SECURITY IN CLOUD VIRTUALIZATION LAYER

A Thesis submitted to Gujarat Technological University

for the Award of

Doctor of Philosophy

in

Computer Engineering

by
Dhara H Buch
[139997107002]

Under the supervision of

Dr. Haresh S. Bhatt

GUJARAT TECHNOLOGICAL UNIVERSITY
AHMEDABAD

August 2020

SECURITY IN CLOUD VIRTUALIZATION LAYER

A Thesis submitted to Gujarat Technological University

for the Award of

Doctor of Philosophy

in

Computer Engineering

by
Dhara H Buch
[139997107002]

Under the supervision of

Dr. Haresh S. Bhatt

GUJARAT TECHNOLOGICAL UNIVERSITY
AHMEDABAD

August 2020

© Dhara H. Buch

DECLARATION

I declare that the thesis entitled **Security in Cloud Virtualization Layer** submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from **June 2014 to August 2020** under the supervision of **Dr. Haresh S. Bhatt** and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.

Signature of Research Scholar: ..... Date: **18/08/2020**

Name of Research Scholar: **Dhara H. Buch**

Place: **Rajkot**

CERTIFICATE

I certify that the work incorporated in the thesis **Security in Cloud Virtualization Layer** submitted by **Dhara H. Buch** was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of Supervisor:..... Date: **18/08/2020**

Name of Supervisor: **Dr. Haresh S. Bhatt**

Place: **Ahmedabad**

Course-work Completion Certificate

This is to certify that **Dhara H. Buch** enrolment no. 139997107002 is a PhD scholar enrolled for PhD program in the branch **Computer Engineering** of Gujarat Technological University, Ahmedabad.

(Please tick the relevant option(s))

☐ He/She has been exempted from the course-work (successfully completed during M.Phil Course)

☐ He/She has been exempted from Research Methodology Course only (successfully completed during M.Phil Course)

☑ He/She has successfully completed the PhD course work for the partial requirement for the award of PhD Degree. His/ Her performance in the course work is as follows-

Grade Obtained in Research Methodology (PH001)	Grade Obtained in Self Study Course (Core Subject) (PH002)
AB	BB

Supervisor's Sign

(Dr. Haresh S. Bhatt)

Originality Report Certificate

It is certified that PhD Thesis titled **Security in Cloud Virtualization Layer** submitted by **Dhara H. Buch** has been examined by us. We undertake the following:

a. Thesis has significant new work/knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled/analyzed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using **<Urkund>** (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index < 10%).

Signature of Research Scholar:..... Date: **18/08/2020**

Name of Research Scholar: **Dhara H. Buch**

Place: **Rajkot**

Signature of Supervisor:..... Date: **18/08/2020**

Name of Supervisor: **Dr. Haresh S. Bhatt**

Place: **Ahmedabad**

URKUND

Document Information

Analyzed document	Final_Thesis_Aug_20.pdf (D77622647)
Submitted	8/9/2020 7:45:00 PM
Submitted by	Dhara Buch
Submitter email	dh.buch@hotmail.com
Similarity	9%
Analysis address	buch.dh.gtuni@analysis.urkund.com

Sources included in the report

W	URL: https://www.ijrte.org/wp-content/uploads/papers/v8i2/B3553078219.pdf Fetched: 12/13/2019 2:47:14 AM	28
W	URL: https://link.springer.com/article/10.1007%2Fs42452-020-2297-z Fetched: 5/24/2020 8:37:48 AM	67
W	URL: https://www.enisa.europa.eu/publications/security-aspects-of-virtualization/at_dow ... Fetched: 8/9/2020 7:46:00 PM	3
J	**Secure model for virtualization layer in cloud infrastructure.(Report)** URL: dd258731-ff9a-4572-a4f2-55ece137196f Fetched: 3/14/2019 9:46:23 AM	8
W	URL: https://repository.asu.edu/attachments/158040/content/Chung_asu_0010E_15260.pdf Fetched: 12/2/2019 2:49:52 PM	6
W	URL: https://hal.inria.fr/tel-02417362/document/ Fetched: 2/7/2020 11:08:38 AM	3
W	URL: https://eprint.iacr.org/2006/288.pdf Fetched: 8/9/2020 7:46:00 PM	6
W	URL: https://www.researchgate.net/publication/220334238_On_the_Power_of_Simple_Branch_P ... Fetched: 8/9/2020 7:46:00 PM	1
W	URL: https://arxiv.org/pdf/1606.01356 Fetched: 8/9/2020 7:46:00 PM	2
W	URL: https://eprints.lancs.ac.uk/id/eprint/125429/1/Cloud_Cross_VM_Attack.pdf Fetched: 8/9/2020 7:46:00 PM	1
J	**Security and Communication Networks** URL: 7783392f-43ce-4c47-8557-5f554a3be6ed Fetched: 3/9/2019 2:05:35 AM	2
W	URL: https://hal.inria.fr/hal-01591808/document Fetched: 11/15/2019 11:53:19 PM	4

URKUND

W	URL: https://core.ac.uk/download/pdf/189753208.pdf Fetched: 8/9/2020 7:46:00 PM	13
W	URL: https://img.chainnews.com/paper/bde25ea4d4a526b7291757457b720d4c.pdf Fetched: 8/9/2020 7:46:00 PM	3
W	URL: https://spectreattack.com/spectre.pdf Fetched: 8/9/2020 7:46:00 PM	1

PhD THESIS Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being a PhD Research Scholar at GTU and in the interests of the facilitation of research at GTU and elsewhere, I, **Dhara H. Buch** having **Enrollment No. 139997107002** hereby grant a non-exclusive, royalty free and perpetual license to GTU on the following terms:

a) GTU is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;

b) GTU is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);

c) GTU is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";

d) The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;

e) I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.

f) I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.

g) If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.

h) I retain copyright ownership and moral rights in my thesis, and may deal with the copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.

i) I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this non- exclusive license.

j) I am aware of and agree to accept the conditions and regulations of PhD including all policy matters related to authorship and plagiarism.

Signature of Research Scholar:.....

Name of Research Scholar: **Dhara H. Buch**

Date: **18/08/2020** Place: **Rajkot**

Signature of Supervisor:.....

Name of Supervisor: **Dr. Haresh S. Bhatt**

Date: **18/08/2020** Place: **Ahmedabad**

Seal: .

Thesis Approval Form

The viva-voce of the PhD Thesis submitted by **Dhara H. Buch** (Enrol No. 139997107002) entitled **Security in Cloud Virtualization Layer** was conducted on **18/08/2020** at Gujarat Technological University.

(Please tick any one of the following option)

☑ The performance of the candidate was satisfactory. We recommend that she be awarded the PhD degree.

☐ Any further modifications in research work recommended by the panel after 3 months from the date of first viva-voce upon request of the Supervisor or request of Independent Research Scholar after which viva-voce can be re-conducted by the same panel again.

Not Applicable

☐ The performance of the candidate was unsatisfactory. We recommend that he should not be awarded the PhD degree.

Not Applicable

Dr. Haresh S. Bhatt

Prof. (Dr.) Pramode Verma

Prof. (Dr.) K. Chandrasekaran

Abstract

Service provisioning with the highest possible level of security is a pioneer as well as challenging task of Cloud Technology. Protection of virtualization, a shielding layer, is of utmost importance to secure the Cloud Computing environment. The significance of virtualization security has motivated us to define a taxonomy on the cloud security issues with the main focus on virtualization.

The study of cloud security issues highlighted a very crucial aspect related to the probability of resource exploitation. Side-Channel Attack is such a severe attack that exploits the resource(s) when they are shared. Extraction of private key bits simply by observing performance parameters like execution time, cache access, and power consumption make the SCA very interesting to be studied as well as challenging to be explored. The exploitation of shared resources like CPU cache, network queue, a memory bus, and the Branch Prediction Unit (BPU) are different forms of SCA. A literature survey reveals that the most explored form among all of them is SCA on the cache memory, though other resources seek equal focus.

Branch Prediction Analysis (BPA) attack is such a form of SCA where the private keys of asymmetric cryptographic algorithms like Rivest Shamir Adleman (RSA) and Elliptical Curve Cryptographic (ECC) are extracted by exploiting the shared BPU components like branch predictor and Branch Target Buffer (BTB). Although it is very common to have a shared BPU, minimal work is done on the BPA attack. Moreover, consideration of virtualization is found in almost none of the research work among the limited work. The identified research gap directed us to explore more in the area of the BPA attack, especially in the virtualization layer.

One VM-One host type of typical characteristic of Virtual Machine in the virtualization layer confines the scope of intra-VM attacks. Still, it gives rise to the Inter-VM (Cross-VM) attacks. Resource sharing configuration among the VMs plays a decisive role in assessing the scope of such a Cross-VM BPA attack. The study of the total four different methods to launch the BPA attack led to analyze their dependency on the underlying resource

configuration. A Hybrid type of de facto standard configuration was found to be vulnerable to the BPA attack.

The new aspect assessing the possibility of a Cross-VM BPA attack leads to the study of the existing solution from a different perspective. Although the analysis of the existing solutions revealed their efficiency for the virtualization environment even in their direct form, their present limitations represented a need for a new solution to detect the Cross-VM BPA attack. Simulation of the attack was also found essential because of the lack of related work in the virtualization environment.

We carried out a detailed behavior analysis of the spy process during our exhaustive simulation. The normal-looking spy processes of the BPA attack differ from the other benign processes by the frequency of the primary actions they perform. The processes launching Direct Timing Attack and Asynchronous BTB Eviction Attack were observed to have high-frequency packet traversal with the VM holding RSA process. Likewise, the spy processes of TDA and both the BTB Eviction methods were found to have a high BTB occupancy ratio. Spy processes of all the four attacks read HPCnts with high frequency.

Furthermore, the spy processes need to access data to decrypt. This becomes the foundation of our solution. A four-eyed model *Chaturdrashta* is proposed to observe the identified four primary actions related to the most probable Direct Timing Attack (DTA) and Trace-Driven Attack (TDA) with four components. The logical division of *Chaturdrashta* into two sub-approaches, namely, *Trilochan* and *Trinetra,* can detect the presence of DTA and TDA, respectively. Additionally, *Trilochan* and *Trinetra* in their original form were also found capable of detecting the presence of the BPA attack launched with the Asynchronous and Synchronous BTB Eviction methods.

Experimental analysis was performed on the VMs created on the KVM Hypervisor on Ubuntu Operating System. The identification of the appropriate threshold values was carried out through experimental analysis so that the scope of false positive is almost nullified. Resource utilization of individual components of *Chaturdrashta* and performance evaluation of *Trilochan* and *Trinetra* reflected that the proposed approach takes a nominal overhead of 1%. With no architecture component manipulation and no dependency on the cryptographic algorithm, the BPA attack can be detected successfully by the time when a

few bits are predicted. Additionally, the host-based implementation of the proposed approach reduces the scope of its exploitation.

Acknowledgement

First of all, I am grateful to the Almighty for giving me the strength and capability to finish this work successfully. Also, the work presented in this thesis would not have been possible without my close association with many people. I take this opportunity to extend my sincere gratitude and appreciation to all those who made this Ph.D. thesis possible.

First and foremost, I would like to extend my sincere gratitude to my research guide Dr. Haresh S. Bhatt, for his dedicated help, advice, inspiration, encouragement, and continuous support throughout my research work. His enthusiasm, integral view on research, and mission for providing high-quality work have made a deep impression on me. During our course of interaction, I have learned extensively from him, including how to raise new possibilities, how to regard an old question from a new perspective, how to approach a problem by systematic thinking, data-driven decision making, and exploiting serendipity.

Also, I offer sincere thanks and gratitude to my Doctoral Progress Committee (DPC) Members – Prof. (Dr.) D. C. Jinwala and Prof. (Dr.) B. H. Trivedi. Without their considerations, constructive suggestions, guidance, and motivation, it would have been impossible to reach this point. Furthermore, I would like to acknowledge Mr. Rohit Tyagi, Dr. Bhavesh Borisaniya, Dr. Shivani Upadhayay, Dr. Chirag Patel, and Mr. Nilesh Vaghela from India for their motivation, support, and assistance. I am also thankful to all my colleagues at GEC Rajkot, officers at GTU, and the Directorate office for providing me a platform to pursue higher studies.

Finally, I am highly indebted to my husband, Hitarth, for his care, moral support, and cooperation. Hitarth has played a pivotal role in achieving this milestone apart from pursuing his Ph. D. I appreciate my two little daughters Aarshavi and Dheemahi, for abiding my ignorance and the patience they showed during my doctoral studies. Words would never express how grateful I am to both of you. I am also extremely thankful to my parents Nikhilbhai and Minaben, for their continuous support at every possible level. I would like to express my sincere gratitude to my mother-in-law, Ranjanben, for supporting and encouraging me in her absence too.

Dhara H. Buch

Table of Contents

List of Abbreviations

ABEA	Asynchronous BTB Eviction Attack
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Asymmetric Private Key
BIOS	Basic Input Output System
BM	BTBAccess Monitor
BPA	Branch Prediction Analysis
BPU	Branch Prediction Unit
BTB	Branch Target Buffer
CAM	CryptoAccess Library Monitor
CPU	Central Processing Unit
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DoS	Denial of Service
DTA	Direct Timing Attack
EC2	Elastic Compute Cloud
ECC	Elliptic Curve Cryptography
HPCnt	Hardware Performance Counter
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
IM	Interrupt Monitor
IPS	Intrusion Prevention System
KVM	Kernel Virtual Machine
NM	Network Monitor
MM	Montgomery Multiplication
OS	Operating System
PaaS	Platform as a Service
PCE	Performance Monitoring Counter Enable

QEMU	Quick EMUlator
QoS	Quality of Service
RSA	Rivest-Shamir-Adleman
S&M	Square & Multiply
SaaS	Software as a Service
SBEA	Synchronous BTB Eviction Attack
SBCFI	State-Based Control Flow Integrity
SCA	Side-Channel Attack
SCP	Secure Configuration Policy
SFTP	Secure File Transfer Protocol
SKC	Secure Key Cryptography
SLA	Service Level Agreement
SMI	Secure Model for IaaS
SMT	Simultaneous Multiprocessing
SOA	Service-Oriented Architecture
SPK	Symmetric Private Key
SPMA	Security Policy Monitoring and Auditing
SQL	Structure Query Language
SRMP	Secure Resources Management Policy
TDA	Trace-Driven Attack
TLB	Translation Lookaside Buffer
VM	Virtual Machine
VMBR	VM Based Rootkit
VMM	Virtual Machine Manager
XML	Extensible Markup Language
XSS	Cross-site Scripting

List of Figures

List of Tables

CHAPTER

1 Introduction

1.1 Introduction

The present era of growing technology has considerably increased the demand for hardware and software resources. Even a supercomputer cannot fulfill the resource requirements of the expanding and upcoming technologies. A Distributed Computing environment plays an important role in offering hardware and software services with a large and integrated pool of shared resources termed as Meta Computing. The Distributed Computing environments can be categorized into Cluster, Grid, Utility, and Cloud Computing environment depending on their area of applicability and the way by which they provide services.

Among these four computing environments, a heterogeneous Cloud Computing environment provides on-demand software, platform, and infrastructure services in an abstract form on a pay-per-use basis. Services offered by the Cloud Computing are accessible through the public network over the internet with the extended resource availability. Wide scope of services and feasibility in accessing them make the Cloud the most preferred computing environment where services are offered in the public domain. At this time, it is essential for Cloud Service Provider (CSP) to provide services with tight security.

Considering the significance of the Cloud Computing environment in the current era and the increased demand for authorized and authenticated services, the research was planned to be carried out in the area of Cloud Computing security. The background details, including the overview of the Cloud Computing environment and the related threats, are discussed in

Section 1.2 and Section 1.3, respectively. Further, Section 1.4 presents the thesis organization.

1.2 Cloud Computing Environment

A set of a shared and integrated pool of configurable computing resources that work together forms a Cloud Computing environment to provide scalable, Quality of Service (QoS) guaranteed and personalized computing infrastructure, on-demand with simple accessibility. Resources like networks, servers, storage, applications, and services can be rapidly provisioned and released with minimal management effort by the interaction of Cloud Service Provider (CSP) [1].

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) [1]. Multi-tenancy, Scalability, Elasticity, and Self-Provisioning of Resources are the main characteristics of Cloud Computing. Small Initial Investment, Low Ongoing Costs, Economies of Scale, Open Standards, and Sustainability are the key features of this environment [2]. Limited points of failure also increase the reliability of the Cloud Service Provider (CSP). The cloud model is composed of essential characteristics like On-demand self-service, Ubiquitous network access, Resource pooling, Location independence, Rapid elasticity, and Measured service.

The services of Cloud Computing are provided through three service delivery models, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The request for hardware and software resources are satisfied ‘on the fly’ as per demand via these service models. Based on how a person leverages SaaS, PaaS, and IaaS, his/her role is decided as either of Provider, Consumer, or Integrator.

The areas where cloud services are offered categorize cloud offerings into three deployment models, Private, Public, and Hybrid.

Cloud providers provide almost unlimited services to cloud consumers with the characteristics of rapid elasticity. The services are provided by keeping the underlying

hardware and software details hidden from the end-user through an abstraction layer called virtualization. The primary elements of a Cloud Computing environment are represented in Figure 1.1, and a brief discussion on each of them is presented in the following subsections.

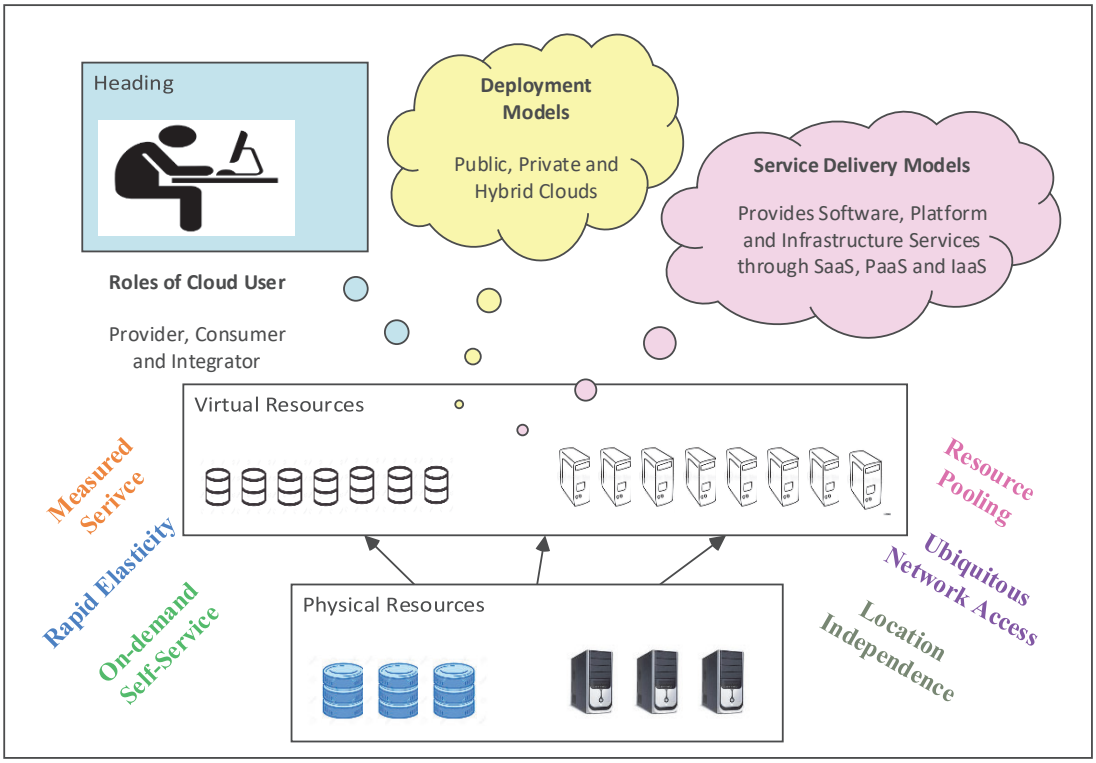


Figure 1.1: Elements of Cloud Environment

1.2.1 Deployment Models

Cloud technology has three types of deployment models: Public, Private, and Hybrid. An institute or organization owns and manages the Private clouds for exclusive use by a specific community. Private clouds are more expensive and secure than Public clouds, and they place no limits on bandwidth limitations. As access is limited to the clients belonging to the private organization, additional security is not required to be imposed. Public Cloud, on the other hand, is fully managed by the service provider, and it is for open use by the generic public. Restricted control over the infrastructure in Public clouds requires high-security regulations. The composition of two or more different types of deployment models established to share and transfer services without affecting an individual's system is known as the Hybrid cloud. A Hybrid cloud is designed for the means of incorporating combined characteristics of Public and Private clouds.

