the real world, the service subscriber would determine the co-resident risk level of their owned VMs. To solve this issue, we tried to build a fine-grained model to better quantify the co-resident risk based on the service subscriber. The performance is another serious factor need to consider and the framework we proposed should have the capability of adapting to the dynamic changing environment.

6.1.1 Threat Model and Security Assumption

To accomplish the co-resident attack, attackers have to make effort to deploy their own VMs on the same physical server with the victim's VMs. In other words, one major portion of the cost for the attacker would be determined by the procedure of co-location with the target. Service providers should have the capability to collect any information they need to profile a service subscriber. For stealth purposes, the attacker wants to reduce the possibility of being detected as much as possible.

Addition to our previous security assumption regarding the whole cloud environment, we have the following assumptions specifically for profiling and classification of service subscribers: ① Service provider has no knowledge about the attacker's presence; ② Service provider has no or very limited knowledge about the attacker's behavior pattern; ③ Attacker's behavior pattern could be evolved and different in every time period; ④ Service could verify a limited amount of normal service subscriber (mark as legal); ⑤ Attacker has no way to compromise the data collected by the service provider for profiling purpose; and ⑥ Attacker can't compromise the detection system implemented by the service provider.

Those above assumptions ensure that our proposed defense mechanism works practically in the real world. Also, the attacker's presence should be a rare event in the real world. We could believe that the major portion of data collected by service providers should belong to the legal service subscribers. A limited amount of legal service subscribers could be verified and marked by service providers.

6.2 Framework Overview

Figure 6.1 demonstrated the diagram of our proposed framework to classify service subscribers and better quantify the Co-Resident Risk Rate. It contained five necessary components to generate a quantified co-resident Risk Rate and one optional component to make our framework more adaptive to the practical environment. Followed is a brief description of each component:

Pre-Process In the Pre-Process component, collected raw data would be processed and organized by applying the feature metrics determined by the service provider. An explanation of feature metrics could be reviewed in the following section. A service provider can update the feature metrics based on their unique requirements.

Clustering The clustering component is responsible to handle the task to categorize the service subscribers. The pre-chosen clustering algorithm will be applied and output the candidate categories of subscribers for Partial Labeling Component.

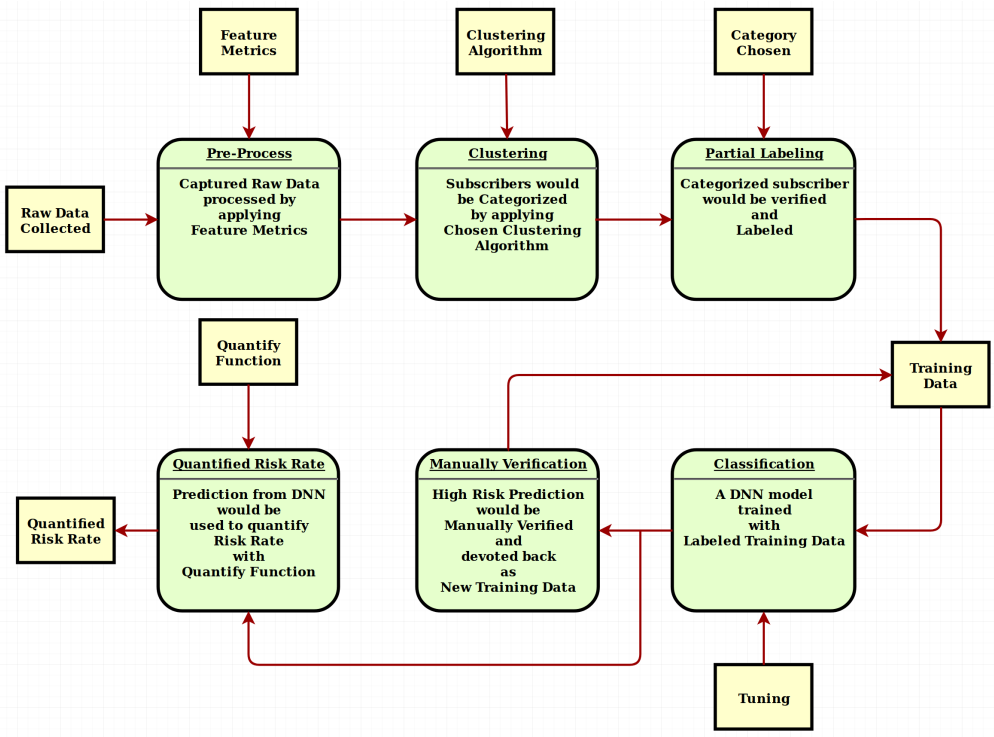

Figure 6.1: Overview of Framework

Partial Labeling In this component, the service provider could finish the examine of candidate categories of subscribers and determine the principle of labeling for each category. Some categories of subscribers could be dropped in this component if it is believed not relevant to the quantify task. In our experiment, the dummy code was assigned to each category automatically and the training data set will be prepared for the classification component.

Classification With the training data, a Deep Neural Network will be trained to perform the classification task for incoming subscriber. Hyper-parameter tuning would be performed to increase training efficiency.

Quantified Risk Rate The risk rate of incoming subscriber will be evaluated based on the prediction of classification component and pre-defined quantify function. In our experiment, the probability of predicted subscriber belongs to the high-risk group will be used as the Quantified Risk Rate directly.

Manually Verification It is an optional component to make our framework more adaptive to the real world. The prediction of new subscriber data could be verified manually by the service provider and devote it back to training data to make our proposed DNN model adapt to the latest environment.

6.2.1 Co-resident attacker's potential behaviors pattern

To successfully accomplish co-resident attack, the attacker need to achieve the co-residency with their target first. Several existing work [56] studied this particular topic and a general behavior pattern could be conclude as follow:

- step ①: a number of VMs will be started simultaneously or asymmetrically.
- step ②: check if any these started VMs were deployed into the same physical machine as target VMs.
- step ③: In order to save the cost, turn off those VMs which failed to co-locate with the target VMs (This step could be skipped by sophisticated attacker).

The above steps could be repeated several times by attacker until co-residency is achieved. Therefore, comparing to the normal service subscriber, a considerable larger amount of VMs have to be started by the attacker. To minimize the cost, VMs failed to achieve co-residency should be short-lived. At the same time, those started VMs with a long life should stay active most of the time, where being active means a high CPU utilization rate could be observed.

In the real world, the attacker might use a single subscriber account or multiple subscriber accounts to accomplish the attack. Nowadays, the Cloud service provider allows a service subscriber to start multiple VMs simultaneously in a single account. Thus, the pattern between these two situations could be the same.

There is another serious factor to consider, the presence of attack could be very rare in the real world, which means major portion, if not all, of data we collected, should from normal service subscribers. We believe that the behavior pattern of normal service subscribers should demonstrate

some derivation from the above pattern of attackers, which need to be verified by our experiments. In other words, if service providers can successfully profile normal service subscribers, the attacker has to adjust their behavior pattern to match the normal pattern to reduce the chance being detected, which will increase the attacker's cost accordingly.

6.2.2 Feature Metrics to profile normal service subscriber

In our proposed framework, the feature metrics is used in the Pre-Process component to process collected raw data. As we mentioned before, service provider can define their own feature metrics for their unique requirement. In our experiments, we proposed an eight dimension feature metrics to profile service subscriber:

- N - the total amount of VMs created and deployed by a specific service subscriber.
- T - the average interval time between starting two VMs. Note that it is the time between starting the ith and the (i+1)th VMs, rather than difference between the stopping ith and the (i+1)th VMs.
- M - the median memory size among VMs for a specific service subscriber.
- A - the overall active rate for a specific service subscriber. Detail explanation would be explained by following section.
- W - the average amount of active VM in each time stamp for a specific service subscriber.
- I - the median of average CPU utilization rate among all VMs in each time stamp for a specific service subscriber.

Feature 1 - 4 is the overall analysis factors for each service subscriber. Feature 5 - 6 are detail features for a specific service subscriber. The reason we have two types features, because not only we want to know the whole picture of subscriber's behavior, we also want to more accurately profile each subscriber's behavior pattern and build the characteristic image with detail inside information.

M - Median Memory Size For each service subscriber, the memory requested by the VM_i created by the subscriber is recorded as M_i, then the feature M is calculated by follow equation:

$$M = \text{median}(\text{M}_\text{i}) \tag{6.1}$$

We believe that the memory size requested could represent the subscriber's expectation of potential workload for the VM.