1.2.2 Service Delivery Models

Resource provisioning is the primary function of a Cloud Computing environment, where the types of services include allocation of hardware resources, providing data storage facility with backup and offering system, and application software resources. This wide range of services is classified into three categories: Software, Platform, and Infrastructure services. They are offered through three service delivery models, SaaS, PaaS, and IaaS, respectively.

1.2.2.1 Software as a Service

Cloud Service Provider (CSP) offers various software applications on the cloud infrastructure to the consumers through Software as a Service (SaaS). The clients can access these applications from the client devices without any need to install and maintain software and without worrying about the underlying resources like network, servers, Operating Systems, data storage, or configurable application settings. Availability and accessibility of the served applications may vary from user to user [1].

1.2.2.2 Platform as a Service

Platform as a Service (PaaS) model offers an integrated software development environment. It provides the base with which various applications, libraries, tools, and services are created. PaaS also provides required Operating System support along with a platform to handle an application. PaaS offers the complete lifecycle of software development from planning, design, implementation to deployment, and testing.

1.2.2.3 Infrastructure as a Service

Hardware and software resources as per demand are readily made available to clients by Infrastructure as a Service (IaaS). The CSP owns the equipment, and has the responsibility of housing, running and maintaining them in a virtualized manner. IaaS offers infrastructure, i.e., processing, storage, network, Operating Systems, and such other computer resources as per the demand of users.

1.2.3 Roles of Cloud User

There are three key roles of the Cloud Computing paradigm: Provider, Consumer, and Integrator. Provider and Integrator have their own responsibilities related to each other. A provider provides services to either a consumer or to another provider, while a consumer consumes services offered by a provider. Consumers have no responsibilities, as they do not have access beyond the services provided to them. Integrator, as the name suggests, collects services from many providers, assembles them, and ultimately introduces a new service to the Consumers. Each party can hold multiple kinds of roles depending on the perspective [3].

1.2.4 Virtualization

Cloud Computing offers software and hardware services from the available set of resources where the resources are physically scattered but connected via a network. The service is offered through a software abstraction layer that lies between the physical resources and end-users. This abstraction layer is known as the virtualization layer. A very large set of heterogeneous scattered resources is integrated to represent a single unit of resource to the users where virtualization hides the internal detail of the system from the users. Virtualization is supported mainly by Resource Sharing and Isolation. Resource Sharing allows hardware and software resources to be accessed concurrently by multiple applications. The property of isolation is supported by prohibiting the running programs from interfering in each other's state despite the parallel access to the shared resources [4].

Physically scattered hardware and software resources are divided into various logical partitions to form a virtual view as per the request made by the user and provided to the end-user as a Virtual Machine (VM). Virtual Machine gives a view of several independent computers running independently, while they may be simultaneously running on the same physical host with the support of Isolation. Virtual Machines are also known as the guest machines having their own guest Operating System (OS). The architecture of the virtualization layer is shown in Figure 1.2.

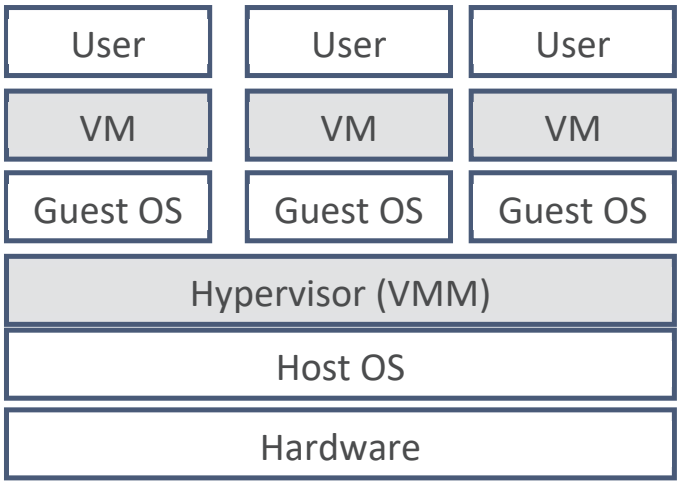


Figure 1.2: Virtualization Architecture

One VM cannot directly access the resources of the other VM as well as of the host machine on which it is operated. Isolation between VMs, and between VM and host machine is provided by a software layer in between, called a Virtual Machine Monitor (VMM) or a Hypervisor. It is also an important aspect of virtualization security is that even when a VM is crashed, the other VM must not stop functioning [5]. Additionally, dynamic VM migration also takes place to handle issues like fault tolerance and load balancing, which are also very important aspects of virtualization.

All the virtualization platforms have a set of virtualized networking components which is needed to be configured for performance, availability, and security. Scattered heterogeneous systems are physically connected with each other via a physical network managed with a physical switch. To support virtualization, these systems are grouped to form logical partition with virtual network and virtual switch, which are software-based representations of hardware types.

1.3 Threats to Cloud Computing Environment

A large set of integrated resources for service provisioning adds essential features like Rapid Elasticity and Scalability to the Cloud Computing environment, but with an increased risk of security. As services are made accessible publicly, the probabilities of attacks also go high. The cloud environment is a multi-tenant environment, where the hardware and

software resources are shared among virtual machines. Data created, used, managed, and destroyed by the cloud users are handled by the third-party CSPs, where the data is stored in a central repository called Data Centres. All these features of the cloud environment make Multi-Tenant Environment, Data Centre, and Loss of Control major areas for security consideration.

Cloud Computing offers advantages like virtualization, flexibility in choosing vendor, elasticity, and cost reduction to the clients [6]. However, at the same time, the opportunities to Cloud Computing also bring some obstacles that the CSPs and service consumers need to tackle.

There are seven security issues, according to Gartner, that the customers should raise while selecting a cloud vendor: Privileged user access, Regulatory compliance, Data location, Data segregation, Recovery, Investigative support, and Long-term viability [6]. Cloud Security Alliance (CSA) has identified top threats to Cloud Computing as Abuse and Nefarious Use of Cloud Computing, Insecure Interfaces and API (Application Program Interface), Malicious Insiders, Shared Technology Issues, Data Loss or Leakage, and Account Hijacking. These significant threats to the cloud environment are discussed here.

1.3.1 Abuse and Nefarious Use of Cloud Computing

Easy access to almost unlimited hardware and software resources has become possible by various services offered by CSPs. Although these services are provided on pay per use basis, free limited trial periods are also offered by some of the providers. In such a case, it becomes very manageable for malicious users to disturb and intervene in various activities of IaaS, PaaS, and SaaS. By leveraging new technologies, criminals perform disruptive activities like data stealing, service hijacking, denial of service, etc. even by hiding their identities.

1.3.2 Insecure Interfaces and APIs

Cloud users are provided services through respective interfaces and APIs by the CSPs. Different types of service offerings are managed through the appropriate APIs. Accordingly, the interfaces and APIs are the central parts of the Service-Oriented

Architecture of the cloud environment. Designing of these interfaces and APIs must be done in such a manner that they do not get exploited.

1.3.3 Malicious Insiders

A malicious insider is one of the most common threats to the cloud environment. Growing IT services and customers converge to the same domain, which provides an easy way for infiltration to the malicious insiders. Providers keep the underlying details hidden, which results in an attractive opportunity for an attacker that ranges from an ethical hacker to organized crime.

1.3.4 Shared Technology Issues

IaaS offers scalable services that require the sharing of resources among multiple tenants. Strict isolation of the tenants is indispensable among tenants for secure services, but the resources are not designed to support it. Although a layer of virtualization is introduced to provide separation, many vulnerabilities can easily break this layer of isolation. Exploited Virtual Machines and Hypervisor can raise many security issues. Prevention of such issues requires compartmentalization of resources for strong separation.

1.3.5 Data Loss or Leakage

The threat of data loss can be avoided up to a certain extent by taking the backup of the data. However, it becomes impossible to recover the data in the absence of backup or the loss of the encoding key itself. In cloud architecture, data is stored in central storage, making it essential to have all the data recovery options available. CSP provides support for data protection, but various ways employed by the adversary for stealing data raises many data security issues.

1.3.6 Account or Service Hijacking

Reused and weak credentials can allow the adversaries to steal them. Access to these credentials makes it possible for the adversaries to eavesdrop activities, manipulate data, redirect to illegitimate sites, and falsify transactions. Robust authentication techniques, proactive monitoring, knowledge of various security policies of CSPs can protect the cloud users from getting trapped in such a problem.

1.3.7 Unknown Risk Profile

Each step towards the up-gradation of features and an increase in types of services expands the security attack surface also. Hence, security must be given the central focus while revising the system for the means of better performance. Moreover, the probable vulnerability issues should be given more consideration than those which are identified from the past records. Identification of such vulnerable issues helps in making the system more sustainable and secure, even in the presence of unknown security attacks.

1.4 Thesis Organization

The study of the Cloud Computing environment has revealed that the issue of security is of utmost importance while leveraging cloud services. The study has also shown that among all the special features of the Cloud Computing environment, the virtualization plays an important role in providing secure services. Although cloud security is increased by the resource hiding feature supported by virtualization, the added layer of abstraction also carries many loopholes from which many security attacks are still possible. The role of virtualization in tightening the cloud security has motivated us to work in the area of Cloud Computing security with virtualization as the main focus of our research.

With a brief discussion on the Cloud Computing environment and various threats to it in this chapter, we proceed with a literature survey in Chapter 2. The existing literature can be divided into two groups – (i) the literature that deals with generic cloud security issues (ii) the literature that deals with virtualization-based cloud security issues. Designing of a new taxonomy on cloud security issues with a central focus on virtualization and an in-depth analysis of existing work done on each of the classified categories in the taxonomy are the major parts which were carried out during the survey work. The chapter is ended with a summary that reveals the research scope in the area of Branch Prediction Analysis (BPA) attack in the virtualization environment.

Chapter 3 begins with the basics of the BPA attack. A detailed discussion on the BPA attack vulnerable algorithm, types of methods launching the BPA attack, difficulties in detecting the attack, and existing approaches for handling BPA attack is presented in this chapter.

The discussion, presented in Chapter 3, reveals that the scope of BPA attack in the virtualization environment is required to be analyzed. A new approach is necessary to be developed to handle the BPA attack in the virtualization environment. The identified research gap in the area of BPA attack in virtualization leads to define the problem statement in Chapter 4. The objectives and scope of the present work are also discussed in the same chapter from the study carried out in the previous three chapters.

A comprehensive study of the BPA attack in the virtualization environment is taken as the main objective. To justify the claim, the scope assessment of the said attack is presented in Chapter 5. Primitive features of the virtualization environment and possible resource sharing configurations make significant parameters in deciding the possibility of BPA attack in virtualization. A detailed discussion on this aspect in the initial part of the chapter is followed by re-visit to the existing solutions for assessing their applicability in virtualization. The chapter is summarized with an identified need for a new solution to overcome the limitations of existing ones.

Chapter 6 covers the major portions of the research contribution. The former part of the chapter presents the results and related discussion on the attack simulation. The simulation provided a direction for studying the behavior of the attack methodologies. A detailed discussion on the behavioral analysis presented in this chapter threw light on many primary features of the attack methods. Proposal for a new solution based on the features identified from the behavior analysis forms the later part of the chapter. It includes the model, flow, and implementation details of the proposed solution, namely *Chauturdrashta*.

Experimental analysis of the proposed solution is the major point of discussion of Chapter 7. The simulation setup and obtained results are discussed in this chapter. The chapter is ended with the performance comparison and overhead calculation of the proposed approach.

In the last chapter, Chapter 8 presents the conclusion and the scope for future work.

CHAPTER

2 Literature Survey and Research Challenges

The study of the cloud environment and the significance of the underlying features reveal the role of security while providing services. The study of the cloud environment and its underlying features reveal the significance of security while providing services. The importance of security for a Cloud Computing environment has motivated us to do research work in that area. The initial phase of the work included the study of the research work already done on various cloud security issues. A brief discussion of this study is given in Section 2.1. The analysis of the previous work led us to define a taxonomy to represent the cloud security issues with the main focus on virtualization. A detailed discussion on research work done in handling generic cloud security issues as well as that in handling security issues at the virtualization layer is carried out in Sections 2.2 and 2.3, respectively. At the end of this chapter, the survey is summarized to identify the area in which more research work is required.

2.1 Survey of Cloud Security Issues

The study of security issues in the Cloud Computing environment emphasizes the primary factors affecting cloud security. Discussion carried out in Section 1.2 shows that virtualization is the core layer in providing abstract services and consequently helps in providing secure services. However, the virtualization layer itself is also vulnerable to many kinds of security attacks. In that regard, it is significant to consider the security issues at the virtualization layer in addition to the generic security issues.

Accordingly, extensive research work done in the area of cloud security can be divided into two broad categories:

A. Survey of literature dealing with generic cloud security issues

B. Survey of literature having cloud security with virtualization as the primary focus

A detailed literature survey was carried out for both of the above categories. As far as category A is concerned, Khan et al. [7] have divided types of attacks as Network, VM, Storage, and Application based attacks. Attacks possible under each of the categories are discussed along with the existing countermeasures. Network-based attacks (like port scanning, spoofing, botnets), VM attacks (like VM creation, migration, Cross VM, rollback), storage base attacks (like data duplication, Scavenging), and Application based attacks (like malware injection, web-based attacks) and shared architecture-based attacks are discussed along with the possible implications. The application of the Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) to handle various attacks are also discussed. Additionally, the research also considers features to regulate cloud compliance and privacy factors. The paper gives an excellent discussion on generic cloud security attacks.

In another approach on the same track, Gonzalez et al. [8] have classified cloud security issues into network security, interfaces, data security, virtualization, governance, and compliance. They categorize the taxonomy of cloud security into Privacy, Architecture, and Compliance. They also present a comparative analysis of various Cloud Computing services all over the world, focusing on how the issues are managed for improvisation. By giving the graphical representation of existing contributions in various issues, the paper also identifies the areas with more research scope.

Gruschka et al. [9] have categorized cloud-related functions into invoking service, using cloud and managing clouds. They define six attack surfaces based on these functions. They discuss the application of the suggested taxonomy for four types of attacks, namely, Amazon EC2 (Elastic Compute Cloud) Hack, Cloud War, Cloud Malware Injection, and Side-Channel Attack (SCA). They only give a rough sketch of probable attacks without considering the countermeasures.

Srinivasan et al. [10] have categorized Cloud Computing security taxonomy into two aspects: Architectural-Technological and Process-Regulatory. They discuss each of the identified parameters under the categories, namely logical storage segregation, identity management, insider attacks, virtualization, cryptography, governance, insecure API, CSP migration, and SLA (Service Level Agreement) trust gap.

Coppolino et al. [11] have also discussed generic cloud security issues. They have listed open issues in a cloud environment and listed security challenges like shared technologies vulnerabilities, data breach, Service Traffic Hijacking, Denial of Service (DoS), and malicious insiders. They map the identified challenges against three attack vectors, namely, network, hardware, and Hypervisor. They discuss existing work and suggested countermeasures for each of the attack vectors with the conclusion that malicious insider is an open issue that requires more research work. The approach gives a rough idea about identified challenges.

Singh et al. [12] have presented a generic but in-depth analysis of cloud security issues. The embedded system, application, trust, client management, clustering, data management, and Operating System (OS) based issues have been examined to discuss the security issues related to them. The study also includes discussion on issues like probable threats with the compromised attribute and security concern with the cloud service providers. Security solutions on digital forensics, virtualization, fault tolerance, and risk analysis are also presented, which measures cloud security in an utterly generic yet efficient way.

In another study on generic cloud security, a detailed taxonomy defining layer-wise security aspects have been presented by Hashemi et al. [13]. They categorize probable security issues into four domains, namely, an infrastructure layer, a platform layer, application layer, and administration of the Cloud Computing environment. Further, for each of the defined categories, possible threats are listed and described briefly. Although they present a detailed taxonomy, including most of the security issues, the discussion does not include countermeasures for handling these issues.

The analysis of the above-mentioned research work reveals that the generic cloud security issues have been examined in detail without giving much insight into virtualization security.

At the same time, virtualization is the main focus of some of the researchers. As far as such virtualization-centric research mentioned in category B is concerned, Pe'k in [14] has discussed hardware virtualization attacks in depth along with a brief discussion of countermeasures. They consider attack targets like a Hypervisor, VM, host OS, management of interface, and different networks as a base for categorizing virtualization attacks. The paper provides excellent material for studying hardware virtualization attacks in detail.

An in-depth analysis of virtualization-based attacks and related handling approaches have been attempted by Modi et al. [15]. Giving a detailed description of firewall types and Intrusion Detection / Prevention Techniques, they have shown how these can be used to tackle various virtualization attacks. They have shown limitations of each such approach with the conclusion that a cloud IDS with defined features is needed to tackle all the virtualization attacks.

Analytical research from an entirely different angle is carried out by Sgandurra et al. in [16] that presents probable attack paths. With an objective to provide a common framework for security solution evaluation, virtualization-based attack solutions are discussed considering the threat model, i.e., hardware, virtualization and cloud assumptions, security properties with Trusted Computing Base, and implementation strategies. Designing a novel solution becomes comfortable with this common base of assumptions and methodologies.

From all the virtual components, the primary study on Hypervisor security has been done by Riddle and Chung in [17]. Hypervisor targeting attacks like Return Oriented Programming, VM Rollback, Side-Channel Attack, Non-Control Data, and so forth are discussed along with their existing countermeasures. They conclude VM Escape as the most difficult attack to cope with. Some of the researchers have presented taxonomy on the virtualization security, which we have discussed here.

Analysis of the research work carried out in both categories A and B reveals that none of them covers both generic and virtualization security issues, including the countermeasures handling them. Accordingly, we planned to define a taxonomy that includes generic cloud security issues with the main focus on virtualization security.

The large surface area of cloud security includes a large number of security issues. However, consideration of the root cause of all the security issues maps to mainly two factors: Multi-Tenancy and Loss of Control. Simultaneous access to the shared resources among tenants raises severe security issues. The central data store is the primary reason behind all the issues related to data protection. At the same time, the control and management of data, as well as service by the third-party Cloud Service Provider (CSP), also increase the risk. Hence, consideration of the security issues related to Multi-Tenancy and Loss of Control cover almost all types of generic issues in a broad sense. However, the study of cloud security cannot be ended without considering SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) as they are the primary elements in providing cloud services [18]. Additionally, according to Almorsy et al. [19], as SaaS is built on the top, i.e., on PaaS and IaaS, security issues of these models are inherited by SaaS also. Working in this direction, cloud security issues are categorized as attacks on the three service delivery models by Iqbal et al. [20]. They discuss attacks like DoS, SQL (Structured Query Language) Injection, Phishing, XSS (Cross-site Scripting), VM Escape, VM rollback, Man-in-the-Middle with an analysis of how IDS/IPS can be applicable for handling them.

Summarizing the above discussion, we planned to include security issues related to Multi-Tenancy, Loss of Control, and issues related to Service Delivery Models in addition to virtualization related security issues as part of our taxonomy. Accordingly, the presented taxonomy on cloud security, considering generic issues along with a primary focus on virtualization security, is shown in Figure 2.1.

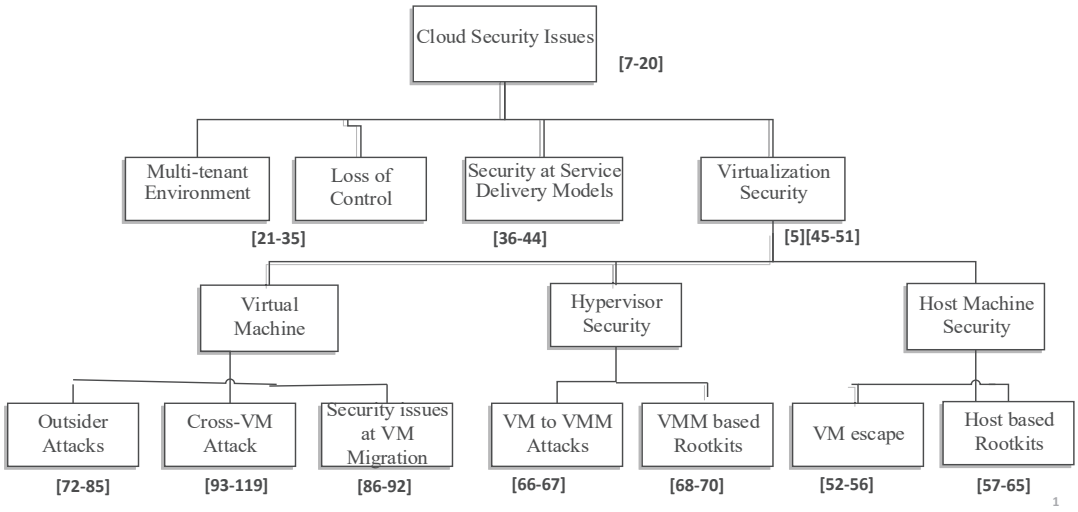


Figure 2.1: Cloud Security Taxonomy

A detailed discussion of the research work carried out in the security issues listed in Figure 2.1 is presented in the following sections.

2.2 Generic Security Issues in Cloud Computing Environment

Security issues at the Cloud Computing environment were broadly categorized into generic and virtualization-base security issues, as discussed in the previous section. Moreover, we identified issues related to the Multi-Tenant environment, Loss of Control, and service delivery models as the three major areas to be covered while studying cloud security. An overview of the existing work done in handling them is given in the following subsections.

2.2.1 Multi-Tenant Environment

In a Cloud Computing environment, concurrent access to software and hardware resources are provided to multiple users. The pool of shared resources includes both small and large, simple, and complex resource entities. VMs sharing resources are known as multi-tenant VMs, and the environment is known as a multi-tenant environment.

In the presence of virtualization, the details of resource sharing are kept hidden from the users giving them an impression of being an only user of that resource. Although resource sharing in a multi-tenant environment increases the overall efficiency of the system, it also increases many security risks. From the security perspective, preventing tenants

from accessing other tenant's resources and also from saturating them are important parameters to be considered for secure multi-tenancy.

Among all the hardware and software resource sharing in a multi-tenant environment, data sharing is a very important entity to deal with. In a cloud environment, data is stored in a Data Centre, which is a centralized repository for storage, management, and dissemination of data and information. Datacenter aggregates storage resources into a common pool, and such large logical storage volume is created. Such a physically scattered and virtually united contiguous storage structure raises many security issues as it is shared among multiple independent clients. Issues like data protection, data isolation, data validation, and authentication are very sensitive, and they need to be handled properly to prevent data exploitation. For data protection, some encryption mechanism can be applied to protect data from unauthorized access. It is also needed to restore the data when it is lost, or its decryption fails. Even it is needed to assure the integrity of the data when VM Migration takes place [21].

Many researchers have discussed data security issues in the multi-tenant environment. T. Takahashi et al. [22] have discussed security challenges and issues at the web, application, Operating System, Hypervisor, and hardware-software layers. Each defined layer is categorized into types of security issues specific to it. The available research done on each of them is also presented. Gansen Zhao et al. [23] have proposed five models, namely, Separation, Availability, Migration, Tunnel, and Cryptography, for handling different data security-related issues. A framework for cloud security based on Identity Management System assuring correctness in user's data on a cloud storage system is presented by Aishwarya C. S. in [24]. In another approach, M. Sudha et al. [25] have implemented a Data Protection Package, which, for a client-server model, maintains authenticity and confidentiality of exchanged data with suggested methods of encryption and decryption. An approach to enhance data authenticity, privacy, and integrity is proposed by M. Gobi et al. [26], which uses encryption schemes and hash functions for data hiding.

Some researchers have worked on dynamic data supporting a distributed scheme. One approach in that direction is proposed by Cong Wang et al. [27], which uses erasure-

correcting code for data authentication with reduced overhead. It also provides localization of data error. The efficiency and resiliency of the proposed approach is proved with performance analysis. Similar approaches are proposed by D. Purushothaman et al. [28] and K. Shiriha Reddy et al. [29]. Through a homomorphism token, a distributed scheme is proposed by them to verify erasure-coded data and handles the colluding attack.

Amit Sangroya et al. [30] analyze data security with risk analysis for some trust parameters. In another method of handling data security by G. Rakesh Reddy et al. [31], it is suggested to distribute the cloud data among multiple CSPs to support data availability and secure storage rather than relying on a single service provider. A. Juels et al. [32] have provided a data auditing framework for ensuring integrity, along with designing protocols for increasing data authenticity.

Data security is handled with a different angle by Smith D et al. [33], where an anomaly detection IDS for automatic management of a cloud system is presented. They have proposed mechanisms for identifying data transformation and selecting features to reduce data size. They have also devised an approach to detect abnormal behavior in an automatic mode.

The above analysis reveals that different researchers have focused on various aspects of data security like data correctness, data protection, dynamic data support, etc. and thus, they help in designing a secure multi-tenant environment with the central data store.

2.2.2 Loss of Control

Services provided by the cloud system require to be transitioned to the CSP. User sensitive information and data are owned, handled, and controlled in the cloud by a third-party provider. Therefore, the danger of misuse, theft, and destruction of data also increases. Management of operations and decisions about computing environments requires the cooperation of CSP. Thus, as the control on resources is lost by the owner, not only the CSP selection but also the data designing should be carried out carefully considering the security aspect.

When there is a loss of control in physical as well as logical aspects, legal protections for privacy and maintenance of accountability becomes very challenging as the data is owned and handled by the third-party CSP.

Handling issues raised due to Loss of Control requires consideration for authentic third-party service provisioning and authentic data services. The later one merely refers to Data Centre security. With a focus on tightening third-party service security, in [34], K. Hamlen et al. discusses security issues related to storage and data layers. They have discussed a scheme to secure third-party publication of documents and to secure query processing with Map Reduce and Hadoop. Following the same track, a security framework for increasing collaboration among cloud providers, service providers, and service consumers is proposed by Mohemed Almorsy [35].

2.2.3 Security Issues of Service Delivery Models

From the three service delivery models, IaaS is the foundation layer. PaaS is built on IaaS, and SaaS is built on PaaS. Each of the service delivery models has its types of security issues. However, because of the inter-dependency, all the issues related to IaaS affect the PaaS also. Similarly, SaaS inherits all the issues of PaaS and IaaS. For providing secure services to the consumers, the service provider needs to take care of all the issues raised by any part of the architecture.

In addition to the software level security, the SaaS model is also responsible for granting various files and application permissions to the users. Hence, identity management becomes an important aspect of assuring authorized and authenticated SaaS services. Cloud architecture is a Service Oriented Architecture (SOA), so PaaS security implies SOA based security. All security issues which exist in SOA domain like Man-in-the-middle attacks, XML (Extensible Markup Language) related attacks, Replay attacks, Dictionary attacks, Injection attacks are the issues of PaaS security also. Additionally, PaaS also provides APIs for managing applications and security. Providing authorized APIs and preventing API manipulations are also parts of PaaS security [18].

IaaS offers infrastructure, i.e., processing, storage, network, Operating Systems, and such other computer resources as per the demand of users. Hence, the security of such infrastructural resources falls under the IaaS security. Additionally, the Service Level Agreement, Utility Computing, Virtualization, Cloud Software, Network & Internet Connectivity, and Computer Hardware are the main components of IaaS [36]. Security of these IaaS components is also necessary to ensure IaaS security

In a Cloud Computing environment, services are provided in abstract form. IaaS combines various resources from separate physical systems and provides the single logical unit of requested infrastructure as a Virtual Machine (VM). With virtualization, IaaS hides the infrastructural details from the users, which helps in providing security. However, there are many kinds of security attacks that break this shielding layer. A detailed discussion on virtualization layer related security issues are presented in the next section.

Tapukula et al. [37] present some IaaS level intrusion detection techniques that can differentiate the attack traffic originating from each VM, even when multiple VMs share a single IP address. Vaquero et al. in [38] has focused on security issue raised due to multi-tenancy at the IaaS layer and has concluded that essential security approaches propose security with encryption techniques or access control mechanisms. They have given a detailed analysis of machine virtualization, network virtualization, and physical domain. A policy-based security approach with the dynamic model has presented by M. Yildiz [39] where the infrastructure scope for network, server, storage, and system management domains is covered.

Arora et al. [40] have proposed a Secure Model for IaaS (SMI), which defines the relationship between IaaS Components and related security requirements with a cubic model. Components of IaaS like Utility Computing, Platform Virtualization, and Computer Hardware are shown to satisfy the security requirements with the help of Secure Configuration Policy (SCP), Secure Resources Management Policy (SRMP) and Security Policy Monitoring and Auditing (SPMA). They claim that given entities provide a base for standardization of the IaaS layer. Another IaaS based flexible and portable

signature-based IDS framework entirely controlled by cloud consumers is proposed by Alharkan et al. in [41].

Among the research work on SaaS or PaaS, Nascimento et al. [42] focused on SaaS web applications in the cloud. They had proposed an anomaly-based IDS for gathering and processing a significant amount of data without threat of an attack. A comparison of the accuracy of the obtained results is also carried out with that of different models.

Subashini [43] has discussed various SaaS security issues like Data security, Network security, Data locality, Data integrity, Data segregation, Data access, Authentication, and Authorization along with a brief discussion on PaaS and IaaS security issues. Jensen et al. [44] have discussed various security issues like XML Signature and Browser Security (SaaS level), Integrity and Biding Security (PaaS), and Flooding attack security (IaaS) issues.

The above discussion reveals that many researchers from various angles analyze security issues related to the three service delivery models. The study of security issues related to multi-tenancy, loss of control, and service delivery models can be combined represented as generic security issues. The next section encompasses the security issues related to virtualization, which is the primary focus of the proposed taxonomy.

2.3 Security Issues of Virtualization Layer

The virtualization is meant to provide a virtual view of all the information and computing resources. Although the concept seems simple at first sight, the need for security in a virtual environment makes the actual implementation is equally complex. According to Kusnetzky et al. [45], implementing security as part of architecture works more efficiently rather than adding it as a patch later on. Accordingly, it can be said that security issues must be considered from the root level rather than analyzing only surface-level problems. The study of virtualization security includes security analysis of each of the virtualization components.

A virtualization environment faces many types of threats where some are accidental, while malicious Insider/Outsider attackers cause others. Accidental threats are known as

operational attacks. VM sprawl, lack of visibility into the virtual environment, unmaintained separation of duties, etc. are such operational threats. Threats which are more realistic and generated because of malicious packets and code like bots, worms, virus, and rootkits are known as malware-based threats.

Virtualization allows the cloud clients to demand any software or hardware resources like processor, memory, disk, Operating Systems as an integration of a single logical unit, which is actually comprised of many scattered physical units. Demand on pay per use basis increases the scalability of the environment. This leads to a varying number and types of VMs that appear and disappear in the network for a unit of time. The lifecycle of the software running in a virtual environment also differs from that in a traditional environment. Mobility, Diversity, and Identity are also the points to be considered for the security of the virtual layer [46].

From the virtualization-centric contributions, Kirch et al. [47] discuss the concept of virtualization along with its various types. It has discussed possible threats to virtualization like VM Escape, Denial of Service attacks, security threats among VMs and VMM. A detailed description of the way of handling them with software and hardware tools is also presented in the same. A similar discussion is carried out by Reuben et al. in [48].

Security issues in the virtualization environment are also discussed by Anand et al. [49]. In another approach, Brooks et al. [50] have analyzed security vulnerabilities in a virtualized environment where attacks like virtual cloud injection, Hypervisor traversal, and virtualized botnets are discussed along with focusing on various components.

Potential of virtualization security risks are suggested by Nan et al. [5], while Li et al. [51] proposed architecture for virtual machine security, virtual network security, and policy-based trust management services.

As the Hypervisor and Virtual Machines are the main components of the virtualization layer, the security issues specific to them are discussed in the following subsections. Further, as these components are operated on the host machine, the security of the host machine is also included in the virtualization security.

2.3.1 Host Machine Security

In a virtualized environment, the host system cannot be accessed directly by the users as the user interface is provided via Virtual Machines only. Despite this structure, there are some cases where the host system is attacked by exploiting the intermediate layer separating virtual and physical layers. Additionally, a compromised host machine ultimately affects each component of the virtualization layer and thus requires being completely error-free. It can be attacked either by VM bypassing VMM or some kernel-level rootkits discussed in the following subsections.

2.3.1.1 Guest to Host Attack (VM Escape)

Guest programs are needed to run in a completely isolated environment without having direct access to physical resources. Sometimes the bugs in the virtualization software bypass the VM layer completely, and guest machines get full access to the host machine by ignoring the presence of Hypervisor. This attack of the guest machine on the host system is known as VM Escape attack. In this type of attack, the VM bypasses the Hypervisor and attacks the host directly. Cloudburst [52] for VMWare and Virtunoid [53] for KVM (Kernel Virtual Machine) are presented as VM Escape attack, but still, useful real-time implementation is yet not devised. According to N. Elhage et al. [53], such an attack can be employed by injecting the Dom 0 root process with bind shellcode to access memory directly. Also, by escalating Hypervisor privilege or lowering calling domain privilege, the attack can be implemented. For handling VM Escape, approaches maintaining Hypervisor integrity are implemented in [54] and [55] by J. Wang et al. and Z. Wang et al. respectively. In a solution [56] by Szefer et al., the Hypervisor is bypassed to make VM communicate to the hardware by implementing methodologies like pre-allocation of resources, modified guest OS, and usage of virtualized I/O. The elimination of Hypervisor also eliminates the probability of this attack, but practically it is challenging to be implemented.

2.3.1.2 Host-based Rootkits

Kernel level rootkit is malicious software that is designed for getting privileged access to the system and cannot be detected unless a dedicated monitoring system is established. Kernel Level rootkits work at Ring 0 and alter the BIOS (Basic Input

Output System) booting sequence so that the rootkits get executed before the execution of the bootstrap program starts. As it hides its presence, to detect a kernel-level rootkit is a challenging task to tackle with.

Kernel level rootkits reside and operate from ring 0 and thus affect the operation of system resources directly from the root level. Approaches mainly observing and handling control flow to identify such rootkits are proposed, which operate either from kernel-level or Hypervisor-level.

Petroni et al. [57] have presented an approach based on State-Based Control-Flow Integrity (SBCFI), for dynamically monitoring Operating System kernel integrity. It claims to have detection ability of most of the control-flow modifying rootkits with negligible scalability overhead.

Riley et al. [58] have presented a VMM based approach NICKLE that keeps a shadow copy of running VM kernel code with memory shadowing and prevents all the unauthorized kernel code access. In their research, Seshadri et al. [59] have presented the design of a tiny Hypervisor, ensuring kernel code integrity by allowing only user-approved code to be executed in kernel mode. It prevents zero-day attacks, although it cannot prevent control-flow attacks. Both approaches support scalability with space overhead, while approaches discussed subsequently add time overhead when implemented with scalability support.

Patagonix is one such approach where a VMM based system is proposed by Litty et al. [60] that detects covertly executing binaries' code modifications and rootkit. For detection, it focuses on processor hardware without considering the OS kernel assumption.

Another Xen based kernel monitoring mechanism by Srivastava et al. [61] trusts no data of OS. However, with the wizard, the proposed monitoring utility, securely and efficiently, intercepts application and OS level behaviors.

Jones et al. [62] have implemented and evaluated VMM-based hidden process detection and identification service called Lycosid. Lycosid can detect the presence of hidden processes and uses implicit information about guest OS to keep itself less vulnerable to guest-initiated attacks.

An approach named RootkitDet is implemented by Zhang et al. [63], which observes guest OS kernel space to identify suspicious code. After identification, it diagnoses the kernel data structure to locate the area that is modified by a rootkit. Finally, it undoes the malicious actions by altering the modified content with its previously known content.

The effect of the approaches preventing the execution of kernel rootkit was bypassed when Return-Oriented Rootkits [64] came into existence. It bypasses the mechanisms set for kernel code integrity protection. Hooksafe [65] was proposed to mitigate this malware. It relocates all the scattered kernel hooks in a contiguous place. Through a thin hook indirection layer, each hook request is diverted to this shadow copy. This approach also traps each attempt to modify the shadow copy.

The above discussion reveals that efficient methods are existing for preventing security attacks on the host system. However, the attack like VM Escape is yet to be dealt more effectively for protecting the host system from directly getting attacked from the guest systems. The security of Hypervisor becomes a very important parameter to prevent such attacks. In that regard, the next subsection discusses the security issues at the Hypervisor.

2.3.2 Hypervisor Security

The Hypervisor or Virtual Machine Monitor (VMM) is the main abstraction layer that isolates virtual machines from the host system. It is a software layer that separates the virtual and physical systems. Damage in the Hypervisor breaks the virtualization layer, and ultimately the resource hiding feature is also compromised. A VMM is mainly attacked either by VMs or by rootkits. Attacks to Hypervisor must be detected/prevented for secure virtual services. A brief discussion of different kinds of attacks to Hypervisor is carried out in this section.

2.3.2.1 VM to VMM attack

The Hypervisor provides isolation among various VMs. Privileged instructions or hypercalls generated by the VMs are required to be translated into appropriate system calls by the VMM.

The presence of Hypervisor isolates VMs from the host machine, and so when the Hypervisor itself is compromised, the abstraction layer breaks down. Such types of attacks are carried out by the applications running on VMs, which target VMM to exploit the virtualization.

In virtualization, VMs need to communicate to VMM, especially when a privileged instruction is executed. The execution of privileged instruction leads to the generation of hypercalls (for Xen Hypervisor) or system calls. Validity testing of generated hypercalls with privileged domain Dom 0 is proposed by Bharadwaja et al. [66] for the Xen environment. In a similar kind of approach, Y. Du et al. [67] propose a VMM based model that, with Behavior Gathering and Analysis components, compares system calls' behavior with a normal one. It also identifies abnormality on the basis of the observed deviations.

In addition to the exploited Hypervisor, VMM level rootkits also affect the performance of the Hypervisor. These rootkits are discussed in the next section.

2.3.2.2 VMM level rootkit

Rootkits are designed to run at ring-1 in such a manner that they exploit hardware virtualization features and intercepts hardware calls made by the original Operating System. This kind of rootkits can be loaded on an Operating System before promoting it into a virtual machine. A Hypervisor rootkit does not have to make any modifications to the kernel of the target to subvert it. Rootkit at VMM level can be considered as a very important security parameter to be handled as it is very difficult to detect its presence in the system.

SubVirt [68] and Blue Pill [69] are two main malware designed for VMM. Blue Pill provides a thin malicious Hypervisor layer between VMs and legitimate VMM and, with its privileged rights, traps each of the requests for the malicious means. Subvirt gets more control over the system by positioning malware at the VM level, and accordingly, it creates a VM based rootkit (VMBR). It allows the malicious services to run on an OS which is protected from the target system.

Area of handling Hypervisor level rootkits, specifically undetected rootkits, requires more exploration, as the rootkits keep themselves undetectable. However, as per Barbosa [70], the Hypervisor does not have full control over Translation Look-aside Buffer (TLB), Branch Prediction, Counter-based Clock, and #GP exceptions using which the rootkits can be detected.

2.3.3 Virtual Machine Security

In a Cloud Computing environment, VMs operated by the same host system are known as co-resident or co-hosted virtual machines. Although co-resident VMs are logically isolated from one another by Hypervisor, physically, they use a common pool of resources. Ristenpart et al. [71] have proposed a technique to place an attacker VM on the same physical system where the victim VM is running. By deliberately making malicious VM co-resident to the victim VM, the attacker opens the door to Outsider attacks.

Another way by which attackers enter the network is via the services that are provided to cloud customers in the form of Virtual Machines. VMs host various services, and if these services are compromised, these compromised VMs attack other co-hosted VMs resulting in Inter-VM or Cross-VM attack. Such Cross-VM attacks are possible even in the presence of isolation.

In addition to Outsider attack and Cross-VM attacks, a VM can be compromised during VM migration also. A brief overview of all such types of attacks to VM is given in this section.

2.3.3.1 Outsider Attack

Allowing a VM to make changes in the existing system is very critical from the security perspective, but some applications require users to be given such rights. The users run malicious applications to compromise VMs and enter in the system to access private information. Although the concept of captive account may be introduced to restrict user rights [72], there are so many other approaches like worms, viruses, backdoors, etc. that attack on VM even without subverting the assigned rights.

Some of the existing approaches used for handling security attacks on VM have suggested the Intrusion Detection System (IDS) / Intrusion Prevention System (IPS) or firewall-based solutions. The IDS / IPS is positioned on a privileged VM, VMM or host system. F. Zhao et al. [73] and K. Kourai et al. [74] have suggested approaches where a dedicated privileged VM is established to monitor Cross-VM communication. Privileged VM is capable of observing all the traffic traversed through the network. It validates the traffic based on a maintained rule list, and the existence of the attacks is identified by tracking the logs. In a firewall-based approach, a privileged domain firewall has been proposed by A. Srivastava et al. [75], which maintains a white-listed process list to check the connection validity for each request.