A - Overall Active Rate In our experiment, for a subscriber, its VM_i in time stamp t_j would be consider as active unless its CPU utilization is over 15. Assume at the time stamp i, one subscriber has total NT_i VMs alive, where NA_i VMs are considered as active. Then, the Overall Active Rate for the subscriber would be:

$$A = (\sum_{i=1}^{n}(NA_i/NT_i))/n \tag{6.2}$$

W - Average Active VM amount In feature A, we calculated the overall active rate among all timestamps, but we lost too much detail information during this average procedure. Most important of all, we should have a feature to demonstrate the overall workload requested by the subscriber. Assume at the time stamp i, one subscriber has a total NT_i VMs alive, where NA_i VMs are considered as active. Then, the average active VM amount for the subscriber would be:

$$A = (\sum_{i=1}^{n} NA_i)/n \tag{6.3}$$

Although detailed information are still not considered with features A and W, they provided a draft image of the active rate for each subscriber. We carry out the investigation further and the detailed information about the categorizing of subscribers is described in the section:6.3.2.

I - Median of Average CPU Utilization Rate With features A and W, we obtained a brief idea about the activeness level for a subscriber, but we still want to know if there is any difference even with the same activeness rate. We introduce feature I : median of Average CPU Utilization Rate

in overall period per subscribers.

Assume at the time stamp i, one subscriber has total NT_j VMs alive. For each VM, its CPU utilization is CPU_j. Then, the median of average CPU Utilization Rate in all time period for the subscriber would be:

$$C = \mathrm{median}((\sum_{\mathrm{j}=1}^{\mathrm{NT_j}} \mathrm{CPU_j})/\mathrm{NT_j}) \tag{6.4}$$

The reason we use the median value instead of mean value is that the median value is more robust with outliers.

Above formulation of feature metrics could be adjusted by service provider for their own purposes and it will not affect our proposed general framework.

6.2.3 Chosen of clustering algorithm

In the clustering component, the service provider can choose a specific clustering algorithm per their own need. In our work, we tested several clustering algorithms and chose DBSCAN in the end. There are two parameters in the DBSCAN algorithm, ε and MinPts, where ε is the maximum distance between two neighbors, and MinPts is the minimum number of points in a cluster. However, once MinPts is set, ε can be determined by drawing a k-distance graph (k=MinPts). In other words, ε can be considered as a function of MinPts. MinPts should not be too small, otherwise, noise in the data will result in spurious clusters.

In our experiments, we used DBSCAN initially cluster subscribers into different categories. Then we manually verified every category and partially label them into three major types: Inactive (Normal), Period Active, Extreme Active. The labeled data will be used as the training data of the classification component.

6.2.4 Dynamically Quantify Co-residency Risk Rate

To better quantify the co-resident risk rate, the task was done by combining classification and quantification in our proposed framework. As we discussed, in our previous work, the co-resident risk rate was only calculated based on VM's vulnerability, which is insufficient in the real world.

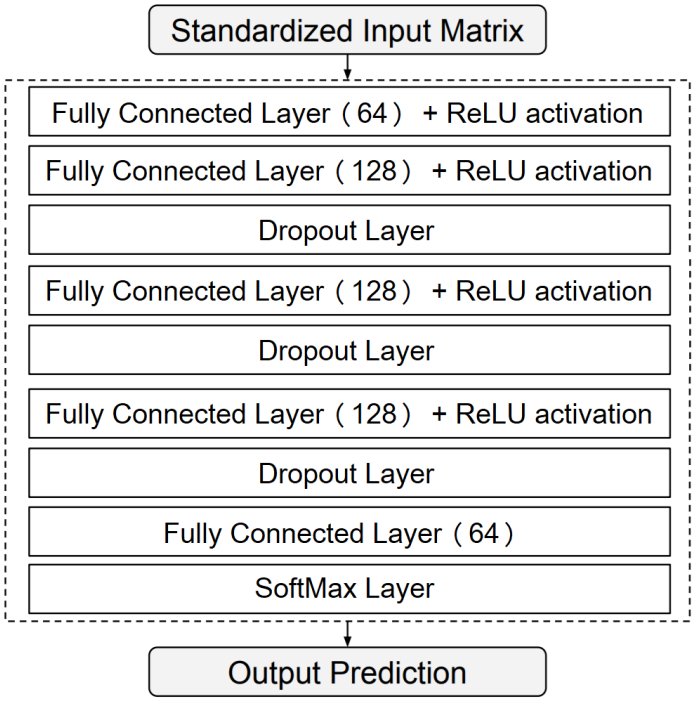

Figure 6.2: Architecture of Proposed Deep Neural Network

After we integrated the latest allocation policy into SMOOP, we believed that the co-resident risk rate of VMs should be mainly determined by their owners. At the same time, even the cloud service provider could collect any information they want and use them to detect malicious activities, the presence of malicious activity is still a very rare case. Major portion information collected should belong to normal service subscriber, which make the task to profile attacker very difficult, if not impossible.

From a different aspect, if the service subscriber can successfully profile normal service subscriber's behavior patterns, the service provider will be able to determine a risk rate by measuring the derivation rate from the normal pattern. In order to do so, the service provider needs to build a non-rule based general system which can dynamically adapt to the environment changes.

Deep Neural Network Architecture

Figure 6.2 shows the Deep Neural Network model used in our experiments. It consists of 9 layers. First, the fully connected layer has 64 nodes (or neurons). Afterward, three fully connected

layers with 128 neurons and Relu activation, followed with the dropout layer. The last layer is a 3-node softmax layer that returns an array of 3 probability scores that sum to 1.

Dropout is a regularization technique that turns neurons on/off in each layer to force them to go through a different path. This operation improves the generalization of the network and prevents over-fitting.To reduce overfitting, we use a dropout [58] layer after the second and third fully connected layer, since its effectiveness is proved in previous work [103] and dropout regularization works really well with fully connected layers. Rectified linear unit (ReLU) is a type of activation function. It is simply defined as f(x) = max(0, x). It turned out that ReLU usually works better in practice than other activation functions, such as sigmod function. Nowadays, it is the most commonly used activation function in the neural network.

The model is trained using back-propagation for Adam Optimizer [67], a stochastic gradient descent that automatically adapts the learning rate. The optimizer works on minimizing the loss function. We use the mean cross-entropy as a loss function in our experiment. The model is also trained with a batch setting, which is not reflected in the layers described above.

Hyper-parameters Tuning

Hyper-Parameters tuning is always a very challenging problem in a Machine Learning task. Normally, the accuracy rate of classification tasks could be dramatically affected by the setting of hyper-parameters. In order to achieve the best classification accuracy, multiple methods were tested, such as random search [19], and grid search was also tested afterward. In the end, we set up the hyper-parameters for training as follows. For the Dropout rate, we set it to 0.5 in each dropout layer. Another important parameter is the learning rate. It determines how fast we move toward the optimal weights in our network. If this parameter is very large, it will skip optimal values. On the other hand, if it is too small, it will take too much time to converge to the optimal values, and it may get stuck in local minima. Fortunately, Adam Optimizer provided a very detailed and flexible API to set up the initial rate and decay rate. In the end, we finish our training by setting the initial learning rate started as 1e - 3 (0.001), beta1 as 0.9, beta2 as 0.999 and decay rate as 0.1.

Handling Imbalanced Data

The imbalance is common in the real world, and most classification data sets do not have an exactly equal number of instances in each class, but a small difference often does not matter. But when there is a modest class imbalance like 4:1 in the dataset or above, it can cause problems [27]. Sometimes imbalance is not just common, it is expected. For example, in datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the Not-Fraud class and a very small minority will be in the Fraud class. We met the same situation in our own dataset, most of the subscribers should be legal and only a very rare event could be considered caused by the malicious attacker.

There are several methods available for dealing with the imbalance situation:

Change Performance Metrics Since we are dealing with an extremely imbalanced dataset, the accuracy rate is not enough to accurately evaluate the performance model. F Measuring Matrix was applied in our experiment. It contained three major metrics as below [27]:

- Precision: the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier's exactness. Low precision indicates a high number of false positives.

- Recall: the number of true positives divided by the number of positive values in the test data. Recall is also called Sensitivity or the True Positive Rate. It is a measure of a classifier's completeness. Low recall indicates a high number of false negatives.

- F1: Score: the weighted average of precision and recall

The analysis of experimental results is in our follow evaluation section.

Oversample Minority Class Oversampling can be defined as adding more copies of the minority class. In our experiment, we implemented oversampling by constructing the batch with a solid number of period and extreme active types, which would be randomly chosen each time.

Undersample Majority Class Undersampling can be defined as removing some observations of the majority class. A drawback is that we are removing information that may be valuable. This could lead to under-fitting and poor generalization to the test set. In our experiment, we also implemented under-sampling by constructing the batch with a solid number of Inactive type, which would be randomly chosen each time.

6.2.5 Quantified the Co-Residency Risk

In our framework design, the quantification component will take the output from the classification component and apply it into a pre-defined quantify function. In our experiment, for simplicity, we used a softmax activation function in the classification component and directly output the prediction of the probability of each category. The probability rate of the normal category will be used to calculate the co-resident risk. In other words, we believe it represents the derivation rate from normal behavior patterns.

6.3 Experimental Result and Evaluation

We conducted our experiments on a Dell Precision Tower T5810 Workstation. The Dell Workstation has an Intel Xeon E5-1620, 32G RAM, and Nvidia Quadro P5000 Graphic Card for GPU acceleration. GPU acceleration was applied in our Model Training procedure and it shortens the training time dramatically.

6.3.1 Insight of Azure Dataset

Unlike existing works, real-world large-scale dataset was used in our research. Azure Public Dataset [31] provided two portion of the Virtual Machine (VM) workload of Microsoft Azure collected in 2019 and 2017. This dataset contains over 12,000 service subscribers with their over 5 million VMs and corresponding 3.1 billion CPU utilization record sampled every five minutes over one month. The total size of the dataset is over 500GBs. To finish the statistical task for the above feature metrics, we had divided the task into multiple portions of data in each temporal steps.

After we applied our proposed feature metrics with Azure Dataset, several observation could be reviewed as follows:

A - Overall Active Rate From Figure 6.3, we noticed a fact that the CPU utilization rate of more than 90% VMs is under 15%, which means most of VMs stayed under very low workload status.

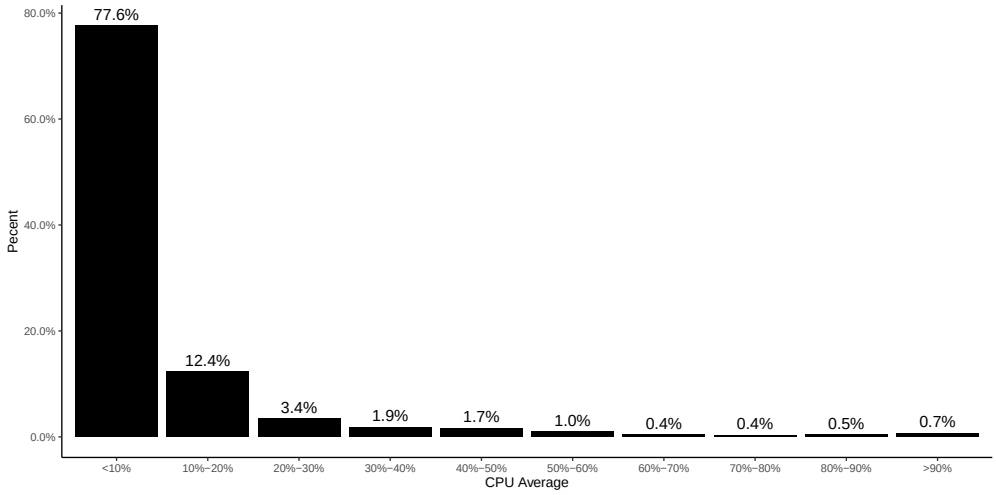

Figure 6.3: Average CPU Utilization distribution among VMs

After we calculated out the Overall Active Rate for all subscribers, the cumulative distribution diagram would be checked in Figure 6.4. Generally, speaking, over 80% subscribers have less than 10% active rate.

W - Average Active VM amount After we calculated the Average Active VM amount for all subscriber, the cumulative distribution diagram would be check in Figure 6.5. Combined with feature A, we could filter out the type of extreme active subscriber.

I - Median of Average CPU Utilization Rate We checked the cumulative distribution diagram in Figure 6.6 and noticed that over 90% subscribers' is the inactive type in Figre 6.8

With all above feature metrics, we obtained a brief idea about service subscribers. In the next section, we will use our proposed feature metrics to finish the task of clustering subscribers.