Traffic monitoring with a privileged VM increases the cost of the entire setup. Instead, if it can be done with the Hypervisor, then extra costing of additional VM can be reduced. U. Tupakula et al. [76], Y. Zhang et al. [77], H. Jin[78], and A. Joshi et al. [79] have proposed some Hypervisor centric approaches where attack detection or prevention mechanisms are operated from the Hypervisor level. Detection of malicious activities is carried out by various approaches like traffic monitoring at fine-granular level, file integrity testing, status checking by vulnerability specific predicates, and guest view casting based reconstruction. The above-defined VMM based mitigation approaches employ VM's internal semantics.

The security checking mechanism has been placed on the host devices as per the approaches suggested by G. Dunlap et al. [80] and T. Garfinkel et al. [81]. The method suggested by G. Dunlap et al.[80] leverages VM logging to analyze intrusion. The suggested method can restore the VM state after being attacked by replaying the stored

sequence of previously executed applications. T. Garfinkel et al. [81] suggest mitigation techniques by monitoring the hardware state of VM. Here, the monitoring device is separated from the host, and it runs with the host privilege.

A distributed approach is proposed by M. Noura et al. [82] to monitor VM activities with the sandbox approach and Host Level-Guest Level Security Analyzer. In another distributed approach by Bryhasan D. Payne et al. [83], the kernel-based hook is provided to trap malware, and it is diverted to a dedicated VM for detection.

Jiang X et al. [84] have suggested an out-of-the-box approach, using the guest view casting technique to reconstruct the internal semantic view of VM from outside in a non-recursive way. The proposed method checks the authenticity with two techniques, view comparison based malware detection and host-based anti-malware software with improved detection accuracy.

D. Patidar et al. [85] have suggested an approach working on a para-virtualized layer to target Distributed Denial of Service (DDoS) attacks. Here, they propose a Hierarchical Secure Paravirtualized System Model with an additional PV-Basement below the virtualization layer. It classifies and handles VMs of similar types from the views of applications with VM-Shadow and Process Detector. This method takes comparatively high space overhead.

Security attack on VM from the outside elements is the most common threat, and hence a considerable amount of mitigation approaches have been suggested to handle them as discussed above. At the same time, security issues to be considered during VM migration must not be ignored for smooth and secure VM migration. The related work is discussed in the next subsection.

2.3.3.2 Attacks during VM Migration

Virtual Machine Migration is a process of shifting VM from one physical machine to the other physical system. The migration facilitates workload balancing, fault tolerance, online system maintenance, and consolidation of virtual machines. During the migration process, VMs are suspended or configured to run without interruption.

This is called Live Migration. Although Live Migration helps in maintaining system state, vulnerabilities associated with that pose many security threats. Trusted source-destination, authentication mechanism, confidentiality, and integrity are the issues that must be considered for secure migration [86]. Mechanisms employed for live migration needs to consider all these issues to protect state transition.

Attacks like Unprotected Transmission Control, Inappropriate Access Control Policies, and Migration Module Loopholes possible during VM migration are analyzed by Kadam et al. [86] along with the discussion existing security mechanisms. In another approach, Aiash et al. [87] have discussed some of the supporting parameters for live migration like the CoM framework, Virtual Trusted Platform, Live Migration Defense framework, and Role-based policies.

Wang et al. [88] and Sammy et al. [89] have also proposed a solution for handling secure migration. The approach by Wang et al. [88] secures live migration of VMs with a policy-based approach where with Attestation Service and Sealed Storage, migration of VM is protected with cryptography. In another approach for live migration, energy-aware provisioning of Cloud Computing resources in virtualized platforms is proposed by Sammy et al. [89], which suggests Dynamic Round-Robin algorithm as a very feasible approach for TOCTTOU, VM Resumption Ordering and Replay attacks. Both approaches increase overhead as the frequency of VM migration increases.

During migration, the amount of data plays an important role in deciding the migration speed. Hence, in an approach by Jeincy [90], the transmission of data in compressed form is suggested for faster migration.

As mentioned before, an overloaded host leads to a data breach. Hence, when a VM is likely to increase the overhead, it is migrated to another server before the system gets saturated. Reeba [91] has proposed such a solution where the future predictions of resource utilization are carried out to presume the need for VM migration for workload balancing.

Oberheide et al. [92] have discussed security issues faced during live migration in context to defined three planes, namely control, data, and migration module. They also present a tool to automate VM's memory and discusses strategies to address deficiencies in VM design. The approach discusses security issues of only Xen and VMWare VMMs.

Live migration raises security issues mainly as the VM can be attacked during the migration process, and hence, it has been a topic of interest for many researchers, as discussed above. The next and last type of VM attack is a Cross-VM attack, where a compromised VM launches an attack on other VM working on the same host. A detailed discussion of the Cross-VM attack is carried out in the next subsection.

2.3.3.3 Cross-VM Attack

Isolation is a property that prevents Virtual Machines from accessing private resources of the other Virtual Machine. When Isolation is compromised, the virtualization layer breaks down. Although VMs are isolated by logically separating the set of resources dedicated to them, there might be some cases where they need information sharing. Operations requiring data exchange must be allowed only after the assurance that the monitoring is with specific configuration and within the allowed limit. Failure in authentication while accessing the shared resource leads to an Inter-VM attack.

Some of the approaches propose Inter-VM (Cross-VM) Communication Mechanisms [93]–[99], which provide a direct path between communicating VMs and eliminates the limitations of common sharing points. Although they increase security with a dedicated path between a pair of VMs, they do not eliminate the chances of an Inter-VM attack with side channels. C. Gebhardt et. Al. [100] and J. Wang et al. [101] discuss the features of each of the Inter-VM attacks in detail.

Approaches are proposed in the direction of increasing the efficiency and security of Inter-VM data transfer operations. [102]–[107] have explored ways to improvise the data transfer mechanism between VMs. Fan et al. [108] have worked in the same direction and have proposed a Hypervisor-based Secure Communication System,

where using ISCP communication protocol, data is encrypted while getting transferred from one Xen domain to other via VMM.

Despite the available high-performance communication mechanisms, there are some issues like Side-Channel Attack, which give rise to Cross-VM security bottlenecks.

The pool of resources shared by the co-hosted VMs raises many vulnerability issues; one of them is the covert channel. A covert channel is a method of communication that is used to illicitly transfer information via shared resources, thus breaking the security policy of the system [109]. Side-Channel Attack (SCA) is one such attack that exploits the shared physical resources like CPU cache, a memory bus, disk bus, Branch Target Buffer (BTB) and Network Queue. Co-resident virtual machines use a covert channel to leak confidential information [71]. VM with malicious process utilizes shared data to extract some confidential information like the encryption key of the other co-resident target VM. This is known as a Side-Channel Attack (SCA). SCA steals secure information only by using measurement and monitoring of various parameters like execution time, consumed power and electromagnetic radiation without actually attacking the target VM. Hence, Side-Channel Attack can be classified as the most complex attack to launch and to mitigate among all the probable Cross-VM attacks.

Literature survey of all the cloud security issues reveals that the majority of the attack procedures employ explicitly looking malicious actions for launching the attack. They can be detected by imposing effective encryption mechanisms or by positioning IDS/IPS or firewall. On the opposite front, attack launched merely by profiling makes SCA very difficult to be detected. Thus, a unique attack launching method and challenge in detection make the SCA very interesting from the research point of view. Considering the significance of SCA, an in-depth analysis of approaches handling SCA is carried out, which is discussed in the next section.

2.4 Side-Channel Attack

As discussed in the previous section, exploitation of the shared resources may result in the Side-Channel Attack, as shown in Figure 2.2. Implementation of Side-Channel Attack on

the multi-core system was claimed first by Zhang et al. [110]. He has proposed an SCA that can work at the fine-grained level and supports core migration. They have implemented the attack that can successfully extract decryption keys from the victim process.

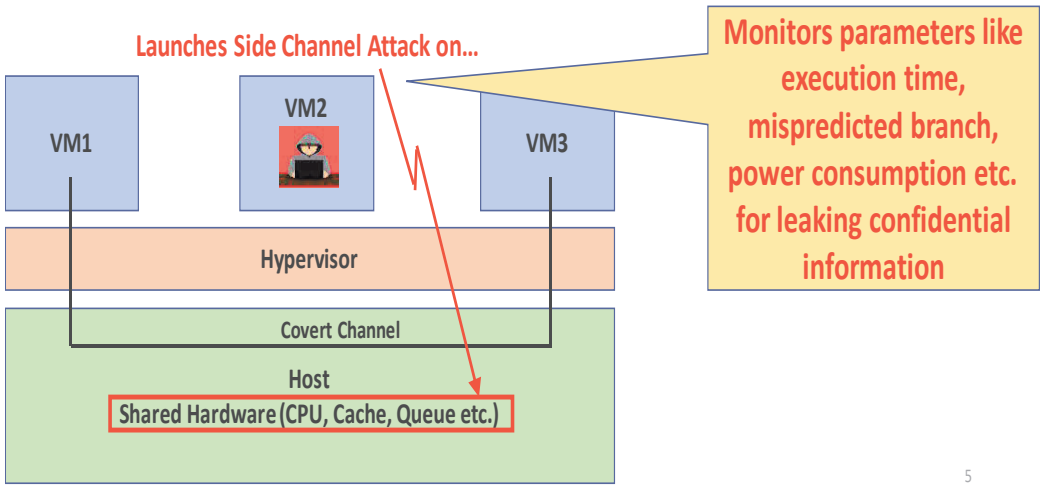


Figure 2.2: Side-Channel Attack Scenario

Side-Channel Attack has been focused on the consideration of the AES (Advanced Encryption Standard) algorithm for encryption in the majority of the research work. AES has a large number of static values to be used during encryption, which is common for all the processes which use the same standard for cryptography. The frequent need of AES S-Box and other such static values keep them stored in the shared cache memory, which increases the scope of stealing secret key bits by malicious co-resident VM.

The analysis of the work done for handling SCA reveals that Side-Channel Attack on AES loaded in the cache memory is the focus of the majority of the research work. For removing AES specific issues, Sevak et al. [111] suggested following a confusion-diffusion approach for randomly selecting an encryption algorithm among DES (Data Encryption Standard), AES, and DES3. SCA was launched through the covert channel, which is the main media that provides a path between a pair of isolated VMs. Yunjing Xu et al. [112] has discussed approach to improve the efficiency of L2 cache covert channel for preventing private key leakage. Similar approaches include mitigation methods of the cache timing attack proposed by Percival [113] and Bernstein [114]. Page [115] proposed a partitioned cache architecture, and Yu et al. [116] proposed a cache-based SCA detection approach that employs observation of resource utilization. For preventing an attacker from retrieving confidential

information, approaches like the addition of noise to cause delay and randomization while accessing memory are suggested by B. Mughal et al. [117]. One more approach in the area of cache attacks on AES has been explored very efficiently by D. Osvik et al. [118] that can extract the entire key. This is unlike other approaches where part of the key is known, and remaining key bits are extracted with time measurement. Here the entire key can be extracted without any knowledge about the corresponding plaintext or ciphertext. With Evict + Time and Prime + Probe methods, it efficiently detects the attacks and also discusses possible countermeasures. Another approach detecting the presence of cache-based SCA on AES was proposed by Y. Kulah et al. [119] that includes the detection of prime-and-probe attack, flush-and-reload attack, ECDSA, and Flush+Flush attack.

Although the above approaches have proposed efficient mechanisms to mitigate this attack, the current implementation of AES removes the scope of extraction of encryption key bits for an AES algorithm. Unlike the initial approach of AES implementation, the static table values are integrated as the part of the algorithm instead of storing them on cache, i.e., the values are not shared among different programs implementing AES. According to this design, there will be no information present on cache from which the key bits can be extracted and so the effect of SCA for the AES algorithm is also nullified.

The study of SCA on other architectural components reveals that exploitation of the components of the Branch Prediction Unit (BPU) leads to Branch Prediction Analysis (BPA) attack. BPA attack can extract the private key of the asymmetric cryptographic processes like RSA (Rivest-Shamir-Adleman) and ECC (Elliptical Curve Cryptography). Moreover, limited work done in handling the BPA attacks makes it interesting and essential from the research point of view.

2.5 Summary

The large surface area of cloud architecture requires security consideration from different perspectives. The study of existing research discussing various cloud security issues revealed that the researchers [7]–[13] have focused on the generic cloud security issues without considering virtualization. At the same time, virtualization was the main focus of many others [14]–[20]. From the analysis of the existing work surveying various cloud security issues, we

identified a need to present a taxonomy including all the generic as well as virtualization-related cloud security issues along with the present approaches handling them.

Accordingly, a taxonomy was proposed with an aim to identify the research gap in the area of cloud security. The presented taxonomy is shown in Figure 2.1. As shown in the diagram, our taxonomy gives more weightage to virtualization layer security issues without ignoring other essential security issues like issues related to the multi-tenant environment, service delivery models, and third party CSP. Approaches are handling security issues of a multi-tenant environment, and data security mainly targets issues like data authentication, authorization, integrity, and privacy [22], [23], [32]–[35], [24]–[31]. As per them, data on a central data storage can be kept secure by applying various data encryption methods. In the majority of the cases, the scheme of central storage works also provided complex encryption algorithms are used along with concrete backup and restore techniques. Apart from the data security, security issues at SaaS, PaaS, and IaaS are also the topics of interest of many researchers. The present work done in that area reveals that mitigation approaches for SaaS, PaaS, and IaaS cover the majority of the related security aspects [36]–[44].

The virtualization layer security [5], [47]–[51] was classified into VM security, Hypervisor (VMM) security, and host security. The exploitation of hosts is very challenging in the virtualization environment, but there are some attacks like VM Escape (Guest-to-Host attack) and host level rootkits, that can damage the host. Although the host is vulnerable to security attacks, there are some approaches to mitigate the rootkits [57]–[65], Additionally, it is also not that easy to launch the VM Escape attack, as discussed in [52]–[56], which reveals that there are ways to secure the host machine. As the Hypervisor provides an intermediate layer between a host and a set of VMs, a secure Hypervisor also increases the security of the host system.

The Hypervisor provides isolation among VMs. Although the end-user does not directly attack Hypervisor, it can be exploited either through VM or by launching a dummy Hypervisor/rootkit. The security of Hypervisor is very important as it is the core of the virtualization. Present mitigation approaches suggest effective solutions to prevent Hypervisor [66]–[70] from all such attacks. However, the handling of undetected rootkits is very challenging as they keep their identity hidden from the Operating System.

Analysis of the last category, i.e., VM security, includes Outsider as well as Cross-VM attacks [71]. Additionally, secure VM migration is also equally important while considering VM security. Various VMM-based or dedicated-VM based IDSs/IPSs are suggested [72], [82]–[85], [74]–[81] that authorizes each of the traversing packet to prevent the system from Outsider attack. Points to be considered for secure migration were discussed in [86]–[92]. However, Cross-V?M attacks, especially Side-Channel Attacks, still require more focus considering its severity.

Security loss during Cross-VM communication can be prevented by providing a dedicated path or imposing packet encryption [93], [94], [103]–[108], [95]–[102]. The major vulnerability in the Cross-VM case lies in simultaneous access to architectural resources like CPU cache, network queue, a memory bus, and BPU. Analysis of only performance parameters of these shared resources to extract private key bits is the main idea behind the SCA, which makes it very difficult to be detected. Thus, a unique attack launching method and challenge in detection make the SCA very interesting from the research point of view.

SCA extracts the private key of asymmetric cryptographic algorithms like RSA and ECC. Analysis of the work done for handling SCA reveals that the majority of the proposed work targets SCA on cache memory [110]–[119]. However, consideration of other shared resources is equally essential for tightening security.

Branch Prediction Analysis (BPA) attack is one such Side-Channel Attack where components of BPU like Branch Predictor and Branch Target Buffer (BTB) are exploited to extract the private key. The researcher found the impact of BPU sharing very interesting and essential to be explored more as minimal work is done in handling the Branch Prediction Analysis (BPA) attack. It motivated us to work on the BPA attack in our research work. A detailed discussion on the BPA attack along with its various attack launching methodologies and its scope in the virtualization environment is carried out in Chapter 3.

CHAPTER

3 Branch Prediction Analysis Attack: A Side-Channel Attack

Side-Channel Attack launched by exploiting the shared components of the Branch Prediction Unit (BPU) results in the Branch Prediction Analysis attack [120]. The attack is launched in four different ways, where one of the methods predicts the private key bits of the target asymmetric cryptographic process like Rivest-Shamir-Adleman (RSA) and Elliptical Curve Cryptography (ECC) by exploiting a part of BPU, i.e., branch predictor. In the other three methods, another component of BPU, Branch Target Buffer (BTB) is exploited for extracting the key bits. Considering the significance of the branch predictor and BTB in launching the BPA attack, background details discussing the functionality of these components are provided in Section 3.1 of this chapter. Although both RSA and ECC are vulnerable to the BPA attack, the RSA process is considered for this research work. A brief discussion on RSA is carried out in Sections 3.2. Further, an explanation of each of BPA attack launching methods is presented in Section 3.3. As discussed in Section 3.4, the normal-looking attack launching process makes the detection of the BPA attack very difficult, which has motivated us to work in that area. Finally, the study of existing approaches handling BPA attack and summary of the chapter are provided in Section 3.5 and Section 3.6, respectively.

3.1 Introduction

In the Branch Prediction Analysis attack, the prediction of the conditional branch instruction is the core action. It is quite important and necessary to get the background details of how the conditional branch instruction works. Execution flow for a conditional branch instruction depends on whether the branch is taken or not. The presence of such conditional branch instruction may require to clear the entire instruction queue resulting in a delay in

accomplishing the task. In such circumstances, a branch predictor, a part of BPU, can perform a very significant task of predicting the execution status of the conditional branch instruction. With an analysis of past behavior of the conditional branch instruction, the branch predictor predicts whether the branch would be taken or not during the next execution. Correctly predicted execution status decreases the delay caused because of the clearance of the instruction queue. The event when the prediction about the execution status of a conditional branch instruction differs from that of the actual execution status is known as a 'branch mispredicted' event.

Branch Target Buffer (BTB) is another component of BPU that stores the target addresses of the branch instructions once the branch is accessed. As BTB access is faster than the memory access, the execution time of a branch instruction reduces if its target address is available in the BTB.

The above discussion reveals that the branch predictor and BTB are two significant components reducing the execution time of a conditional branch instruction. Side-Channel Attack like Branch Prediction Analysis attack takes the benefit of the efficiency of the branch predictor or BTB to extract the logical value of the condition for the victim branch instruction. Cryptographic algorithms like Rivest-Shamir-Adleman (RSA) and Elliptical Curve Cryptography (ECC) are vulnerable to BPA attack. Both RSA and ECC are asymmetric cryptographic algorithms, and the BPA attack is launched to extract the private decryption key of these algorithms.

RSA algorithm is considered for this work, and a brief explanation of the RSA algorithm is given in section 3.2.

3.2 BPA Attack Vulnerable Algorithm: RSA

RSA is an asymmetric cryptography algorithm that is used to generate a pair of public and private keys. As the first step, it selects two prime numbers, P and Q. Modulus value is calculated as P*Q. A small exponent e is chosen to be an integer such that it is not a factor of n. At the same time, $1 < e < \varphi(n)$, where $\varphi(n)$,==(P-1)(Q-1). With public-key (n,e), a plaintext M is converted into ciphertext C as $C = M^e \bmod n$.

For composing private key, an element d is found such that e*d=1 mod n. Key pair (d,n) makes the private key where $M=C^d$ mod n.

Modular exponentiation required during the encryption and decryption is commonly implemented using the Square and Multiplication (S&M) algorithm. During the squaring operation, a conditional multiplication instruction (step 4 Figure 3.1) is executed only when the corresponding key bit value is one.

```
Input:
C: Ciphertest
d: Private Key
n: Module
SqandMul (C, d,n)
1       M' ← 1
2       For i=1 to keylength
3               M'=MontMul(M',M',n);
4               If d_i = 1 then
5                       M' = MontMul(M',C,n)
```

Figure 3.1: Square & Multiply Algorithm

The exponential multiplication in the S&M algorithm is executed by the Montgomery Multiplication algorithm shown in Figure 3.2.

```
MontMul (A,B,n)
1       A' ← AR mod n
2       B' ← BR mod n
3       S=A'B' + (((A'B'N')mod R) N)/R
4       If S >= n
5               S=S-N
6       End
7       Return S
```

Figure 3.2: Montgomery Multiplication Algorithm

The extra reduction step (step 4 of Figure 3.2) of this algorithm is dependent on the input and the private key bit values. This step is the target branch instruction for two of the four different attack launching methods, Direct Timing Attack, and Asynchronous BTB Eviction Attack. For the other two attack methods, the conditional branch instruction of S&M algorithm (step 4 Figure 3.1) is the target instruction. A detailed discussion on BPA attack mechanisms is carried out in the next section.