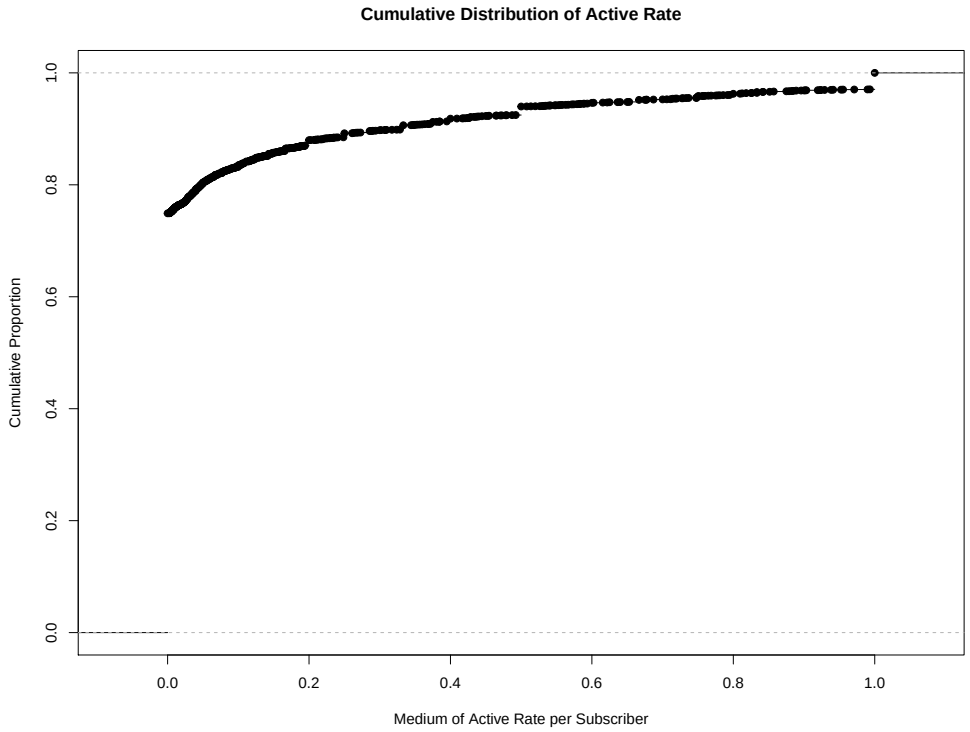

Figure 6.4: Cumulative Distribution of Active Rate per Subscriber

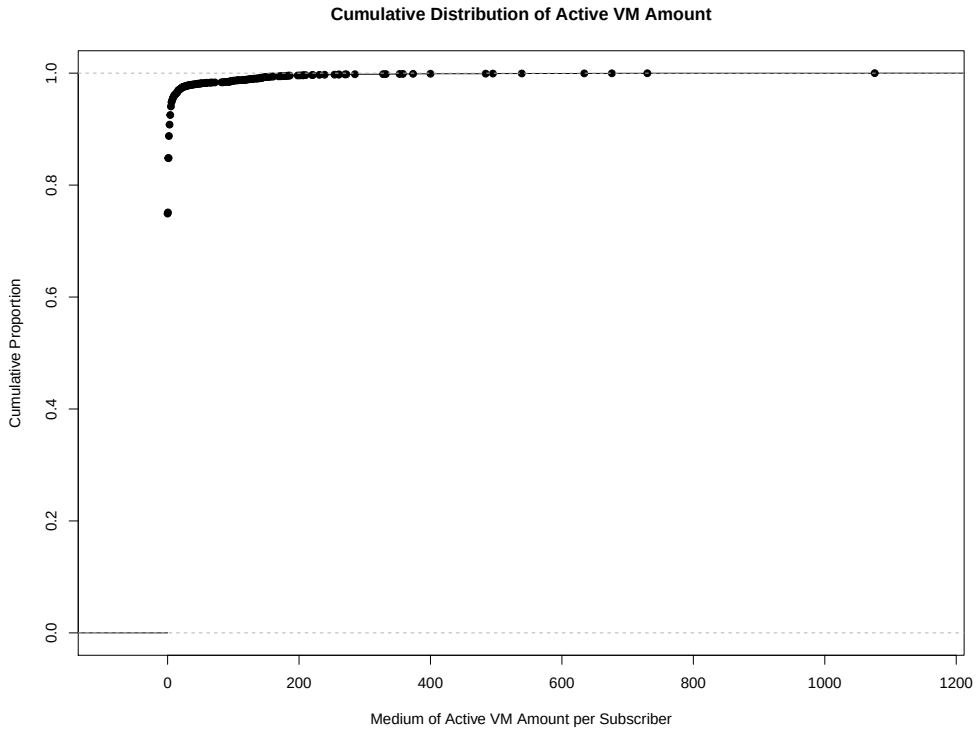

Figure 6.5: Cumulative Distribution of Average Active VM Amount per Subscriber

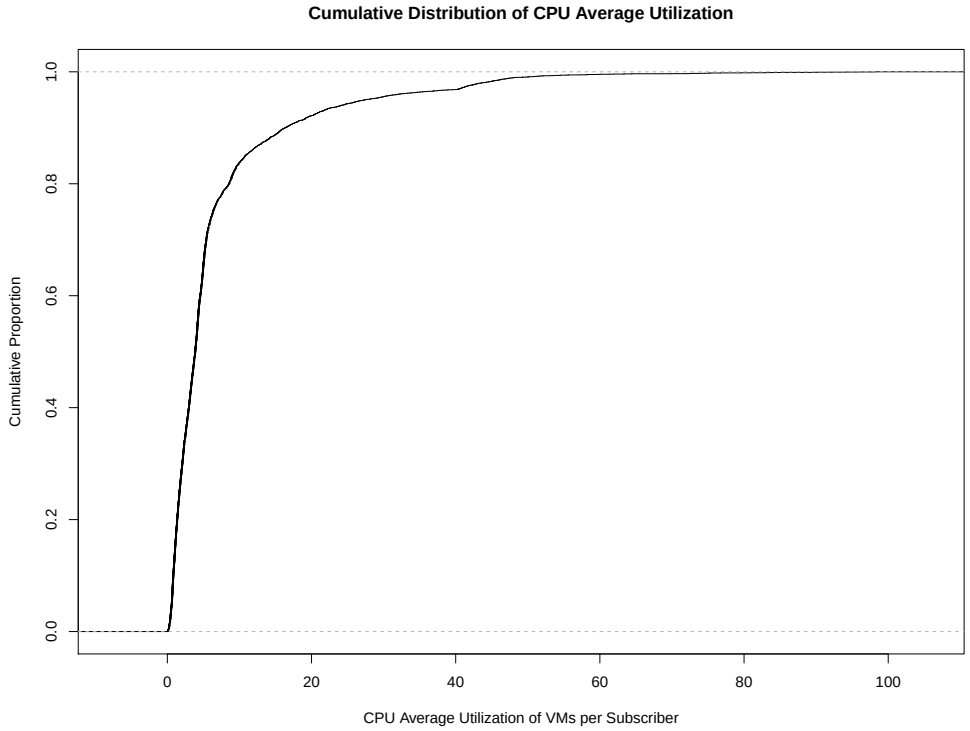

Figure 6.6: Cumulative Distribution of CPU Average Utilization per Subscriber

6.3.2 Clustering of Subscribers

Since Azure Dataset contains over 12,000 service subscribers, we want to cluster them into several candidate groups for future use.

Aspect from Activeness Rate

First of all, based on observation from Figure 6.7, we noticed that around 65% subscribers only created one VM. Another 25% subscribers created less than five VMs in their lifetime.

Combined with the cumulative distribution of feature I, we could guess that most service subscribers should only maintain a small amount VMs at the inactive level.

To verify our guess, we finished the active rate curve in all timestamps for all subscribers. Based on the observation with all curve diagram, we defined several categories of subscribers based on the activeness rate only:

- Inactive Subscriber: In Figure 6.8, the very top section is a typical curve for the inactive type

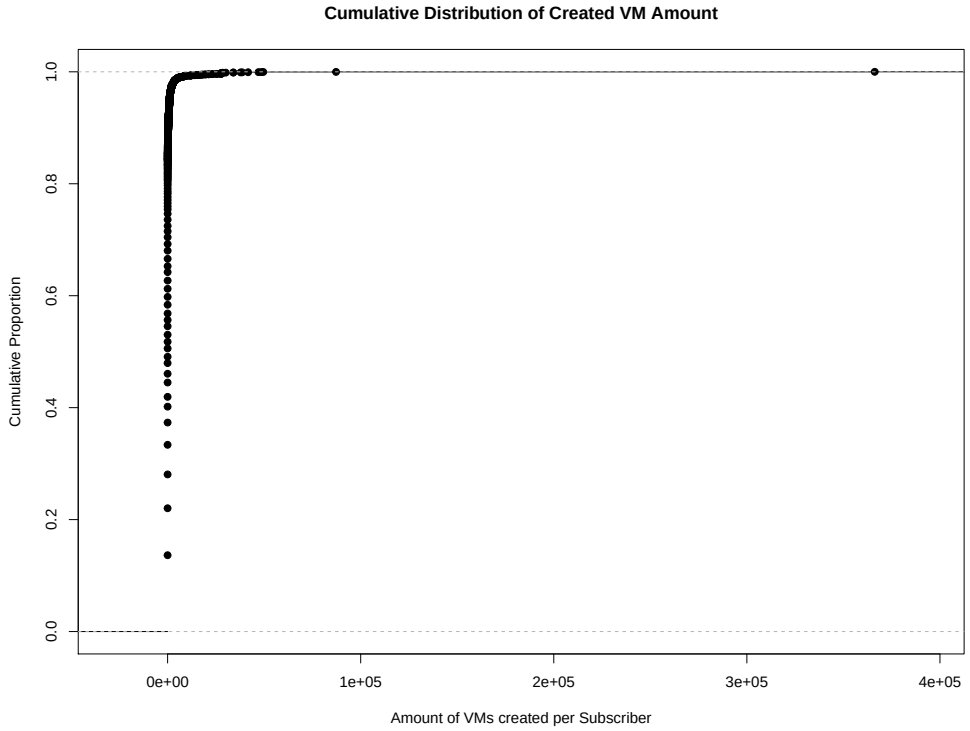

Figure 6.7: Cumulative Distribution of Created VMs Amount per Subscriber

subscriber we defined, and over 80% subscriber falls into this category. Within this category, a very small amount of VMs was created by the subscriber and most of the time, the VM stay inactive.

- Period active subscriber: The middle portion of Figure 6.8 is a typical curve for the Period active subscriber we defined. Multiple VMs created by the subscriber and the activeness level is shifted dramatically over the subscriber's whole lifetime.

- Extreme active subscriber: The typical curve for this type of subscriber is located at the very bottom of Figure 6.8. Generally speaking, this type of subscriber maintained an extreme high activeness rate over their whole lifetime.

In our experiments, DBSCAN only performed as an initial tool to cluster subscribers. During the experimental procedure, we tested the MinPts ranging from 3 to 20. Since we believed over 99% data should come from normal service subscriber, we expected that there would have

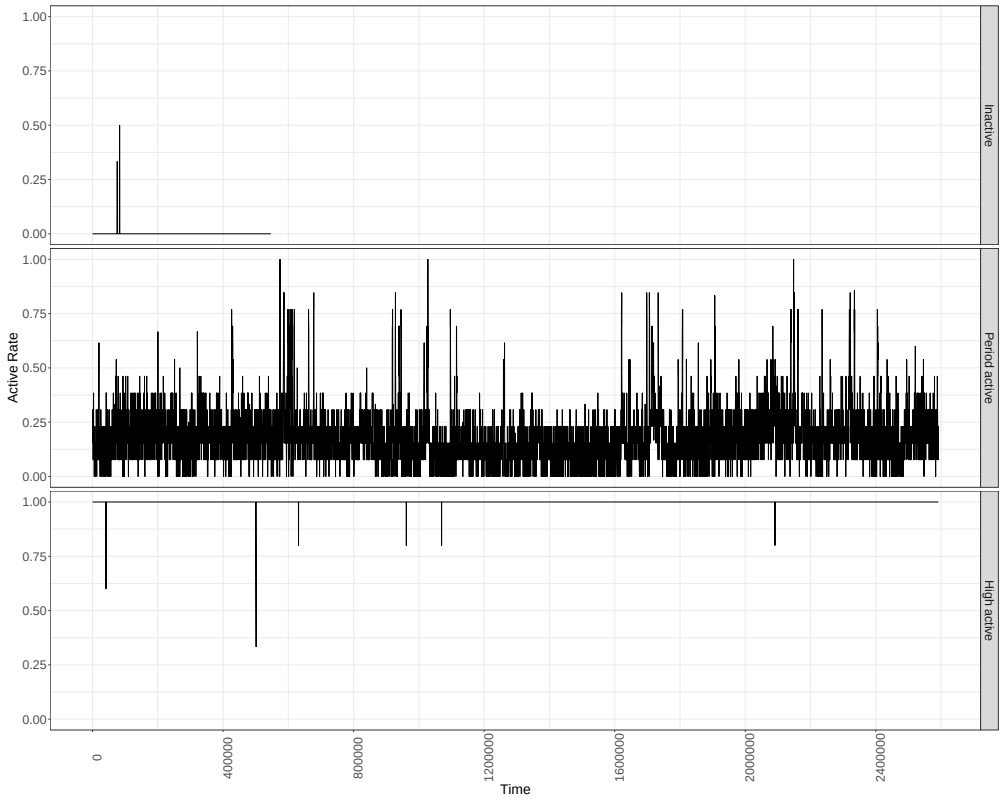

Figure 6.8: Subscriber categorized with activeness level

one clustering output should contain a major portion of the subscribers. In the end, we set the MinPts to 5 and ε to 1.5. After the initial clustering, we manually verify all user's activeness level curves among the whole lifetime. Three major categories were separated as 6.8 the point here is if the attacker is simulated as a normal user, the cost will be high. In our future work, The detail curve level diagram will be used in our Convolutional Neural Network sub-module and we could avoid the problem by losing too much detail information by over abstract. We demonstrated the coefficient relationship between extreme active subscribers and others in Figure6.9.