3.3 BPA attack Mechanisms Introduction

In the original contribution by Onur et al. [120], four different methods were suggested to launch a BPA attack where the algorithm under consideration was RSA. A brief discussion on the possible BPA attack methodologies is provided below:

3.3.1 Direct Timing Attack

The Direct Timing Attack method leverages the deterministic behavior of the prediction algorithm. For extracting secret bits of private key d during RSA decryption, it is assumed that the first i bits are already known to the adversary, and he is trying to extract the $i+1^{th}$ bits.

For simulation, a large set of ciphertext messages M (around 1000) is randomly selected for decryption. The entire simulation process is divided into two phases, namely Offline and Online phases.

- **Offline Phase**

In the initial Offline phase, known bits of d (d_0 to d_{i-1}) are traced for a message m in M. During the trace, it is found whether the underlying conditional instruction results in branch or not, i.e., branch is taken or not. The trace makes a sequence T as (T_0,T_1,…,T_{i-}

$_1$) where T_j (j=0 to i-1) is 0 or 1, depending on whether the branch is taken or not as per the corresponding bit d_j.

Further, for predicting next bit d_i, it is checked whether the conditional branch instruction will be executed or not for both the possible values 0 and 1. Let the obtained results be represented as A_0 and A_1. At the same time, the generated trace T is fed to the dynamic branch predictor to predict whether the branch will be taken or not for both the possible values 0 and 1 for bit d_i, and they are denoted as Pr_0 and Pr_1, respectively. Outputs of the predictor, Pr_0 and Pr_1 are compared with A_0 and A_1, respectively. The result of comparison falls into four possible cases due to the misprediction event during the Montgomery Multiplication (MM) at $(i+1)^{th}$ squaring. The four cases are mapped to four sets: M1, M2, M3, and M4.

M1 = {m / m does not cause a misprediction during MM of $(i+1)^{th}$ squaring if $d_i=1$}
M2 = {m / m causes a misprediction during MM of $(i+1)^{th}$ squaring if $d_i=1$}
M3 = {m / m does not cause a misprediction during MM of $(i+1)^{th}$ squaring if $d_i=0$}
M4 = {m / m causes a misprediction during MM of $(i+1)^{th}$ squaring if $d_i=0$}

3-1

The above process is repeated for each message m in M. As an outcome of the Offline phase, let there be S_i number of messages in M_i.

- **Online Phase**

During the Online phase, for each message m_{ij} (j=1 to S_i) in set M_i (i=1 to 4), the number of branch misses $BM(m_{ij})$ is calculated while decrypting m_{ij} for the entire secret key. At last, the average of all $BM(m_{ij})$, known as $avg(BM(M_i))$ is evaluated as follows:

$$avg(BM(M_i)) = \sum_{j=1}^{S_i} BM(m_{ij}),\ 1 \le i \le 4$$

Prediction of the next bit value di is carried out based on the following equation :

$$if\ avg(BM(M_1)) > avg(BM(M_2)\ \text{and}$$

$$avg\big(BM(M_3)\big) < avg(BM(M_4))\ \text{then}$$

$$di = 1$$

$$else$$

$$di = 0$$

3-2

The above steps of the Online phase are repeated for extracting the next secret bit of the decryption key until all the bits are extracted [120].

3.3.2 Asynchronous BTB Eviction Attack

For a cryptographic process (here, RSA) running on a Simultaneous Multithreading (SMT) architecture, the adversary executes a concurrent dummy process. The dummy process is designed to evict the entries of BTB continuously by executing a large set of conditional branch instructions. Hence, whenever the conditional branch instruction of the cryptographic algorithm is taken, the required target address will not be found in the BTB, and the branch-miss event is generated. There is a high possibility that a branch with a target address in BTB would have been accessed in the near past. The absence of the target address in BTB may lead the predictor to predict the branch as not taken. Accordingly, as the dummy process continuously evicts entries from the BTB, whenever the branch is to be taken, the misprediction event is generated.

Similar to the DTA method, simulation of exponentiations is carried out where the partitioning is done based on whether the target branch is taken during the computation of m^2 (mod N) or not (step 3 of Figure 3.1).

Clearance of BTB takes either of the three forms:
Total Eviction : Entire BTB is cleared
Partial Eviction : Part of BTB storing target address of the target branch instruction is evicted

Single Eviction : Single BTB entry storing the target address of the target branch is evicted

Among these three eviction approaches, the total eviction is easy to implement, where the other two eviction methods can improve the success ratio of the attack. To implement partial eviction, identifying the small group of BTB entries storing the target address is possible. Even the single BTB entry storing the target address can also be identified as Kerckhoff's Law. Dummy process can predict the secret bits without any need to be synchronous to the cipher process, and so the attack is known as Asynchronous attack [120].

3.3.3 Synchronous BTB Eviction Attack

Synchronism is the main issue that differentiates the dummy process of the Synchronous and Asynchronous BTB Eviction attacks. The dummy process evicting the BTB entries requires synchronism with and the cryptographic process. If the dummy process can keep pace with the cryptographic process by establishing synchronism, it clears the BTB just before the execution of the conditional branch instruction of the S&M algorithm.

For a bit i, if the branch is taken, then an event of misprediction is generated. Misprediction event results in an increase in the execution time for that iteration compare to other iterations where the branch is not taken. Accordingly, the increase in the execution time reflects the presence of bit 1, corresponding to that position in the decryption key. In this way, by clearing single BTB entry just before the execution of the conditional branch instruction of i^{th} iteration, bit value d_i is predicted, and the entire key is generated accordingly.

3.3.4 Trace-Driven Attack

A slightly different approach to predict secret bits is employed as Trace-Driven Attack (Time-Driven Attack). A spy process with a number of conditional branch instructions equal to the BTB set is started before the execution of the cipher gets initiated. The spy process continuously fills the area of BTB, which is used by the cipher to store the target address of the conditional instruction. Hence, when the cipher tries to retrieve the target

address of the target branch from BTB, there will be a branch miss as the entire BTB set is filled with the instructions of the spy process. At this time, one of the spy process instructions is required to be evicted from the BTB to make room for the conditional branch instruction of the RSA process. The spy process continuously measures the execution time. When the spy process tries to access the branch instruction that was evicted, it will require more time than usual as it is not present in BTB. The spy process observes an increase in the execution time at this moment. As the increase in the execution time is mainly due to the execution of the conditional branch instruction of the RSA process, implicitly, it reflects the presence of bit 1 for that position in the private key.

As per the above-explained method, every event with more execution time for the spy process is interpreted as the presence of a bit with value 1 in the key. The remaining positions are considered to be zero. Thus, by continuously measuring the execution time of the spy process, the adversary can extract the entire series of secret key bits.

For filling the BTB by conditional branch instructions, the spy process can apply either of the following approaches:

Total Occupancy: Entire BTB is filled

Partial Occupancy: Part of BTB where RSA instructions are associated, is filled.

3.4 BPA Attack Detection Difficulty

The above study highlights a very important characteristic of BPA attack launching procedures. The extraction of the private key is carried out merely by observing an appropriate performance parameter. Parameters like the total number of branch misses and execution time are observed by various attack launching methods to predict the bit. Accordingly, the common action that is found in each of the four attack launching methods is profiling, i.e., monitoring of performance parameters. Profiling looks like a normal action, and so it is very difficult to detect the presence of such an attack, which is launched only by profiling. No IDS/IPS or firewall can identify the profiling action as malicious. Moreover, profiling cannot be prohibited, also looking at its usage in analyzing the system performance. In this regard, it is very difficult to detect the presence of the BPA attack. There are some researchers who have taken this challenge and have contributed in the

direction of handling the BPA attack. A discussion on existing approaches handling BPA attack is carried out in the next section.

3.5 Existing Solutions

Cryptographic algorithms like RSA and ECC use the Square and Multiply (S&M) algorithm to perform modular exponentiation, as discussed in Section 3.2. Execution of the conditional branch instruction in the Square and Multiply algorithm (step 4 in Figure 3.1) depends on the respective secret key bits. As per [120] [121], time difference observed during the execution of the S&M process corresponding to the cases when the bit value is 0 and 1, is the key parameter to predict the value of the bits. This time difference vanishes if a conditional branch instruction is executed irrespective of the bit value. As per this theory, a replacement of the Square & Multiply algorithm by Montgomery Ladder Algorithm [122] (as shown in Figure 3.3) was suggested for the OpenSSL library. The balanced branch instruction in the Ladder algorithm does not exhibit execution time difference, whether the bit is 0 or 1, and thus, it eliminates the scope of the BPA attack [123].

```
Input:
C: Ciphertest
d: Private Key
n: Modulo
Mont_Ladder (C, d,n)
        R0 ← 1
        R1 ← C
        For i from 0 to n-1 do
            if di=0 then
                R1 ← (R0 * R1) mod n
                R0 ← (R0 * R0) mod n
            End
            Else if di=1 then
                R0 ← (R0 * R1) mod n
                R1 ← (R1 * R1) mod n
            End
        End
        Return R0
End
```

Figure 3.3: Montgomery Ladder Algorithm[122]

We need to modify each vulnerable library for the effective implementation of this solution. However, even if the Ladder algorithm is used for RSA implementation, a BPA attack is still possible as per S. Bhattacharya et al. in [124][125] where observation of the number of branch misses is carried out in place of CPU clock cycles. Additionally, S. Bhattacharya et al. [126] have also shown that even for RSA with Chinese Remainder Theorem (CRT) implementation, the BPA attack is possible. Elimination of all the points against the possibility of the BPA attack keeps the problem live.

Many researchers have proposed approaches to handle BPA attack. Among them, Agosta et al. [127] suggested to either eliminate or to replace the conditional branch instruction from the vulnerable cryptographic algorithms with indirect branch instructions. In another solution, Ya Tan et al. [128] suggested a mechanism for locking some of the BTB entries of the processes, which prevents the spy process from filling the entire BTB with branch instructions of the spy process. The spy process fails to predict the bits as it cannot keep pace with the execution flow by filling all the entries of the BTB. They also state that the attack is possible even with RSA blinding.

Julien et al. [129] have proposed a blacklisting approach that allows only white-listed processes to access the hardware counters. Prohibiting the access of performance parameters prevents the launching of a BPA attack as it needs to read CPU time or the total number of branch misses. In a mitigation technique for BPA with Direct Timing Attack (DTA), S. Bhattacharya et al. [130] have proposed an approach to manipulate dynamic predictors. They have suggested executing a randomization module in concurrence to the compromised process. The execution of the randomization module alters the state of BPU because of which the simultaneously running compromised process loses its correlation with the key. The suggested approach prevents of BPA attack launched with the DTA method.

Analysis of the approaches handling BPA attack reveals that the majority of the approaches manipulate the functioning of the components like BTB, Branch Predictors or Performance Counters. Manipulation of the architectural components may affect the performance of some of the legitimate processes.

The study of the existing work done in the area of BPA attack and their limitations has led us to define the problem statement, as discussed in Section 4.

3.6 Summary

The attack launched by exploiting the shared resource is known as the Side-Channel Attack. When the exploited shared resource is either of the BPU components, such SCA is specifically identified as the Branch Prediction Analysis (BPA) attack [120], [121]. The asymmetric cryptographic processes like RSA and ECC are the common victims of this attack. The private key bits of these processes can be predicted by launching the BPA attack using any of the four different attack launching methods. The attack detection approach of all methods differs from one another in a broader form. However, the core action behind the prediction of the private key is the same in all four ways. The private key bits are predicted merely by carrying out profiling, i.e., reading the values of different performance parameters like execution time and the number of misprediction conditional branch instructions. For launching the BPA attack, a genuine act of profiling is misused to extract the confidential information. Valid action for invalid means makes the detection of BPA attack a challenging task.

As a reflection of the detection difficulties of BPA Attack, minimal work is done to handle the attack. Execution time difference shown by the S&M algorithm (used by RSA and ECC) based on the bit value 0 or 1, is the key parameter to predict the key bit. An approach was suggested to eliminate the scope of the attack by replacing the vulnerable S&M algorithm by the Ladder algorithm [122] having a balanced branch. However, the claim was also violated by S. Bhattacharya [124][125][126], where the possibility of a BPA attack is shown even in the presence of a balanced branch Ladder algorithm and CRT. Among the limited number of approaches that efficiently detect/prevent BPA attack, some of the approaches affect the normal system performance by manipulating any architectural component [128][129][130] and one approach is specific to the target cryptographic algorithm [127]. It suggests a requirement of an approach to detect the BPA attack that overcomes the limitations of the present approaches.

The BPA attack requires consideration of virtualization as the attack environment as per the current technology. In this regard, its applicability in the virtualization environment needs to be assessed, where it is simply stated in [120] that BPA attack can be launched in the presence of virtualization and also sandboxing without providing more details. Keeping all the above points together, the proposed problem definition of our research work is presented in Chapter 4.

CHAPTER

4 Problem Definition and Scope of the Work

4.1 Introduction

BPA attack can be launched by four different methods, as discussed in Chapter 3. The victim process and the spy process are the two primary elements of all the four attack methods. In the original proposal of the BPA attack by Onur et al. [120], a non-virtualization environment was considered, where they have suggested that the attack is possible even in the presence of virtualization and sandboxing.

In a non-virtualization environment, both the victim and spy processes are considered to be present on the same machine [120]. Whereas, both the processes will not be on the same VM considering the one VM-one service kind of typical setup in the virtualization environment. Moreover, the hardware resources, like CPU core, memory, system bus, as well as some software resources, may be shared or dedicated as per the underlying VM configuration. VM sharing configuration becomes a very important parameter to be considered to assess the scope of the BPA attack in virtualization.

In [120], a possibility of a BPA attack in virtualization is shown without providing any more details regarding the same. They have not provided any information regarding the practical issues raised while launching a BPA attack in the virtualization environment. Additionally, the solution approaches [127], [128], [129], and [130] also lack in giving useful information about attack simulation in a virtualization environment. Thus, we found that a detailed study is needed to explore the scope of the BPA attack in the virtualization environment.

As discussed previously, consideration of virtualization requires consideration of underlying resource sharing configuration in VMs for exploring the scope of BPA attack. In turn, a study of existing BPA attack handling approaches should also be carried out from a different perspective to assess their applicability in the virtualization environment.

Analysis of existing BPA attack handling approaches carried out in section 3.5, poses a necessity to overcome their limitations. We also find a need to work out robust, accurate, and independent solutions to handle the BPA attack in the virtualization environment. With reference to the above discussion, we list out the objectives and scope of our work in section 4.2

4.2 Objectives and Scope of the Work

We define the objective and scope of our work as:

- Assess the scope of the BPA attack in the virtualization environment
  Four different BPA attack methodologies follow different procedures to predict the private key bits. Each method has its own requirement of shared resource(s) between the victim and spy processes for launching the BPA attack. As per the typical scenario of virtualization, both the processes are most likely to be located on separate VMs. As the VMs may or may not share the resources, the scope assessment of the BPA attack in virtualization must be carried out by taking resource sharing configuration into account. (Section 5.1)

- Identify the applicability of existing approaches to handle the BPA attack in the virtualization environment
  The study of the existing approaches reveals that the majority of them consider non-virtualization as the attack environment. Applicability of these approaches in handling BPA attack in virtualization should be thoroughly assessed, especially when both the victim and attacker processes lye on separate VMs. (Section 5.2)

- Simulate BPA attack in the virtualization environments
  As discussed in Section 4.1, the BPA attack is simulated in a non-virtualization environment where the attack methodology under consideration is DTA,

Asynchronous BTB Eviction Attack, and TDA in [125], [120] and [128] [120] respectively. Simulation of various BPA attack launching methods in virtualization needs to be carried out to identify the possibility of a successful BPA attack. Attack simulation can be considered as a prerequisite of carrying out experimental analysis of an attack handling approach (Section 6.1)

- Work out a new solution to overcome the limitations of existing solutions
  As summarized in Section 3.6, there is a need for such a BPA handling approach for virtualization that can work without disturbing the normal system functioning. Additionally, it should be independent of the cryptographic algorithm vulnerable to the BPA attack. Above mentioned fact has led us to identify our primary objective of working out a new solution that overcomes the limitations of existing solutions. (Section 6.2)

- Carry out experiments for proving the effectiveness and accuracy of the proposed solution.
  The proposed approach needs to be implemented for the means of validating its efficiency. Experimental analysis of the proposed approach is required to observe its efficiency in detecting the presence of a BPA attack. Time taken in accomplishing the detection process is also very important in measuring the amount of loss, if any. The fulfillment of this objective is quite significant for proving the effectiveness and accuracy of the proposed approach. (Section 7.1)

- Evaluate the performance of the proposed approach.
  The performance of the proposed approach is required to be evaluated from three angles: (1) By comparing the scope with existing approaches handling the BPA attack (2) By measuring the overhead of the proposed solution. (3) Bt evaluating time taken in detecting the presence of the BPA attack (Section 7.2)

CHAPTER

5 Related Work

The study carried out on the BPA attack in Chapter 3 throws light on two aspects: (1)minimal research work has been carried out to handle (to detect/prevent) BPA attack (2) The related research work, found in [120][121][123], [125]–[130], does not present any analysis for identifying the possibility of the BPA attack in the virtualization environment. Applicability of the existing solutions in virtualization also becomes very important in this regard. The above discussed two aspects are the primary concerns of this chapter.

It has been summarized in chapter 3 that the resource sharing configuration of VMs is the decisive parameter for assessing the applicability of BPA attack in the virtualization environment. Related discussion on this aspect is carried out in Section 5.1. BPA attack launching issues in virtualization with reference to all the four attack launching methods are discussed in this section. Section 5.2 focuses on the second aspect, i.e., all the existing approaches handling BPA attacks are revisited to analyze their scope in the virtualization environment.

5.1 BPA Attack in Virtualization Environment

According to the original contribution of Onur et al. [120], the victim and the adversary processes are located on the same system. It is in accordance with the fact that multiple services are hosted by a single system in a non-virtualization environment. The exploitation of one of the services can affect other services running on the same host in such an environment. Services like Hypertext Transport Protocol Secure (HTTPS), secure Video Conferencing (VC), and Secure File Transfer Protocol (SFTP) can be easily exploited [131] to make a door for the attacker to enter the system. The exploited service can launch an attack on the other service/process running on the same system.

The scenario is different in the virtualization environment, where each machine (VM) is treated as an individual machine having the required resources. One VM typically hosts only a single service in the virtualization environment. Virtualization hides the internal details of the underlying resources from the end-users for the means of providing security. The layer of abstraction provided by virtualization is meant for the purpose of security. However, neither the one VM-one service configuration nor the resource hiding feature of virtualization prevents the VM from being exploited. A VM can get compromised by exploiting the services running on it like HyperText Transfer Protocol Secure (HTTPS), Secure File Transfer Protocol (SFTP), and secure VC.

A compromised service on a VM can launch an attack on the other process that is running on a separate but co-resident VM. Launching of BPA attack from a compromised process to the other co-resident VM may give rise to a Cross-VM BPA attack. The probability of such a Cross-VM BPA attack and possible attack mechanisms depend on how VMs are sharing the hardware Central Processing Unit (CPU) core and software (Cryptographic Library) resources. The dependence of a BPA attack on the shared resources makes it an interesting topic for research. It motivates us to experiment on such a Cross-VM attack. It also leads to the identification of the attack mechanism(s) with which the Cross-VM BPA attack becomes possible.

The VM technology can optimize between sharing and isolation of resources. The security of this environment depends on the configuration adopted by the system administrator for hosting multiple services. There are three ways to configure the VMs : (a) Fully Isolated (b) Fully Shared and (c) Hybrid. The Fully Isolated configuration is similar to that of a dedicated machine. It provides high security but lacks in terms of effective resource utilization. Essential characteristics of Cloud Computing, like load balancing and fault tolerance, also do not get supported. The second type of configuration, i.e., Fully Shared, offers better resource utilization by means of resource sharing compare to the first type., although it lacks in terms of security. More sharing enables better resource utilization, whereas strict isolation provides better security. The Hybrid configuration is one such configuration presenting a defacto standard, and it is applicable for efficient resource management with optimized security.

The type of VM configuration for a system plays a significant role in deciding the system vulnerability to the Cross-VM BPA attack. The possibility of a Cross-VM BPA attack is eliminated if a VM is isolated by allocating dedicated hardware and software resources. However, such a dedicated allocation of resources is practically very challenging. From the experience of the experts working on VMware or Citrix, it has been found very difficult to configure such a VM. The same has been experienced by us while trying to configure a dedicated VM using KVM. This option is rarely preferred by VM administrators considering its implementation issues. In most prevailing options, the virtualization administrator allocates dedicated memory size, CPU frequency, and disk space size as security best practices. Multiple VMs generally share CPU cores in such a scenario.

Sharing characteristics of software resources (here, cryptographic library) is also important in identifying the scope of BPA attack in addition to the hardware resources. The sharing features required for a successful launch of the BPA attack depends on the attack launching methodology.

Among the four attack launching methods, the first method, DTA, extracts unknown key bits by simulating the behavior of the branch predictor. It requires a shared cryptographic library between the spy and victim processes. While considering DTA in the Cross-VM environment, let VM1 and VM2 are holding the victim and spy processes, respectively. As shown in Figure 5.1, if both the VMs have their separate cryptographic library, then the Direct Timing Attack cannot be launched. The Cross-VM DTA from VM2 to VM1 is possible only when the library is shared, as shown in Figure 5.2.

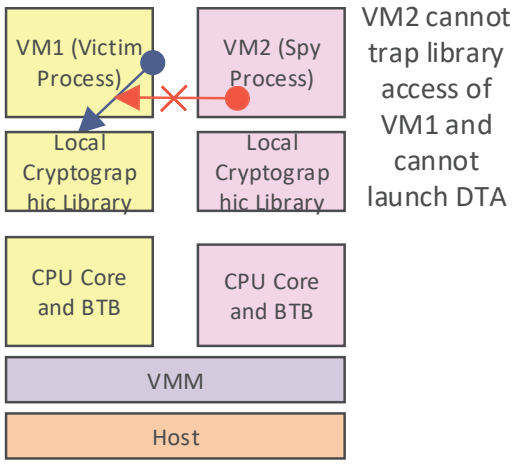


Figure 5.1: VM Configuration with Dedicated Resources

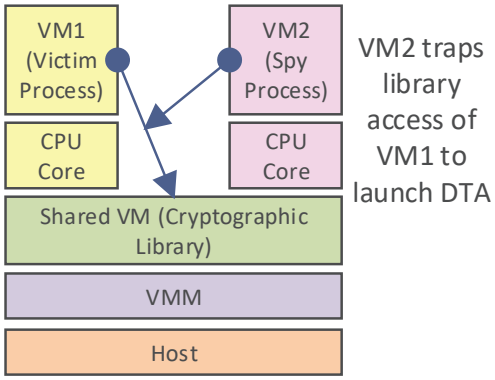


Figure 5.2: VM Configuration with Shared Cryptographic Library

The other three methods launching BPA attack manipulate BTB for tracking the behavior of the victim cryptographic process. BTB is a part of the CPU core, and so, the BPA attack with Asynchronous BTB Eviction method (ABEA), Synchronous BTB Eviction method (SBEA), and Trace-Driven Attack (TDA) is possible only if the same CPU core hosts both the victim and spy VMs. This fact is reflected in Figure 5.3 and Figure 5.4. One more point to be noted is, in addition to BTB, the cryptographic algorithm is also required to be shared for ABEA because of its DTA-like attack launching method.

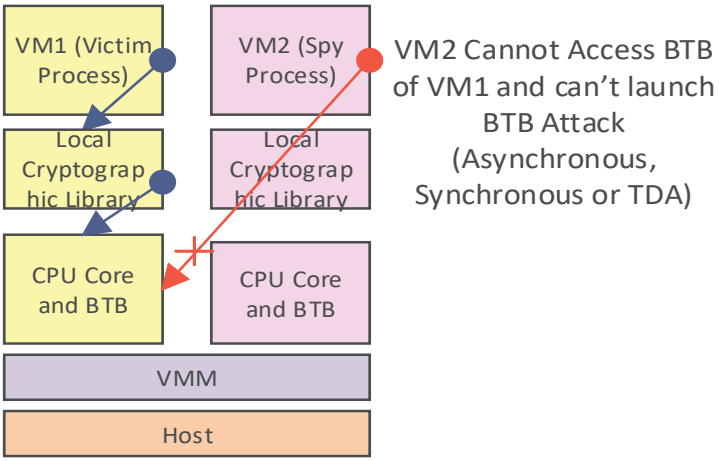


Figure 5.3: VM Configuration with dedicated resources

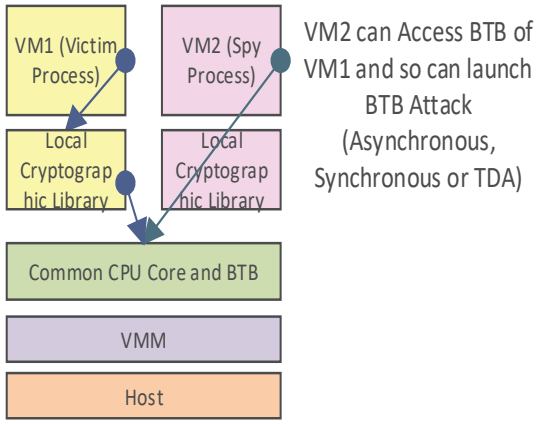


Figure 5.4: VM Configuration with Shared CPU Core (Shared BTB)

The fact inferred from the above discussion suggests that DTA can launch the Cross-VM BPA attack if the VMs share only the cryptographic library. On the other hand, BTB (CPU core) must be shared to launch the BPA attack by the other three methods. Table 5.1 presents a summary of the discussion.

Table 5.1 Applicability of BPA Attack Mechanisms on Cross-VM Platform

BPA Attack Mechanism	**Required Sharing Configuration**	
	Cryptographic Library	**BTB (CPU Core)**
Direct Timing Attack	Shared	Shared / Separate

Asynchronous Attack	Shared	Shared
Synchronous Attack	Shared / Separate	Shared
Trace-Driven Attack	Shared / Separate	Shared

The above discussion reveals that the applicability of a BPA launching method depends on the underlying resource sharing configuration. Correlation between the resource sharing configuration and the possibility of a successful BPA attack has led us to re-analyze the existing approaches handling BPA attack from a different angle to assess their scope in the virtualization environment. The related discussion is carried out in the next section.

5.2 Applicability of Existing Solutions

It is summarized in Chapter 3 that minimal work is carried out in handling BPA attack. When this work was studied, keeping in mind its scope in the virtualization environment, it was found that except Julien et al.[129], the majority of the solutions are suggested for traditional server environments, i.e., non-virtualized environments. Although other solutions can also be applied in virtualization in their direct form, each of them has its own limitations.

The approach suggested by Julien et al. [129] can be directly applied to handle the Cross-VM platform irrespective of the underlying attack mechanism and sharing configuration. The proposed approach allows only white-listed processes to access Hardware Performance Counters (HPCnts) and so the approach may fail if white-listed processes like HTTPS and SFTP get compromised. Suggested algorithmic changes by Agosta et al. [127] can address both common and separate core Cross-VM BPA attack irrespective of the employed attack mechanisms. The preventive steps suggested by this approach are required to be implemented in each of the vulnerable algorithms.

A direct application of the solution given by Ya Tan et al. [128] is possible for a Trace-Driven Attack on a common core Cross-VM platform. The solution has suggested locking some of BTB entries to prevent the spy process from filling the BTB entirely. This kind of solution may lead to locking of BTB entries of a legitimate process having a large number of conditional instructions. Unnecessarily locked entries affect the performance of other

legitimate processes that leads to false positive. Mitigation technique suggested by S. Bhattacharya et al. [130] can work for common as well as separate core Cross-VM BPA attacks launched with the DTA method. The performance of the legitimate processes having a large number of conditional instructions also gets affected as a side effect of this approach. The summary of the behavior of existing approaches is presented in Table 5.2.

Table 5.2: Performance Analysis of Existing BPA Attack Handling Approaches

Approach	Targets Virtualization?	Applicable to handle Cross-VM BPA?	Focused BPA mechanism	Affects Legitimate Process?	Dependent on the Cryptographic Algorithm?
Agosta et al. [127]	No	Yes	Independent of Attack Mechanism	No	Yes. requires modification in each vulnerable algorithm
Tan et al. [128]	No	Only for VMs with common core	Trace-Driven Attack	Yes	No
Julien et al. [129]	Yes	Yes	Independent of Attack Mechanism	Yes	No
S. Bhattacharya [130]	No	Yes	Direct Timing Attack	Yes	No

5.3 Summary

One VM hosting only one service is a typical setup of virtualization. When a service running on a VM gets compromised [131], it can launch a BPA attack [120] on a cryptographic process running on a separate VM. BPA attack in such a case takes the form of a Cross-VM attack. The possibility of such a Cross-VM BPA attack depends on the resource sharing configuration of the VMs. It was realized that the type of resource that must be shared

between the victim and the attacker VM depends on the employed methodology. Where the cryptographic library must be shared for the DTA method, shared BTB is the mandate for the other three methods (i.e. Asynchronous, Synchronous, and Trace-Driven Attack).

Discussion regarding the Cross-VM BPA attack adds one more angle in assessing the performance of the existing solution. We need to reanalyze them to assess their scope in virtualization. Scope assessment of the existing four solutions presented in Table 5.2 reveals that the solution proposed by only Julien et al. [129] considers the virtualization environment. Although other solutions can be applied to handle the Cross-VM platform in their direct form, all of those solutions, including the virtualization-based solution, have their own limitations. Solutions proposed by Ya Tan et al. [128], Julien et al. [129], and S. Bhattacharya et al. [130] manipulate architectural components like the BTB, Hardware Performance Counters and Branch Predictor respectively. Manipulation of the architectural component(s) may affect the performance of the legitimate processes. The approach suggested by Agosta et al. [127] needs manipulation in each vulnerable cryptographic algorithm.

Analysis of existing solutions has shown a need for a BPA handling approach, which is independent of the target cryptographic algorithm. Moreover, the BPA attack must be handled without affecting the performance of legitimate activities. Proposal for such a solution in-line with the identification of a need for a new BPA handling approach is presented in the next chapter.

CHAPTER

6 Research Contribution

We summarized in section 5.3 that there is a need to propose a solution to handle the Cross-VM BPA attack considering the limitations of existing solutions. Proposal and implementation of the new solution require simulation of the attack also to check its effectiveness. The attack simulation also contributes to giving directions while designing the solution. Considering the significance of attack simulation, we have opted for the BPA attack simulation before proceeding for attack detection. We found that there are some cases where the BPA attack is successfully simulated [120][125][128] where a non-virtualization environment is considered. We have also simulated them on a Cross-VM platform in accordance with the environment considered in our work. Results of attack simulation are discussed in Section 6.1.

As per the objectives defined in Chapter 4, the new solution should not affect the normal system performance. Designing such a solution requires careful observation of the behavior analysis of attack procedures. An explanation of the observed behavior of all the four methods is discussed in Section 6.2.1. The attack simulation, as well as the behavior analysis, have provided many important inputs for designing the attack detection solution. A detailed explanation of the proposed solution is provided in Section 6.2.2.

6.1 Attack Simulation

We have considered a virtualization environment for the attack simulation, as discussed in the previous chapter. There were two separate VMs, where one was the victim VM, and the other was the attacker VM that would launch a BPA attack. Resource sharing between the two VMS can be carried out in a total of three ways: Fully Isolated, Fully Shared, and Hybrid. In this simulation, we have opted for the Hybrid type of resource sharing

configuration as it is the defacto standard and the most efficient one among the three types, as discussed in Section 5.1.

The victim and attacker VMs share some of the software and /or hardware resources as per the Hybrid type configuration. The type of resource shared between the two VMs depends on the attack method under consideration. Figure 6.1 represents the basic simulation environment for the attack simulation. Simulation setup specific for each of the attack methods is shown in the respective diagrams, Figure 6.2 and Figure 6.6.

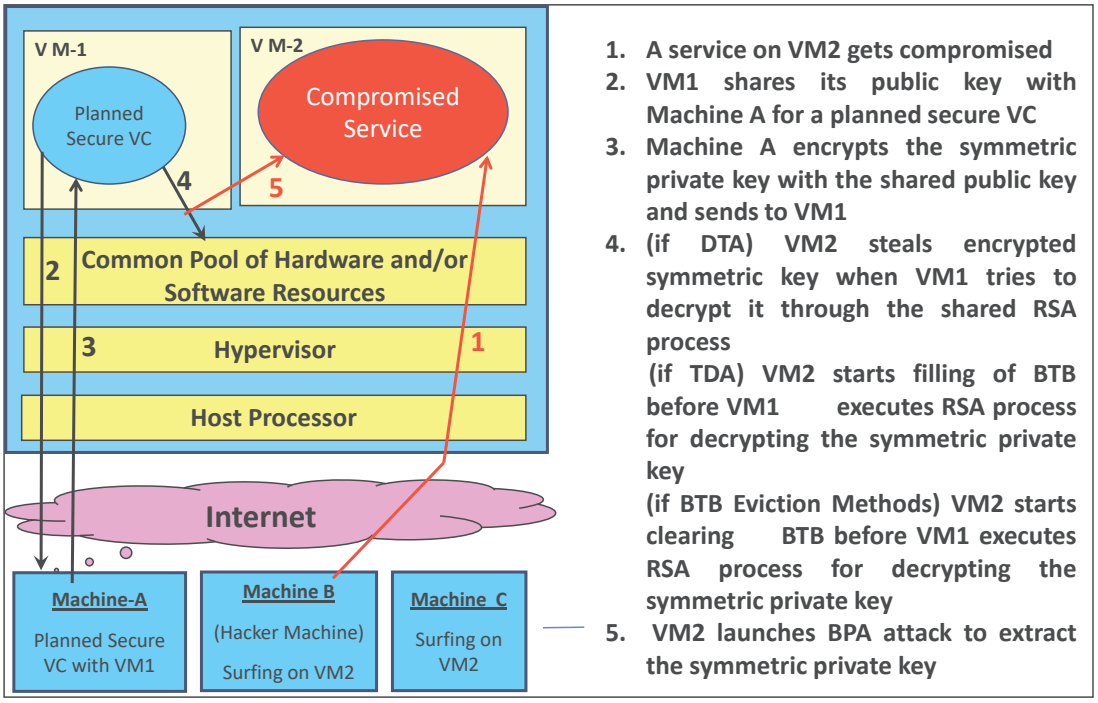


Figure 6.1: Simulation Environment

We have created VMs with KVM Hypervisor and Ubuntu 15.10 Operating System to simulate the operating environment. As shown in the diagram, VM1 has planned a secure Video Conferencing (VC) with a Machine-A. It is assumed that VM2 is hosting HTTPS, and another attacker on Machine-B has compromised that service and, in turn, compromises VM2.

VM1 shares its public key with Machine-A and gets encrypted Symmetric Private Key (SPK) in response. VM1 needs to decrypt the received key by passing it to the RSA process. The compromised VM2 becomes active at this point to launch the BPA attack. The action

employed to extract the key depends on the method under consideration, as shown in Step-no. 4 in Figure 6.1.

A detailed discussion on the simulation setup for each method and obtained results are discussed in the respective subsections of this section. In the simulation of both DTA and TDA, VMs are created with KVM Hypervisor where the host Operating System (OS) is Ubuntu 15.10.

6.1.1 Direct Timing Attack

The simulation environment of Cross-VM DTA is shown in Figure 6.2. As shown in the diagram, VM1 and VM2 are the victim and the attacker (compromised) VMs, respectively. The cryptographic library (especially RSA) is kept shared between them through a shared VM, VM3. VM1 is initiating a secure video conferencing with another machine A for which a Symmetric Private Key (SPK) is exchanged by generating a public-private RSA key pair. With the assumption that the VM2 can access encrypted Asymmetric Private Key (APK) of RSA, VM2 launches DTA on VM1.

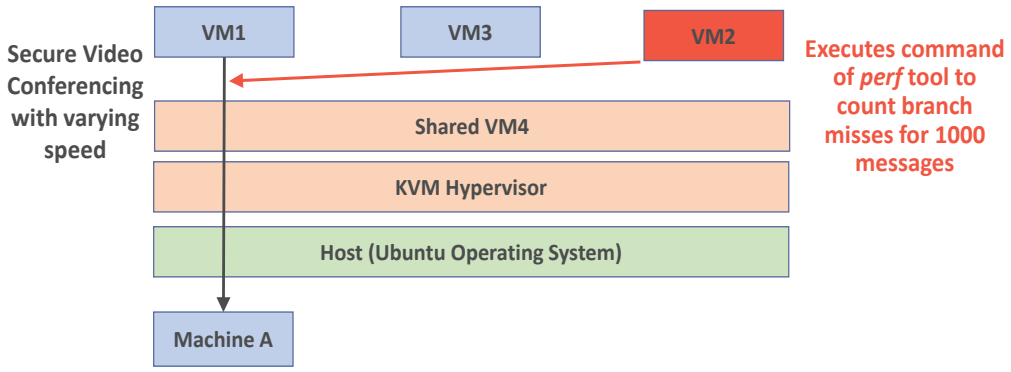


Figure 6.2: Attack Environment: Direct Timing Attack

As per the procedure of Direct Timing Attack, we have taken a data set of 1000 different messages for decrypting RSA private key. As discussed in section 3.3.1 (Eq. 3-1), messages are divided into four sets, namely, M1, M2, M3, and M4, during the Offline phase. During the Online phase, a number of branch misses instructions were counted for each message of each set by executing the following command with the *perf* [132] tool.

$perf stat -e branch-misses java RSA_File_to_Monitor msg_to_decrypt

Here, avg(BM(M_i)) represents the average number of branch misses of all the messages belonging to set M_i (i=1 to 4). Further, the average of BM(M_1) is compared with BM(M_2), and that of BM(M_3) is compared with BM(M_4) as per Eq. 3-2. As a sample case, the results obtained during the prediction of the 50th bit of the key were plotted. Results represented in Figure 6.3 corresponds to the comparison between avg(BM(M1)) and avg(BM(M2)). In a similar way, a comparison between avg(BM(M3)) and avg(BM(M4)) is shown in Figure 6.4. Obtained results reveal that avg(BM(M1)) and avg(BM(M4)) are more than avg(BM(M_2)) and avg(BM(M_3)) respectively. Accordingly, it can be said that the result correctly predicts the 50th bit as 1. Every bit was predicted recursively using the same method.

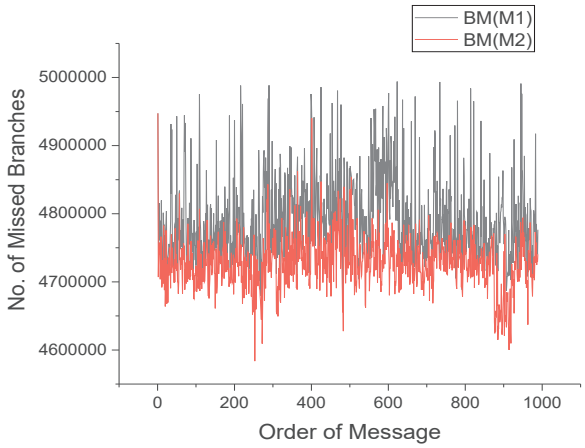


Figure 6.3: Distribution of BM (M1) and BM (M2)

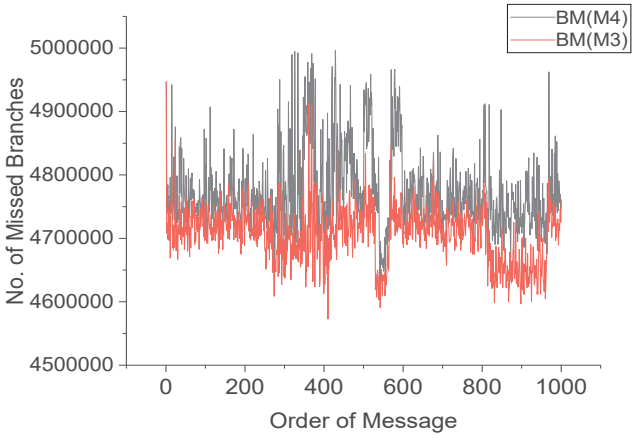


Figure 6.4: Distribution of BM (M3) and BM (M4)

The result can be plotted for each bit as in Figure 6.3 and Figure 6.4. Experimental results for two 1024 bits secret keys used for simulation are shown in Table 6.1. It represents the actual and predicted key values where the mispredicted bits are highlighted in bold and underline.