We demonstrated our clustering result in Figure6.10 and Figure6.11. Most of the service subscribers were clustered into one category, which is quite fit for our expectations. Most importantly, a total of 78 subscribers were discovered as the outlier. By checking with their detail active rate curve, we believe their behavior pattern would be the closest to potential high-risk ones.

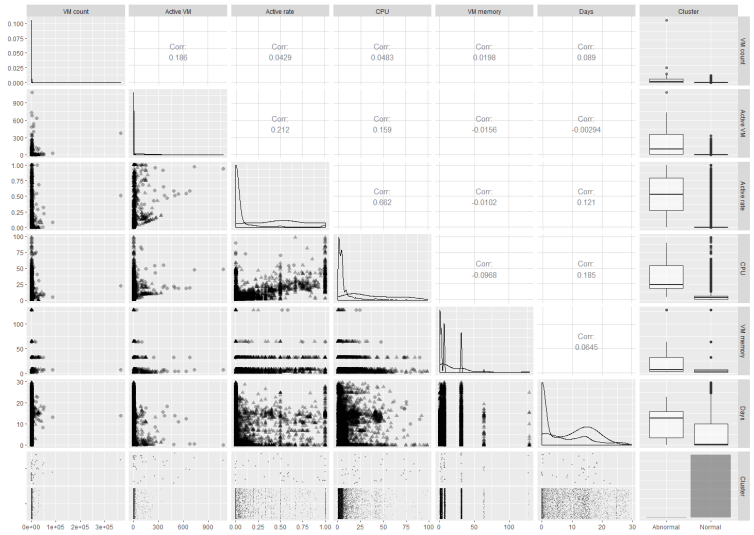

Figure 6.9: Cluster Pairing

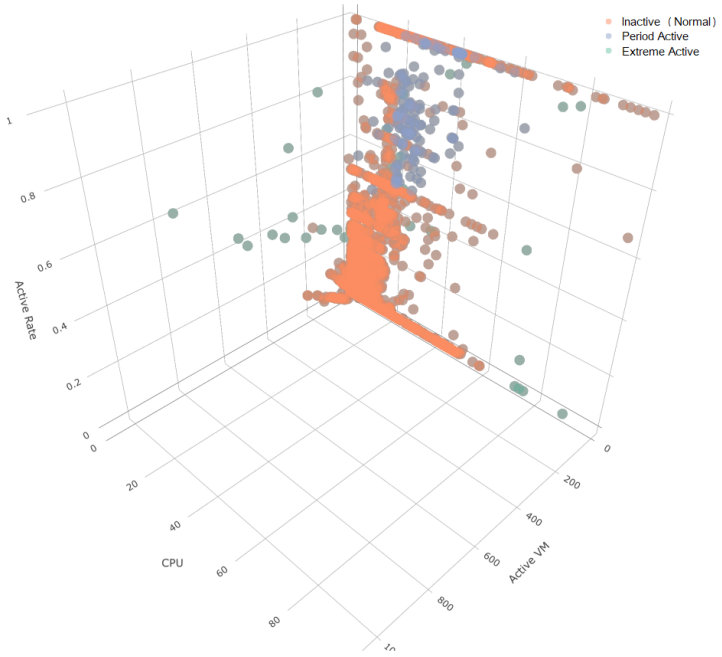

Figure 6.10: Cluster Result

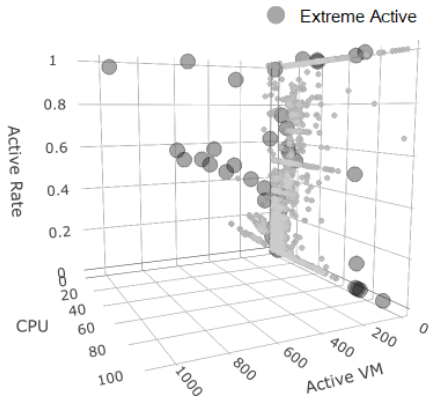

Figure 6.11: Cluster Result (Extreme Active)

6.3.3 Training and Evaluating Deep Neural Network

After we initially clustered service subscribers, we manually labeled them into three major categories: normal subscriber, period active subscriber, and extreme active subscriber, by verifying their own detail active rate curves. As a result, we labeled all subscribers as follow 6.1:

Table 6.1: Subscribers in each Category

Type	Amount
Normal	11573
Period Active	379
Extreme Active	78

Oversampling Minority Class Considering only a total of 78 subscribers are located into extreme active category, and only 379 subscribers are located into period active category. Comparing with a total of 11573 normal subscriber, we are facing an extreme imbalanced dataset. First, we divided the whole dataset into two parts: Training set (90%) and Test set (10%). For each training session of our model, we finished 100 batches in each epoch and we finished a total of 100 epochs. To handle this imbalance, we have to oversample these two minor classes. We randomly choose 200 from the normal type, 100 from period active type and 40 from extreme active type and shuffle them to form a batch of the training set, and this batch of training set will be used in one step. Since all training tasks were finished off-line, the inference procedure for new subscriber data only need few seconds to generate the final prediction.

We reserved 10% of total service subscribers' data for test purposes and they have never be used to optimize our model in the training procedure. We only recorded the checkpoint when the accuracy rate of the test dataset is improved. Thus we could observe the loss value was shifted several times in our experiment in Figure 6.12. Also, the accuracy rate of the test dataset could be reviewed in Figure 6.13.

From Figure 6.13, we could observe the increase of accuracy rate. Even we tried to use dropout to avoid model overfit problem, but we still could observe that when training accuracy keeps rising from 94% to 96%, the accuracy rate for test set actually declined from 97.7% to 95%. Considering

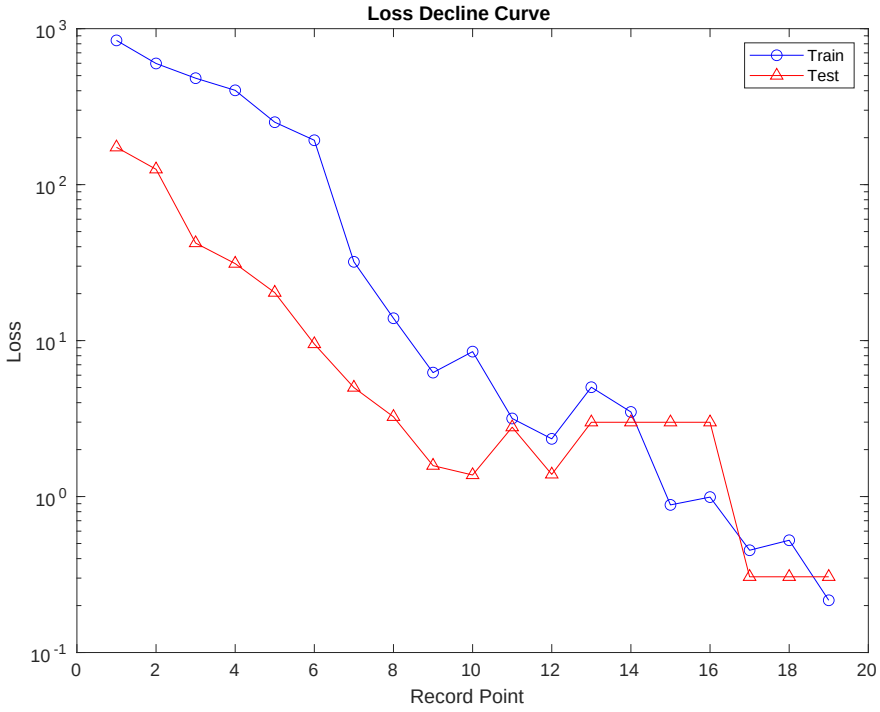

Figure 6.12: Cross-Entropy Loss Curve

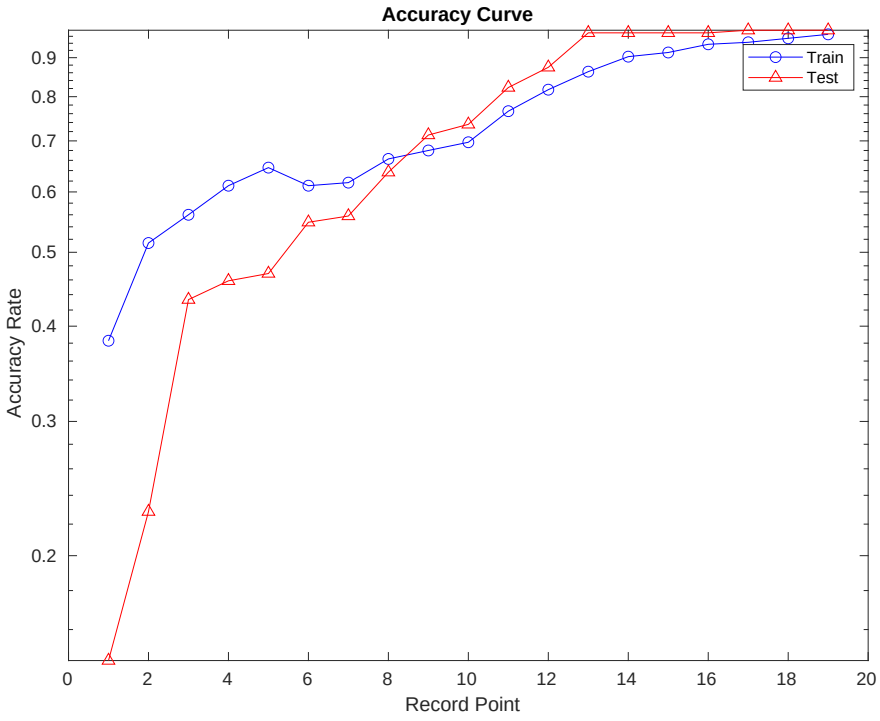

Figure 6.13: Training Accuracy Curve

Table 6.2: Best F1 score result

Category	Recall	Precision	F1 Score
Normal	0.9907	0.9877	0.9892
Period Active	0.6667	0.6429	0.6207
Extreme Active	0.25	0.5	0.3333

an extreme imbalanced dataset was handled here, we need to use several other performance metrics to better evaluate our model.

After we achieved the best accuracy rate of 98% for test dataset, we calculated out all the recall, precision and F1 score for all three categories as in Table6.2:

In the test dataset, it only contains a total of eight extreme active subscribers, which was used to simulate the practical environment. The reason is, in the real world, the appearance of an extremely active pattern would be a very rare event. This extreme imbalance also existed in the training dataset. Table 6.2 demonstrated the best F1-score result for test dataset with total 98% accuracy rate for test.

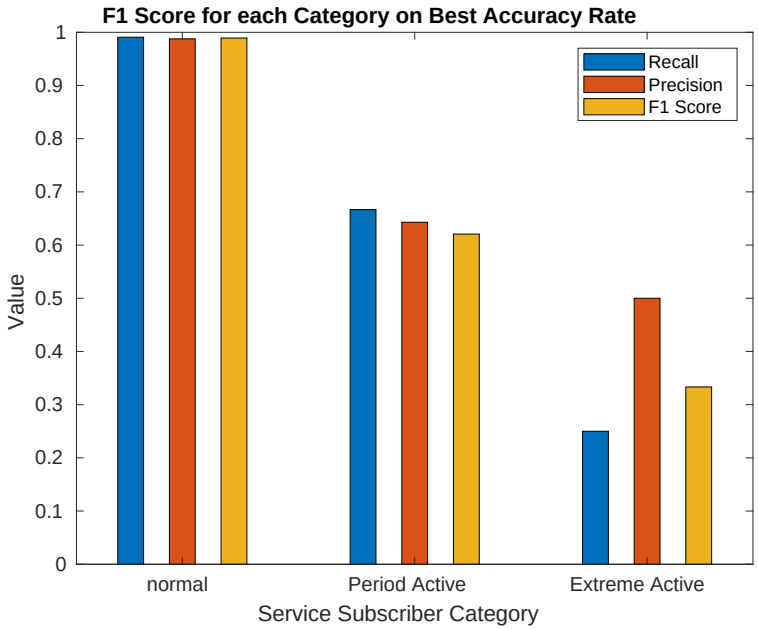

Figure 6.14: F Measure Matrix

By reviewing the F measuring matrix from Figure 6.14, the performance of our current classification component on extreme active pattern was affected by the extreme data imbalance in our experiment. There are two possible explanations for this issue. First, even we applied multiple

methods to handle the imbalanced training dataset, such as oversampling the minority category, the training of our model was still under noticeable affection of data imbalance. In the meanwhile, this is the major defect of the neural network, it got its limitation to handle unseen data. Another possible reason is the initial clustering is not accurate enough. Since there are only 78 service subscribers were labeled into the extreme active category, there is a possibility that too much detail feature information was lost during the abstraction procedure of calculating feature metrics. To fix this issue, we proposed to construct a Convolutional Neural Network sub-module for handling the detail active rate curve per each subscriber. In that case, this manual label procedure would be replaced by an automatic component, and a lot of improvement could be achieved in our future work. Also on the other hand, as we discussed above, this issue is expected. Since we only used the probability rate of inactive type in the reference procedure to calculate the risk rate, which will not be affected by this issue.

CHAPTER 7: CONCLUSION AND FUTURE WORK

While cloud computing is providing basis of services in our daily life, it continues to evolve and offers new concepts and capabilities. Meanwhile, new attacks will target on the new features and take advantage of them. In order to keep up with the cycle of threats and mitigation, we need to leverage these new capabilities for securing the cloud. In our dissertation, we proposed a comprehensive framework to better secure the whole cloud and applied machine learning (ML) architecture to better quantify risk rate in the cloud. We mainly focused on the Infrastructure as a Service (IaaS) layer type of cloud services.

7.1 Summary of Contributions

In our dissertation, we have developed a comprehensive approach for security assessment of virtual machine placement, and presented an approach to quantify the security risks of the cloud based on the vulnerabilities caused by various factors including networks, physical machines, VMs, and co-residence of VMs.

In chapters 3 and 4, we proposed our security metrics and detail explanations were provided. Based on our security metrics, the security risks of a specific VM placement could be comprehensively measured. After the security risk rate is measured, we have a new scheme to generate VMP based on multiple objective optimization with the given resources and other constraints. Our proposed method searches for an improved placement on a specific target, balancing on multiple objectives. The experimental results demonstrated the effectiveness and improvement of our approach compared with existing solutions. As we discussed and investigated, the priority of strategies can be adjusted in real-time and new strategy can be added, in response to the runtime environment. We proposed our SMOOP algorithm to integrate our solutions. We demonstrated how to consider incremental placement requirements and modified our SMOOP to be adapted to specific VM deployment situations. The latest VM allocation policies were also integrated into our framework to better defend the co-resident attacks. Thus, our system can keep evolving to fit

real-world situations.

One major defect of our earlier work was that our experiments were based on simulation data. After we obtained Microsoft Azure VM tracing dataset, we extended our work based on the large scale and real-world datasets. In chapter 6, we proposed a dynamic adaptive framework to better quantify the co-resident risks. A set of feature metrics was proposed to accurately profile the behavior pattern of normal service subscribers in the cloud. Based on the analysis of Azure Dataset, we used DBSCAN algorithm to initially cluster into several groups of service subscribers and manually labeled them into three major categories: Inactive, Period Active, and Extreme Active. With the labeled datasets, a classification component was trained to identify new datasets and the output was used in the quantification component to quantify the co-residency risk rate. Based on our experimental results, our classification component demonstrated robustness to new data. It also achieved an accuracy of 98% for test dataset and its performance was verified by examination of F Measuring Matrix.

7.2 Future Work

In our future work, we will continue working on our system and extend it in two directions. First, as we mentioned above, too much detailed information about the service subscriber was lost during the data abstract procedure. Also, we observed that extreme active pattern is a very rare event in all datasets, and this extreme imbalance highly affects the classification task. By analysis of the F measuring matrix, there is still a lot of space for improvements to more accurately identify those rare events. To conquer the challenge, we propose to build another sub-module in the classification component to handle the detail feature diagram directly. In this way, we can more accurately profile normal subscriber's behavior patterns. Benefited from the improvement, we will be able to achieve a better classification at the same time.

Second, our framework was trained and tested in one particular dataset. The general compatibility of our proposed framework may be questioned by a new dataset. With the newly collected dataset, a well-trained classification component should be able to adapt to a new working en-

vironment. With appropriate transfer training, We could extend applying our framework to test the general adaption capability. There is also another possible research direction, if we can successfully profile normal subscriber pattern, it can be used to predict specific subscriber's future workload request or detection of abnormal appearance.

BIBLIOGRAPHY

[1] Cloud computing https://en.wikipedia.org/wiki/cloud_computing.

[2] Cloudsim http://www.cloudbus.org/cloudsim/.

[3] Webopedia: Online tech dictionary for students, educators and it professionals.

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, 2016. USENIX Association.