Table 6.1: Actual and Predicted Private Decryption Key

Sr. No.	Actual Key	Predicted Key
1	179801dc316e4689401958cdcd28e3e5d3b7c5af0088bd0c29f0c410edaeeb55861b86370cc0d4b266ca0fc39dda32bd610a9439a7a3fcf8945228e8c0fff1d69bca53d8752efc37bbc3d886730e8dec37de42974f1f089b421f58f4c7eabb650580ba8cb008b7f16e0bb29805df8d169a05c697408fbf09ecd8800514f18521	179801dc316e4689401959cdcd28e3e5d3b7c5af0088bd0c29ffe410edaeeb55861b86370cc0d4b266ca0fc39dda32bd610a9439a7a3fcf8945228e8c0fff1d69bca53d88A2efc37bbc3d886730e8dec37de42974f1f089b421f58f4c7eabb650580ba8cb008b7f16e0bb29805df8d169a05c697408fbf09ecd8800514f18521
2	3199ca0695d34a1f357b7df8069ff01cbec878298843254f699cfe2984acc3c89a0a49279289d187a107b962de3289bac43fd5009ce9c4365059fafa2ebd26fb7095523a64f5ee6f30c43af64830139d78b0972e0fecb954b44c3b171a7250e622c30bb2907067 97a8be360e0ac84fb5da7bc6c15361fc240971d74500bea1a1	3199ca0695d34a1f357b7df8069ffB1cbec878298843254f699cfe2987fcc3c89a0a49279289d187a107b962de3289bac43fd5009ce9c4365059fafa2ebd26fb7095523a64f5ee6f30c43af64830139d78b0972e0fecb954b44c3b171a7250e622c30bb2907098 97a8be360e0ac84fb5da7bc6c15361fc240971d74500bea1a1

We have simulated different 30 keys. Simulation for each key was also carried out for around 50 times. The maximum, minimum, and average percentage of correctly predicted bits were calculated from the results obtained for each key. The obtained maximum, minimum, and average values for randomly selected 25 keys are shown in Figure 6.5.

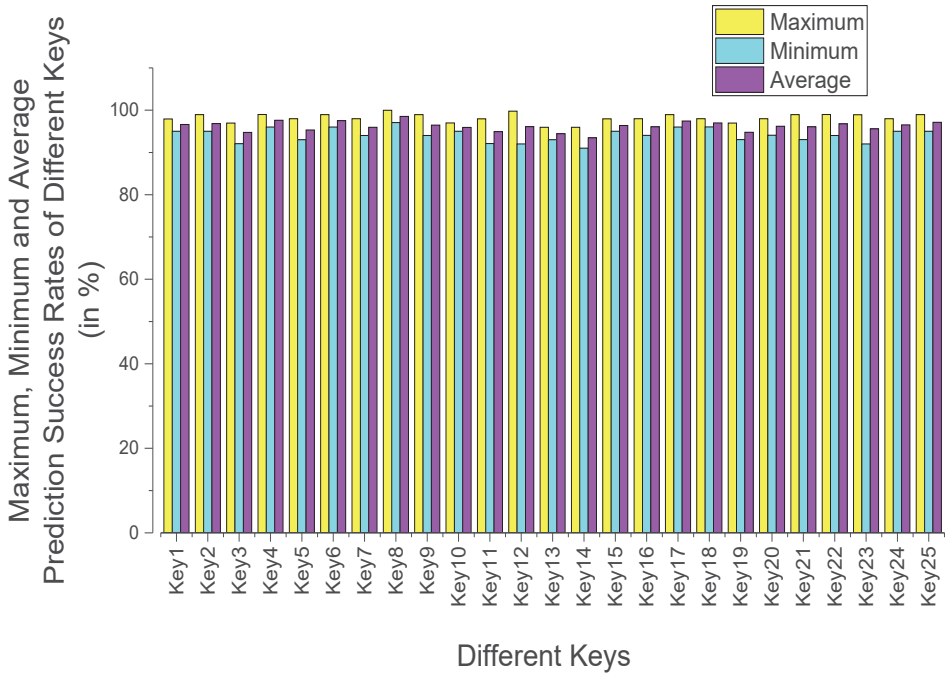


Figure 6.5: Key Prediction Rate for different keys

Results shown in Figure 6.5 reveal that our prediction accuracy is a maximum of 97-98%. The success rate of predicting the bit values is claimed to be 100% in [125]. Although 100% success could not be achieved in this experiment, that much accuracy was not needed as our objective is to simulate the attack rather than stealing the data. Most importantly, our proposed solution predicts the attack by the time it predicts a few bits. Hence, simulation with 97-98% accuracy is sufficient to consider it as an attack in progress.

6.1.2 Trace-Driven Attack

The simulation of TDA requires to fill the BTB in continuous form. The spy process for launching TDA needs to be written carefully so that all the entries of the BTB gets filled. A simulator that can efficiently simulate architectural components is required to be chosen for simulating TDA. The Gem5 [133] simulator is one such open-source multithreaded simulation platform.

We created two VMs, VM1, and VM2, on the Gem5 simulator and configured to have common CPU core and so common BTB as per the requirement of TDA. The attack environment is shown in Figure 6.6.

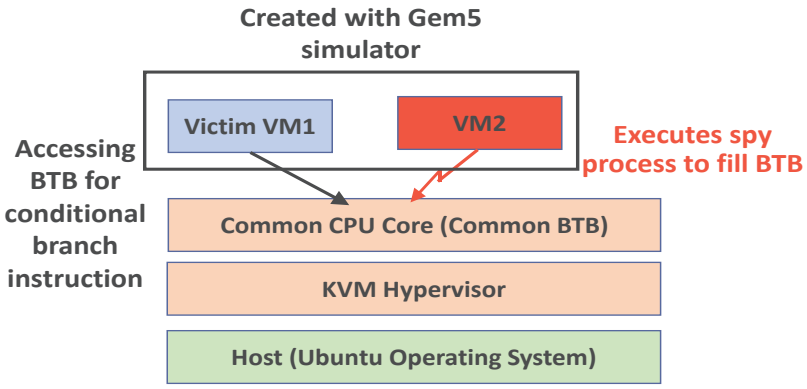


Figure 6.6: Attack Environment of TDA

Parameters used for simulation are mentioned in Table 6.2. The spy process and RSA cryptographic process were initiated in VM1 and VM2, respectively. The BTB Entries parameter of the Gem5 simulator was set as per the total number of branch instructions in the spy process. Spy process was started before the RSA process starts, and it was set to run continuously to observe the execution time as per the attack methodology.

Table 6.2: Parameters used for Gem5 Simulator

Gem5 Simulation Parameter with Values		
BTBEntries	=	4096
BTBTagSize	=	16
globalCtrBits	=	2
globalPredictorSize	=	8192
Max Threads of SMT	=	4
localHisoryTableSize	=	2048
localPredictorSize	=	2048
localCtrBits	=	2

The plot in represents how bits were predicted with the TDA procedure. No. of clock cycles (Execution Time) of the spy process is the decisive parameter to predict the logic of the private key bit of the RSA decryption algorithm. The x-axis values represent bit positions of the private key, while the y-axis values represent the observed number of CPU clock cycles. The plot in Figure 6.7 represents that the observed number of clock cycles were more at those places where the corresponding key bit is 1 in the RSA key. On the opposite side, the number of clock cycles observed by the spy process was comparatively low during the time when the concurrently running RSA process encounters 0 in the key bit. Extracted bit for the range 30-39 is explicitly highlighted for showing the result of the attack launching procedure. As shown in the diagram, bits were predicted from the observed difference in the execution time of the spy process.

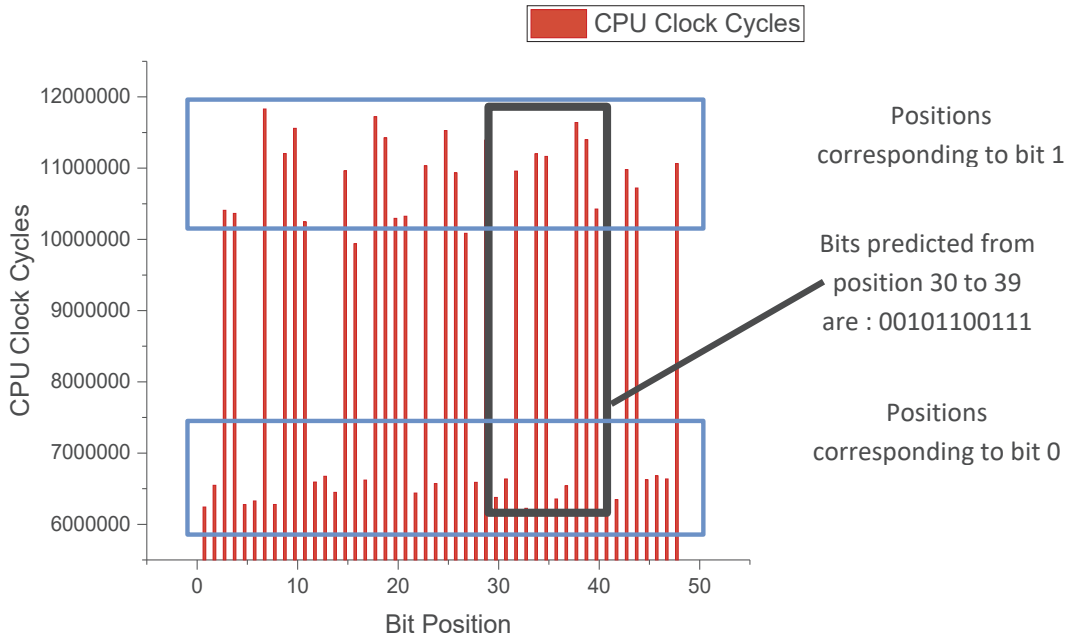


Figure 6.7: Results of Simulation for Trace-Driven Attack

The simulation of TDA was carried out using around 20 key values. The simulation was done 30-35 times for each key. Although it is difficult to represent results obtained over the entire key for all the values, some samples are represented in Figure 6.8. The plot represents observed execution time for four different keys. As the key length is more, only part of the key is shown to give the visualization of the predicted bits. Plotted results reveal that the observed execution time for each key is clearly falling in either of the two

parts: one representing a high value of execution time and the other representing a low value of time. It means that the observed time never gives an ambiguous result, so that bit prediction becomes difficult. As shown in the diagram, a clear separation between the two ranges of CPU clock cycles are observed for all the four keys. The bits position corresponding to the lower clock cycles represent bit 0, where higher values of clock cycles represent bit 1.

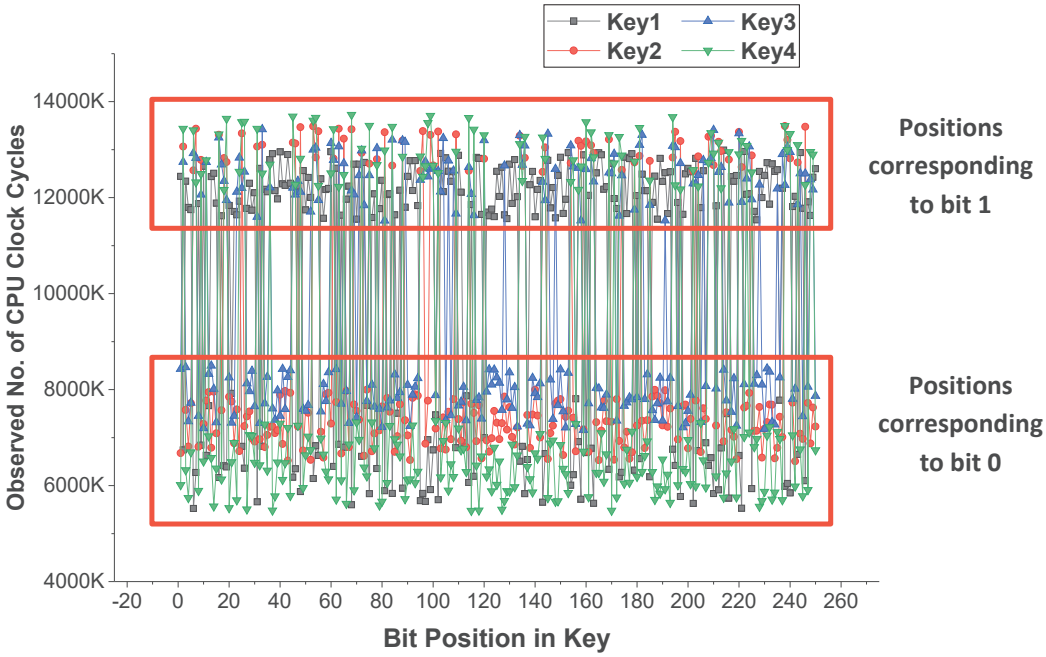


Figure 6.8: Observed CPU CLock Cycles over a part of Key for different Key Values

Simulation results discussed in this subsection gives support to the fact [120] that TDA can be simulated in a virtualization environment.

6.1.3 Asynchronous BTB Eviction Attack

Eviction of BTB entries by executing a dummy process with a large number of conditional instruction is one of the core actions of the Asynchronous attack. The target branch of the MM algorithm needs to be brought to BTB every time it is executed because of the continuous eviction of BTB entries. The bits are predicted in the Asynchoinous method by performing simulation with the DTA-like method. As discussed in Subsection 3.3.1, the taken sample of a larger message set is distributed among four sets, M_1 to M_4 (as per Eq. 3-1), based on the predicate defined over the total number of missed branches

in case of DTA. The definition of a predicate is changed to consider the actual execution status of the target branch while launching an attack with the Asynchronous BTB Eviction method (ABEA), as discussed in Subsection 3.3.2.

The above-discussed factor reveals that the simulation setup required for the Asynchronous BTB Eviction method is similar to that of DTA (Figure 6.2) because of the similar attack procedure. As an additional requirement, BTB is also needed to be shared between the compromised and the victim VM for simulating this Asynchronous method.

A dummy process was executed to evict the BTB entries, and a large set of sample messages were distributed among the four sets as per Eq. 3-1, where the predicate was replaced as per the method of Asynchronous BTB Eviction. CPU clock cycles of each of the message in the four sets were measured using the following command of *perf* [132] tool :

$perf stat -e cpu-clock java RSA_File_to_Monitor msg_to_decrypt

If the average execution time (observed CPU clock cycle) of the M_1 and M_2 sets is more significant than that of M_3 and M_4, then the bit is predicted as 1 else it is predicted as 0. The simulation was carried out with 1000 messages, where the obtained results are plotted in Figure 6.9. The x-axis represents the bit positions, and the corresponding value on the y-axis shows the difference between the average execution time mentioned above. The positive difference reveals the presence of bit 1, while 0 is revealed by the negative difference.

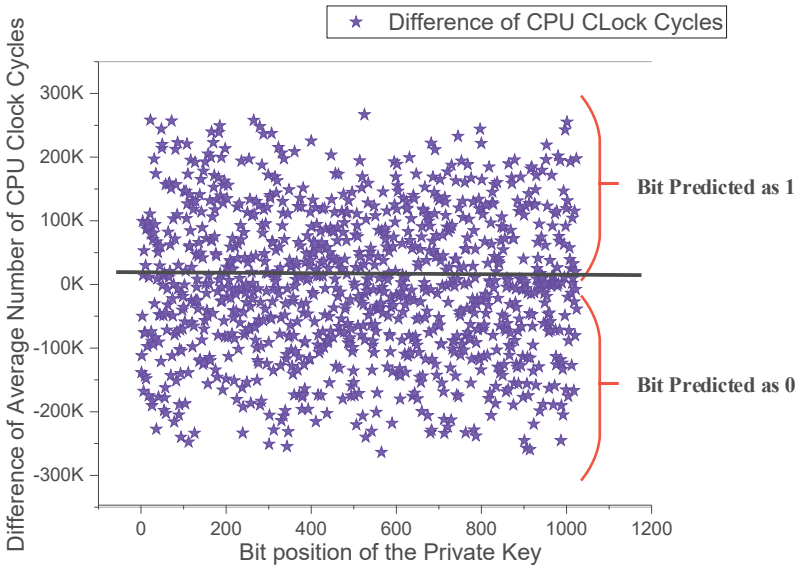


Figure 6.9: Bits Prediction from the Observed CPU Clock Cycles

We carried out a partial simulation of this attack, where the success ratio varied from 30%-60%. Although an overview of the attack method is provided in [120], we did not find a detailed attack procedure in existing research work. Additionally, there is one more point to be considered regarding the simulation of this Asynchronous attack. The attack considers the S&M algorithm. If the Mongomery Ladder algorithm replaces the S&M algorithm, the possibility of the attack may get eliminated because of the balanced branch instructions. A similar issue arises in the case of DTA also, but the replacement of the reading action of CPU clock cycles by the number of branch misses can still make the attack possible [125]. An analysis on a similar track is required for the Asynchronous BTB Eviction attack also. Moreover, the primary aim of the simulation is not to extract the correct bits. The fundamental objective of the simulation is to observe the underlying actions. In this regard, with partial simulation, the Asynchronous BTB Eviction method is considered for designing a solution approach. Scope analysis of ABEA in the presence of the Ladder algorithm is taken as future work.

6.1.4 Synchronous BTB Eviction Attack

The attack detection method of Synchronous BTB Eviction Attack (SBEA) discussed in Subsection 3.3.3 highlights the need for a synchronous dummy process. Establishing synchronism between the dummy and the RSA processes is extremely difficult especially when both the processes are located on different VM. No research work has been found that shows how practically it can be launched even on a non-virtualization environment. We have taken the simulation of SBEA as our future work, looking at its implementation complexity. However, as discussed further in the next chapter, the proposed approach is efficient in detecting the presence of SBEA even if it is launched successfully.

Obtained results have revealed the possibility of a BPA attack on the Cross-VM platform. The observation drawn from the simulation has provided a direction in proposing a solution for handling these attacks. Behavior analysis of all the four attack launching methods and proposal of a solution are the major contributory factors of our work, which are discussed in Section 6.2.

6.2 The Proposed Solution

Analysis of existing solutions discussed in Section 5.2 has highlighted two essential points to be considered while designing a solution: the solution should not manipulate any architectural component, which ultimately affects the normal system performance. It should be independent of the type of victim cryptographic algorithm, else suggested modification for attack prevention needs to be applied in each vulnerable algorithm in each library.

The study of different attack launching methodologies reveals that the procedure followed to extract the private key bits looks like a normal process. The absence of the abnormal action during the process of launching the attack makes it very difficult to be detected. In such a case, not any IDS/IPS or firewall, but only a behavioral analysis can direct in identifying the abnormal processes. We have carried out a detailed study on the behavioral patterns of BPA attack launching methodologies, which is discussed in Section 6.2.1. A solution named *Chaturdrashta* is presented based on the behavior analysis of the attack methods, which is described in Section 6.2.2. *Chaturdrashta* explicitly focuses on the

detection of DTA and TDA only, but it can be applied to handle Asynchronous and Synchronous BTB Eviction methods also in its direct form.

6.2.1 Behavior-based Approach

Branch Prediction Analysis attack can successfully extract the private key bits within a very short span. The most crucial but interesting factor of this attack is its attack launching method itself. Analysis of the primary actions of various attack launching methods was carried out with consideration that it may contribute to building the base of the new solution. Behavioral analysis of each of the attack launching methods is separately discussed in this subsection.

6.2.1.1 Direct Timing Attack

The process of DTA is divided into Offline and Online phases, as discussed in Subsection 3.3.1. During the Online period, decryption of messages is performed that calls the decryption process (here, RSA). The simulation was done over a large message set (size of at least around 1000). For each of the messages, the decryption was performed twice of the private key length. A large number of messages with a lengthy key led to a very high number of calls to the RSA process. In the Cross-VM environment, the spy process executing the Online phase of DTA and the RSA process are located on separate VM. Frequent calling of the RSA process resulted in a large number of packet traversal between the compromised and the shared VM holding cryptographic library. This action of packet traversal with very high frequency is one of the prominent actions performed during the DTA process.

The study of the Online phase highlighted one more aspect. Every time the decryption algorithm is called, the total number of branch misses is read. As the total count of calling the decryption process is high (as discussed previously), the count of reading the total number of branch misses also becomes high. Consideration of the actual implementation detail represents that to get the total number of branch misses, an appropriate type of Hardware Performance Counter (HPCnt) are needed to be read. An event called *branch-misses* is called to obtain this HPCnt value. Observing the system performance by reading an appropriate Hardware Performance Counter is known as profiling. As profiling for reading the total number of branch misses is

performed with high frequency during the Online phase, it can be taken as the second prominent action of the DTA process.