[5] Zaina Afoulki, Aline Bousquet, and Jonathan Rouzaud-Cornabas. A security-aware scheduler for virtual machines on iaas clouds. *Report 2011*, 2011.

[6] S. Al-Haj, E. Al-Shaer, and H. V. Ramasamy. Security-aware resource allocation in clouds. In *2013 IEEE International Conference on Services Computing*, pages 400–407, June 2013.

[7] M. Alicherry and T. V. Lakshman. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *2013 Proceedings IEEE INFOCOM*, pages 647–655, April 2013.

[8] Amazon. Amazon web services.

[9] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 217–224, New York, NY, USA, 2002. ACM.

[10] George Amvrosiadis, Jun Woo Park, Gregory R. Ganger, Garth A. Gibson, Elisabeth Baseman, and Nathan DeBardeleben. On the diversity of cluster workloads and its impact on research results. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '18, pages 533–546, Berkeley, CA, USA, 2018. USENIX Association.

[11] M. Aslam, C. Gehrmann, and M. BjÃ¶rkman. Security and trust preserving vm migrations in public clouds. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 869–876, June 2012.

[12] Amittai Aviram, Sen Hu, Bryan Ford, and Ramakrishna Gummadi. Determinating timing channels in compute clouds. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 103–108, New York, NY, USA, 2010. ACM.

[13] Amittai Aviram, Sen Hu, Bryan Ford, and Ramakrishna Gummadi. Determinating timing channels in compute clouds. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 103–108, New York, NY, USA, 2010. ACM.

[14] Yossi Azar, Seny Kamara, Ishai Menache, Mariana Raykova, and Bruce Shepard. Co-location-resistant clouds. In *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security*, CCSW '14, pages 9–20, New York, NY, USA, 2014. ACM.

[15] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[16] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[17] Salman A. Baset, Long Wang, and Chunqiang Tang. Towards an understanding of oversubscription in cloud. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in*

Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE'12, pages 7–7, Berkeley, CA, USA, 2012. USENIX Association.

[18] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *J. Mach. Learn. Res.*, 2:125–137, March 2002.

[19] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[20] P. Berkhin. *A Survey of Clustering Data Mining Techniques*, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[21] Khalid Bijon, Ram Krishnan, and Ravi Sandhu. Mitigating multi-tenancy risks in iaas cloud through constraints-driven virtual resource scheduling. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, SACMAT '15, pages 63–74, New York, NY, USA, 2015. ACM.

[22] Khalid Bijon, Ram Krishnan, and Ravi Sandhu. Virtual resource orchestration constraints in cloud infrastructure as a service. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 183–194, New York, NY, USA, 2015. ACM.

[23] Khalid Zaman Bijon, Ram Krishnan, and Ravi Sandhu. A formal model for isolation management in cloud infrastructure-as-a-service. In Man Ho Au, Barbara Carminati, and C.-C. Jay Kuo, editors, *Network and System Security*, pages 41–53, Cham, 2014. Springer International Publishing.

[24] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, and D. H. Lorenz. Guaranteeing high availability goals for virtual machine placement. In *2011 31st International Conference on Distributed Computing Systems*, pages 700–709, June 2011.

[25] Stefano Bistarelli, Simon N. Foley, and Barry O'Sullivan. Detecting and eliminating the cascade vulnerability problem from multilevel security networks using soft constraints. In

Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence, IAAI'04, pages 808–813. AAAI Press, 2004.

[26] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[27] Tara Boyle. Dealing with imbalanced data.

[28] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, CÃ©sar A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[29] Ignacio Cano, Lequn Chen, Pedro Fonseca, Tianqi Chen, Chern Cheah, Karan Gupta, Ramesh Chandra, and Arvind Krishnamurthy. ADARES: adaptive resource management for virtual machines. *CoRR*, abs/1812.01837, 2018.

[30] E. Caron, A. D. Le, A. Lefray, and C. Toinard. Definition of security metrics for the cloud computing and security-aware virtual machine placement algorithms. In *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 125–131, Oct 2013.

[31] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 153–167, New York, NY, USA, 2017. ACM.

[32] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, May 2002.

[33] CVE-2007-4993. Xen guest root can escape to domain 0 through pygrub.

[34] CVE-2007-5497. Vulnerability in xenserver could result in privilege escalation and arbitrary code execution.

[35] Foad Dabiri, Navid Amini, Mahsan Rofouei, and Majid Sarrafzadeh. Reliability-aware optimization for dvs-enabled real-time embedded systems. In *Proc. of the 9th int'l symposium on Quality Electronic Design*, pages 780–783, 2008.

[36] Kamal Dahbur, Bassil Mohammad, and Ahmad Bisher Tarakji. A survey of risks, threats and vulnerabilities in cloud computing. In *ISWSA)*, pages 40–48, 2011.

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–198, Apr 2002.

[38] MediaWiki default. Kernel based virtual machine.

[39] D. S. Dias and L. H. M. K. Costa. Online traffic-aware virtual machine placement in data center networks. In *2012 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–8, Dec 2012.

[40] György Dósa and Jiri Sgall. First Fit bin packing: A tight analysis. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 538–549, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[41] Kai-Bo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In Nikunj C. Oza, Robi Polikar, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 278–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[42] Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors. *Optimal Analysis of Best Fit Bin Packing*, pages 429–441. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[43] Ian T. Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. *CoRR*, abs/0901.0131, 2009.

[44] Rohith Gandhi. Support vector machine - introduction to machine learning algorithms.

[45] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[46] Ali Gholami and Erwin Laure. Security and privacy of sensitive data in cloud computing: A survey of recent developments. *CoRR*, abs/1601.01498, 2016.

[47] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods. Cloud-trust-a security assessment model for infrastructure as a service (iaas) clouds. *IEEE Transactions on Cloud Computing*, 5(3):523–536, July 2017.

[48] B. Grobauer, T. Walloschek, and E. Stocker. Understanding cloud computing vulnerabilities. *IEEE Security Privacy*, 9(2):50–57, March 2011.

[49] N. Gruschka and M. Jensen. Attack surfaces: A taxonomy for attacks on cloud services. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 276–279, July 2010.

[50] Anonymous Hacker. Xbox 360 hypervisor privilege escalation vulnerability.

[51] Jin Han, Wangyu Zang, Li Liu, Songqing Chen, and Meng Yu. Risk-aware multi-objective optimized virtual machine placement in the cloud. *Journal of Computer Security*, 26(5):707–730, Aug 2018.

[52] Jin Han, Wanyu Zang, Songqing Chen, and Meng Yu. Reducing security risks of clouds through virtual machine placement. In Giovanni Livraga and Sencun Zhu, editors, *Data*

and Applications Security and Privacy XXXI, pages 275–292, Cham, 2017. Springer International Publishing.

[53] Jin Han, Wanyu Zang, Songqing Chen, and Meng Yu. Reducing security risks of clouds through virtual machine placement. In Giovanni Livraga and Sencun Zhu, editors, *Data and Applications Security and Privacy XXXI*, pages 275–292, Cham, 2017. Springer International Publishing.

[54] Y. Han, T. Alpcan, J. Chan, C. Leckie, and B. I. P. Rubinstein. A game theoretical approach to defend against co-resident attacks in cloud computing: Preventing co-residence using semi-supervised learning. *IEEE Transactions on Information Forensics and Security*, 11(3):556–570, March 2016.

[55] Y. Han, J. Chan, T. Alpcan, and C. Leckie. Virtual machine allocation policies against co-resident attacks in cloud computing. In *2014 IEEE International Conference on Communications (ICC)*, pages 786–792, June 2014.

[56] Y. Han, J. Chan, T. Alpcan, and C. Leckie. Using virtual machine allocation policies to defend against co-resident attacks in cloud computing. *IEEE Transactions on Dependable and Secure Computing*, 14(1):95–108, Jan 2017.

[57] M. M. Hasan and M. A. Rahman. Protection by detection: A signaling game approach to mitigate co-resident attacks in cloud. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 552–559, June 2017.

[58] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[59] H. Hlavacs, T. Treutner, J. Gelas, L. Lefevre, and A. Orgerie. Energy consumption side-channel attack at virtual machines in a cloud. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 605–612, Dec 2011.

[60] Joseph Douglas Horton, RH Cooper, WF Hyslop, Bradford G. Nickerson, OK Ward, Robert Harland, Elton Ashby, and WM Stewart. The cascade vulnerability problem. *Journal of Computer Security*, 2(4):279–290, 1993.

[61] R. Jansen and P. R. Brenner. Energy efficient virtual machine allocation in the cloud. In *2011 International Green Computing Conference and Workshops*, pages 1–8, July 2011.

[62] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono. On technical security issues in cloud computing. In *2009 IEEE International Conference on Cloud Computing*, pages 109–116, Sep. 2009.

[63] R. Jhawar, V. Piuri, and P. Samarati. Supporting security requirements for resource management in cloud computing. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 170–177, Dec 2012.

[64] S. Jin, J. Ahn, S. Cha, and J. Huh. Architectural support for secure virtualization under a vulnerable hypervisor. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 272–283, Dec 2011.

[65] Ian A. Kash, Greg O'Shea, and Stavros Volos. Dc-drf: Adaptive multi-resource sharing at public cloud scale. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 374–385, New York, NY, USA, 2018. ACM.

[66] Taesoo Kim, Marcus Peinado, and Gloria Mainar-Ruiz. STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 189–204, Bellevue, WA, 2012. USENIX.

[67] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2014.

[68] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep. 1990.

[69] Richard E. Korf. A new algorithm for optimal bin packing. In *Eighteenth National Conference on Artificial Intelligence*, pages 731–736, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[70] Min Li, Yulong Zhang, Kun Bai, Wanyu Zang, Meng Yu, and Xubin He. Improving cloud survivability through dependency based virtual machine placement., 2012.

[71] lMohammad Masdari, Sayyid Shahab Nabavi, and Vafa Ahmadi. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66:106–127, May, 2016.

[72] S. Louhichi, M. Gzara, and H. Ben Abdallah. A density based algorithm for discovering clusters with varied density. In *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pages 1–6, Jan 2014.

[73] Zoltán Ádám Mann. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *ACM Comput. Surv.*, 48(1):11:1–11:34, August 2015.

[74] H. Maziku and S. Shetty. Network aware vm migration in cloud data centers. In *2014 Third GENI Research and Educational Experiment Workshop*, pages 25–28, March 2014.

[75] R. Melhem, D. Mossé, and E. (Mootaz) Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Trans. on Computers*, 53(2):217–231, 2004.

[76] Muhammad Baqer Mollah, Md. Abul Kalam Azad, and Athanasios Vasilakos. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, 84(Supplement C):38 – 54, 2017.

[77] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 2, pages 1702–1707 vol.2, May 2002.

[78] et al. Peter Mell, Tim Grance. The nist definition of cloud computing. 2011.

[79] D T Pham, S S Dimov, and C D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[80] Dung H. Phan, Junichi Suzuki, Raymond Carroll, Sasitharan Balasubramaniam, William Donnelly, and Dmitri Botvich. Evolutionary multiobjective optimization for green clouds. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 19–26, New York, NY, USA, 2012. ACM.

[81] D. K. Pradhan. *Fault Tolerance Computing: Theory and Techniques*. Prentice Hall, 1986.

[82] S. Rethishkumar and R. Vijayakumar. Two-player security game approach based co-resident dos attack defence mechanism for cloud computing. In *2017 IJSRCSEIT International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, volume 2, pages 552–559, June 2017.

[83] S. Rethishkumar and R. Vijayakumar. Defender vs attacker security game model for an optimal solution to co-resident dos attack in cloud. In S. Balaji, Álvaro Rocha, and Yi-Nan Chung, editors, *Intelligent Communication Technologies and Virtual Mobile Networks*, pages 527–537, Cham, 2020. Springer International Publishing.

[84] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[85] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[86] J Rutkowska and R Wojtczuk. Xen owning trilogy. *Talk at Black Hat*, 2008.

[87] C. Servin, M. Ceberio, E. Freudenthal, and S. Bistarelli. An optimization approach using soft constraints for the cascade vulnerability problem. In *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*, pages 372–377, June 2007.

[88] Sachin Shetty, Xuebiao Yuchi, and Min Song. Security-aware virtual machine placement in cloud data center. In *Moving Target Defense for Distributed Systems*, pages 13–24. Springer, 2016.

[89] J. Shi, X. Song, H. Chen, and B. Zang. Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 194–199, June 2011.

[90] R. Shi, Y. Gan, and Y. Wang. Evaluating scalability bottlenecks by workload extrapolation. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 333–347, Sep. 2018.

[91] Soichi Shigeta, Hiroyuki Yamashima, Tsunehisa Doi, Tsutomu Kawai, and Keisuke Fukui. Design and implementation of a multi-objective optimization mechanism for virtual machine placement in cloud computing data center. In *International Conference on Cloud Computing*, pages 21–31. Springer, 2012.

[92] Vikas Sindhwani and S. Sathiya Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 477–484, New York, NY, USA, 2006. ACM.

[93] Pratiksha Doshi Sumit Siddharth. Five common application vulnerabilities.

[94] Smitha Sundareswaran and Anna C. Squcciarini. Detecting malicious co-resident virtual machines indulging in load-based attacks. In Sihan Qing, Jianying Zhou, and Dongmei Liu,

editors, *Information and Communications Security*, pages 113–124, Cham, 2013. Springer International Publishing.

[95] A. H. Sung and S. Mukkamala. Identifying important features for intrusion detection using support vector machines and neural networks. In *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, pages 209–216, Jan 2003.

[96] Jakub Szefer, Eric Keller, Ruby B. Lee, and Jennifer Rexford. Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 401–412, New York, NY, USA, 2011. ACM.

[97] Lisa Tagliaferri. An introduction to machine learning.

[98] H. Takabi, J. B. D. Joshi, and G. Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security Privacy*, 8(6):24–31, Nov 2010.

[99] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. In *ACM SIGCOMM Computer Communication Review 39)*, pages 50–55, 2008.

[100] Venkatanathan Varadarajan, Thomas Ristenpart, and Michael Swift. Scheduler-based defenses against cross-vm side-channels. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 687–702, San Diego, CA, August 2014. USENIX Association.

[101] Venkatanathan Varadarajan, Yinqian Zhang, Thomas Ristenpart, and Michael M Swift. A placement vulnerability study in multi-tenant public clouds. In *USENIX Security*, pages 913–928, 2015.

[102] Bhanu C. Vattikonda, Sambit Das, and Hovav Shacham. Eliminating fine grained timers in xen. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 41–46, New York, NY, USA, 2011. ACM.

[103] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[104] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, Nov. 2005.

[105] Wikipedia. An introduction to artificial neural network https://en.wikipedia.org/wiki/artificial_neural_network.

[106] H. Wu and W. Wang. A game theory based collaborative security detection method for internet of things systems. *IEEE Transactions on Information Forensics and Security*, 13(6):1432–1445, June 2018.

[107] J. Wu, L. Ding, Y. Lin, N. Min-Allah, and Y. Wang. Xenpump: A new method to mitigate timing channel in cloud computing. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 678–685, June 2012.

[108] J. Xu and J. A. B. Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 179–188, Dec 2010.

[109] Y. Xu, W. Cui, and M. Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy*, pages 640–656, May 2015.

[110] Zhang Xu, Haining Wang, and Zhenyu Wu. A measurement study on co-residence threat inside the cloud. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 929–944, Washington, D.C., August 2015. USENIX Association.

[111] S. Yu, X. Gui, and J. Lin. An approach with two-stage mode to detect cache-based side channel attacks. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 186–191, Jan 2013.

[112] S. Yu, X. Gui, F. Tian, P. Yang, and J. Zhao. A security-awareness virtual machine placement scheme in the cloud. In *2013 IEEE 10th International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pages 1078–1083, Nov 2013.

[113] Xuebiao Yuchi and S. Shetty. Enabling security-aware virtual machine placement in iaas clouds. In *MILCOM 2015 - 2015 IEEE Military Communications Conference*, pages 1554–1559, Oct 2015.

[114] C. Zhang, V. Gupta, and A. A. Chien. Information models: Creating and preserving value in volatile cloud resources. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 45–55, June 2019.

[115] Weijuan Zhang, Xiaoqi Jia, Chang Wang, Shengzhi Zhang, Qingjia Huang, Mingsheng Wang, and Peng Liu. A comprehensive study of co-residence threat in multi-tenant public paas clouds. In *Information and Communications Security*, pages 361–375. Springer, 2016.

[116] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *2011 IEEE Symposium on Security and Privacy*, pages 313–328, May 2011.

[117] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 305–316, New York, NY, USA, 2012. ACM.

[118] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-tenant side-channel attacks in paas clouds. In *Proceedings of the 2014 ACM SIGSAC Conference on*

Computer and Communications Security, CCS '14, pages 990–1003, New York, NY, USA, 2014. ACM.

[119] Yinqian Zhang and Michael K. Reiter. Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 827–838, New York, NY, USA, 2013. ACM.

[120] Yulong Zhang, Min Li, Kun Bai, Meng Yu, and Wanyu Zang. *Incentive Compatible Moving Target Defense against VM-Colocation Attacks in Clouds*, pages 388–399. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

VITA

Jin Han is from China. He is studying for his Ph.D. degree at The University of Texas at San Antonio.