The above discussion reveals that a high-frequency packet traversal with the RSA process and a high frequency of reading an HPCnt are the two primary actions performed during the execution of the DTA process. An observation of the frequency with which the above two actions are performed can become decisive parameters in identifying the BPA attack.

The study of other methods launching the BPA attack is also carried out in the same manner.

6.2.1.2 Trace-Driven Attack

The attack methodology of TDA explained in Subsection 6.1.2 exhibits that the spy process continuously fills the BTB of BPU with a large number of conditional branch instructions. This step of the spy process keeps all the entries of the BTB occupied by the instructions of the spy process. In other words, the BTB occupancy of the spy process becomes much higher than any other running processes. Accordingly, high BTB occupancy is one very noteworthy feature of the spy process.

In addition, to continue filling the BTB, the spy process measures its execution time after each iteration to predict the bit value as per observed time difference. Measurement of the execution time is also a profiling action that leads to a reading of its respective HPCnt by executing the *cpu-clock* event. As the execution time is measured frequently, profiling is also performed with a high frequency, as in the case of DTA. Summarizing the TDA actions, it is revealed that the TDA spy process performs two primary actions: Occupying BTB with high ratio and profiling with high frequency for measuring the execution time.

6.2.1.3 Asynchronous BTB Eviction Method

The study of the attack procedure as well as the observed behavior while simulating the Asynchronous BTB eviction method revealed two essential characteristics:

(1) Asynchronous BTB Eviction method employs actions similar to that of DTA with only one difference in the defined predicate to divide a sample of the large message

set among four sets. The predicate is defined on the total number of missed branches in DTA, where the actual outcome of execution status on conditional branch instruction is considered in defined predicate for the BTB Eviction methods. (2) The DTA-like procedure is employed in concurrent to a dummy process that executes a large number of conditional instructions to evict BTB entries.

The first characteristic represents that the primary action of the BTB Eviction methods is exactly the same as that of DTA. This is with reference to the fact that apart from the predicate definition, all the other details of the simulation are the same as DTA. The simulation details discussed in Subsection 6.1.1 and 6.1.3 reveal that a large amount of packet traversal has resulted during the BTB Eviction methods also as in the case of DTA. Further, where the total number of branch misses are observed during the simulation of DTA, simulation of total CPU clock cycles are monitored while simulating this Asynchronous method. An in-depth study of the core action behind observing the total branch misses as well as clock cycles, reveal that respective HPCnts are accessed in both of the cases. Accessing HPCnt with high frequency becomes a primary action of BTB Eviction methods, also like DTA.

Although we found a very high resemblance between the DTA and the Asynchronous BTB Eviction method, a correlation between the Asynchronous method and the TDA was also highlighted by the second characteristic defined above. The DTA-like procedure was executed to predict the key bit in the BTB Eviction methods, where a dummy process is continuously evicting the BTB entries. The frequent eviction from BTB is implemented by executing a large number of conditional instructions, which also results in a high BTB occupancy ratio. Additionally, CPU clock cycles are also measured continuously as part of the simulation. As per the above discussion, the primary actions of BTB Eviction methods resemble those of DTA as well as with TDA.

6.2.1.4 Synchronous BTB Eviction Method

The dummy process of the Synchronous method continuously evicts BTB entries by executing a large number of conditional instructions. The functionality is similar to the dummy process of the Asynchronous method, with the only difference of synchronism between the dummy and the RSA process. The above action results in a

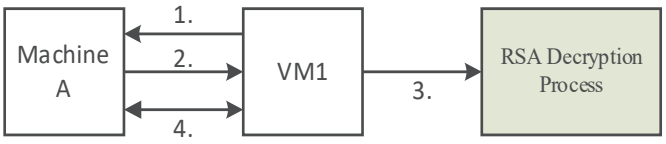
high BTB occupancy of the dummy process, like in the case of an Asynchronous attack. As the primary actions of the dummy process of the Asynchronous method resemble those of TDA, as discussed in Section 6.2.1, the primary actions of the Synchronous method also resemble the actions taken during the TDA.

Behavioral study of all the methods reveals that the packet traversal with the shared RSA process, frequent filling of BTB, and reading of HPCnts are the combined primary actions of DTA, TDA, and BTB Eviction methods. One interesting factor regarding each of the identified primary actions is the frequency with which the actions are performed. All the actions are basically normal-looking and seem harmless at first sight. They can be performed by some legitimate processes also. Applications like System Profiler, Update-Manager, and Quick EMUlator (QEMU) also perform profiling. In such circumstances, the spy process performing profiling is differentiated from the legitimate process on the basis of the frequency of profiling, which is quite high in the case of the spy process. The fact is equally applicable to each of the primary actions discussed above, which are also performed with high frequency.

The difference in action frequency is such that there would be almost no scope of false positive if the spy process is caught based on its action frequency only. This fact is revealed through experimental analysis in Chapter 7. However, deep dive into the implications of the BPA attack, i.e., post-attack observation, becomes essential for a confirm detection of the attack. Usage analysis of the extracted key drive us to consider one more essential point related to the BPA attack.

6.2.1.5 Post-Attack Methods

The extraction of the key by the BPA attack is employed where the Symmetric Key Cryptography (SKC) is implemented for secure communication. A case of SKC is presented in Figure 6.10, where secure communication is planned between Machine A and VM1. VM1 generates a Public-Private key pair of RSA where Machine A uses the Public key of that pair to encrypt the Symmetric Private Key (SPK) and sends it to VM1. VM1 calls the RSA decryption process to decrypt the SPK using Asymmetric Private Key (APK) of the generated key pair.



1. VM1 generates pair of public-private RSA key and publishes the public key
2. Machine A encrypts the Symmetric Private Key (SPK) with the Public Asymmetric Key of VM1 and sends it to VM1
3. VM1 decrypts SPK with Asymmetric Private Key from the pair
4. Secure message exchange starts with decrypted PSK

Figure 6.10: Application of RSA

As a result of the attack, the APK of the target crypto algorithm, i.e., RSA, becomes available to the attacker. Once APK is available, the attacker can get SPK. Further, the attacker would be able to steal all the confidential messages traversed between Machine A and VM1 as all of the packets use the SPK for encryption.

An attacker can successfully extract the SPK by launching a BPA attack, where the extracted key can be used to decrypt the encrypted data packets provided the packets are available to it. Availability of key without having access to encrypted data packets results in the fruitless launching of the BPA attack. For a fruitful attack, an obvious step that an attacker takes after stealing of SPK is to trap the data packets by launching a MITM attack [134].

From the above discussion, it can be said that trapping of packets by launching attacks like MITM, is an obvious step that the adversary takes. We can summarize the primary actions launched during all the three types of attacks under consideration as follows:

- Direct Timing Attack:
  - ✓ Frequent access to RSA
  - ✓ Frequent reading of performance counters to measure the total number of branch misses
  - ✓ Packet trapping activity (post-attack) for fruitful DTA

- Trace-Driven Attack
  - ✓ Frequent filling BTB during TDA

- ✓ Frequent reading of performance counters to measure the execution time of the spy process.
- ✓ Packet trapping activity (post-attack) for fruitful TDA

- Asynchronous BTB Eviction Attack
  - ✓ Frequent access to RSA
  - ✓ Frequent Eviction of BTB Entries
  - ✓ Frequent reading of performance counters to measure the execution time of the dummy process
  - ✓ Packet trapping activity (post-attack) for fruitful attack

- Synchronous BTB Eviction Attack
  - ✓ Frequent Eviction of BTB Entries
  - ✓ Frequent reading of performance counters to measure the execution time of the dummy process.
  - ✓ Packet trapping activity (post-attack) for fruitful attack

By keeping the above mentioned primary actions into consideration, a four-eyed solution *Chaturdrashta* is presented to handle the BPA attack. Although we explicitly focus on DTA and TDA for proposing the detecting solution, *Chaturdrashta* can detect the presence of the BPA attack even if it is launched with BTB Eviction methods. A detailed discussion of the proposed approach is presented in the next subsection.

6.2.2 *Chaturdrashta* : Model of the Proposed Solution

The proposed approach focuses on the primary actions employed during the Direct Timing Attack and Trace-Driven Attack. The discussion revealed that each of the attacks is comprised of three primary actions where two of the actions (access to HPCnts and initiation of packet trapping activity) are common between DTA and TDA, and one action is different for both the methods. Accordingly, we need to observe the total four actions for detecting the presence of both DTA and TDA. We have proposed a four-eyed approach by taking the above discussion as the base. We have named our algorithm *Chaturdrashta,* which is a Sanskrit word, and its meaning is one with four eyes.

This four-eyed solution is comprised of four monitors to observe previously discussed primary actions. CryptoLibrary Access Monitor (CAM), BTBAccess Monitor (BM), Interrupt Monitor (IM), and Network Monitor (NM), and the corresponding action they observe, are shown in Figure 6.11. The model for *Chaturdrashta* is shown in Figure 6.12.

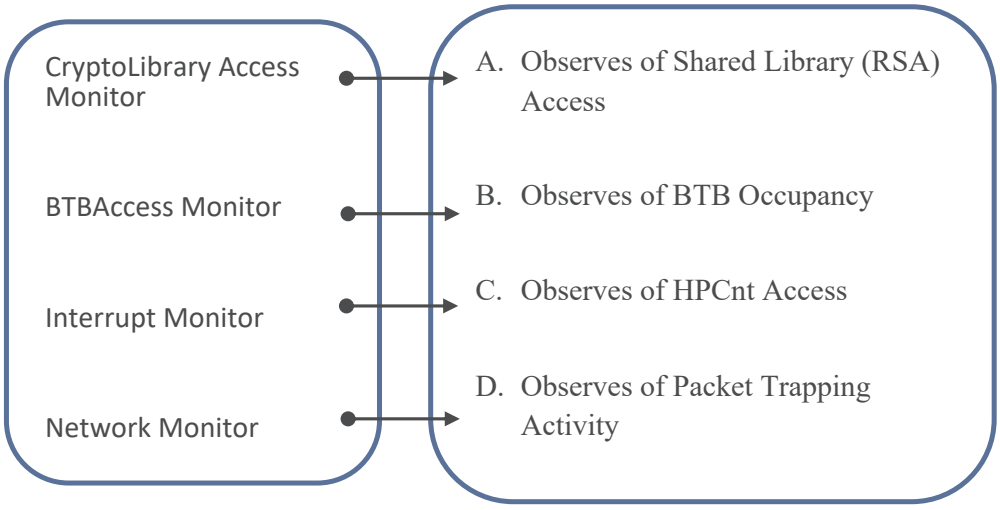


Figure 6.11: Functional Mapping of *Chaturdrashta* Components

All four monitors are implemented on the host level. CAM, BM, and IM are initiated before any of the VM is started. The monitors are set to generate alert as per the action observed of all the running VMs. All the running VMs are assumed to be legitimate in the initial stage, where the status of each VM is initialized as *trustworthy*. The status of the VM(s) may get gradually changed to *suspicious*, *spy*, and *malicious* when the monitors detect it. The functional explanation of each of monitors is given below:

- CryptoLibrary Access Monitor (CAM) :

  It monitors the frequency of cryptographic library access, i.e., packet traversal intensity between the shared VM holding cryptographic library (specifically RSA) and all the other VMs. Our CryptoLibrary Access Monitor module runs in the background on the host system. An alert would be generated for the VM generating high traffic, and it is recommended to change the VM status from *trustworthy* to *suspicious*.

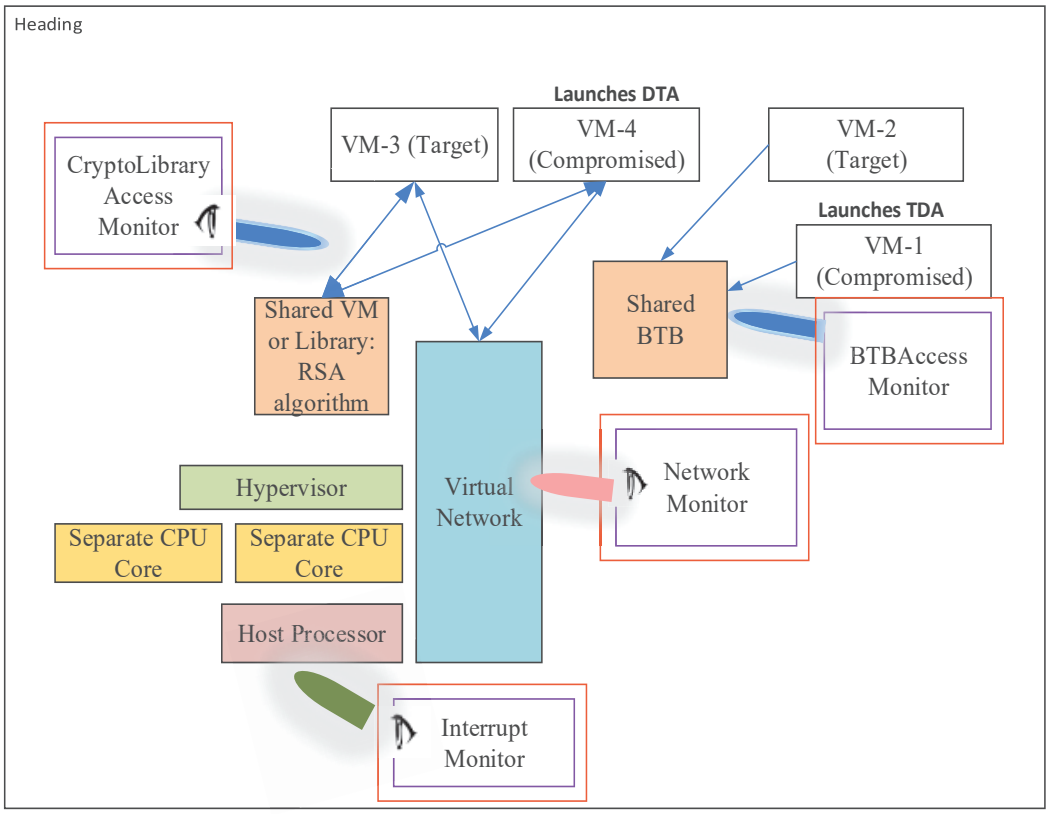


Figure 6.12: Model of *Chaturdrashta*

The graphical presentation of functions performed by the CAM is given in Figure 6.13.

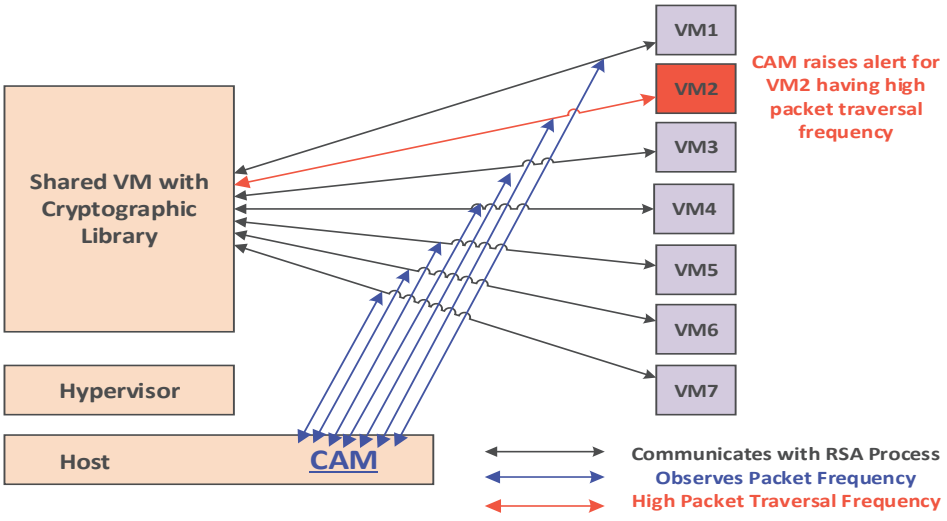


Figure 6.13: CryptoLibrary Access Monitor : Functional Representation

We have implemented CAM with *tcpdump* tool to monitor packet traversal frequency at a specific interval. However, a more sophisticated module could be developed in the future to replace *tcpdump*.

- BTBAccess Monitor (BM)

Identification of the process / VM that frequently fills the BTB is the primary function of BM. A process that frequently fills BTB holds high BTB occupancy. As a process running in the background on the host system, BM calculates the BTB occupancy ratio of all the other running processes to identify the process with high BTB occupancy.

BTB occupancy of different processes is calculated by observing their BTB hits. The values of this event can be obtained by reading the appropriate Model Specific Register (MSR) by executing *rdmsr* assembly language instruction of x86 architecture. We need to initialize ECX, hardware statistics counter, and index registers with appropriate values for retrieving the details regarding the branch instructions. Retrieved number of BTB hits and known size of BTB reveals the BTB occupancy of a process.

Experimental analysis carried out to observe the behavior of BM revealed that the BTB occupancy of the spy process is much more than any other process. BTB occupancy of the spy process averaged out from the observed large number of results was set as the threshold. For this specific threshold value of the BTB occupancy ratio, BTBAccess Monitor generates alert for all the processes resulting in BTB occupancy higher than the threshold value. BM recommends status change from trustworthy to *suspicious* for the process/VM that runs a process with a high BTB occupancy ratio. Figure 6.14 represents the function carried out by the BM.

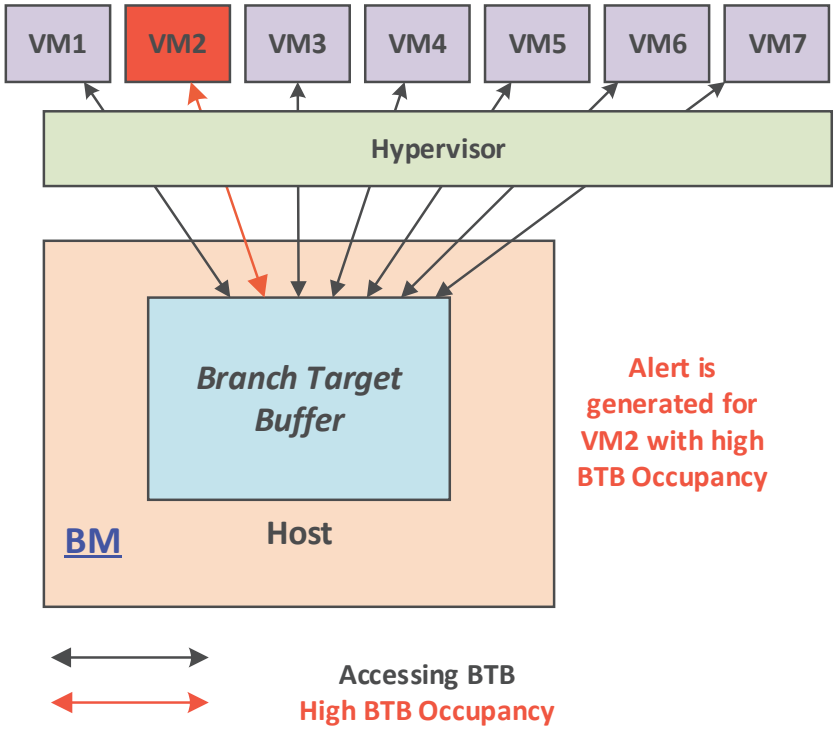


Figure 6.14: BTBAccess Monitor : Functional Representation

- Interrupt Monitor (IM)

IM is designed to monitor the access frequency of the Hardware Performance Counters for all the VMs. HPCnts can be accessed from any privilege level provided PCE (Performance Monitoring Counter Enable) bit of control register CR4 is set. The bit is 0 in default case that requires a special privilege to read HPCnts.

When the access to HPCnts is requested, it results in a system call, namely, SYS_PERF_EVENT_OPEN. System calls generated while trying to access Performance Monitoring Counters (PMCs), Time-Stamp Counters (TSCs), and Machine Status Registers (MSRs) are trapped into the respective process defined in *msr.h* of the kernel. System call generated during HPCnt reading, i.e., SYS_PERF_EVENT_OPEN, is trapped in a kernel function named *native_read_pmc()* located in the *msr.h* file.

Implementation of IM is done by modifying this Linux kernel function *native_read_pmc()* of the Ubuntu Operating System. Code is added in this function to count the total number of times each of the running processes accesses the HPCnts. The

functionality of IM is represented in Figure 6.15. Like BM, IM sets a threshold for the counter keeping track of HPCnt access frequency. An alert is generated for the VM for which HPCnt access frequency crosses the threshold value. The threshold value is decided after observing the results by setting it to different numbers. Details discussion on how the threshold value is decided, is given in Section 7.1.4. Status of the VM for which alert is generated is recommended to *suspicious*.

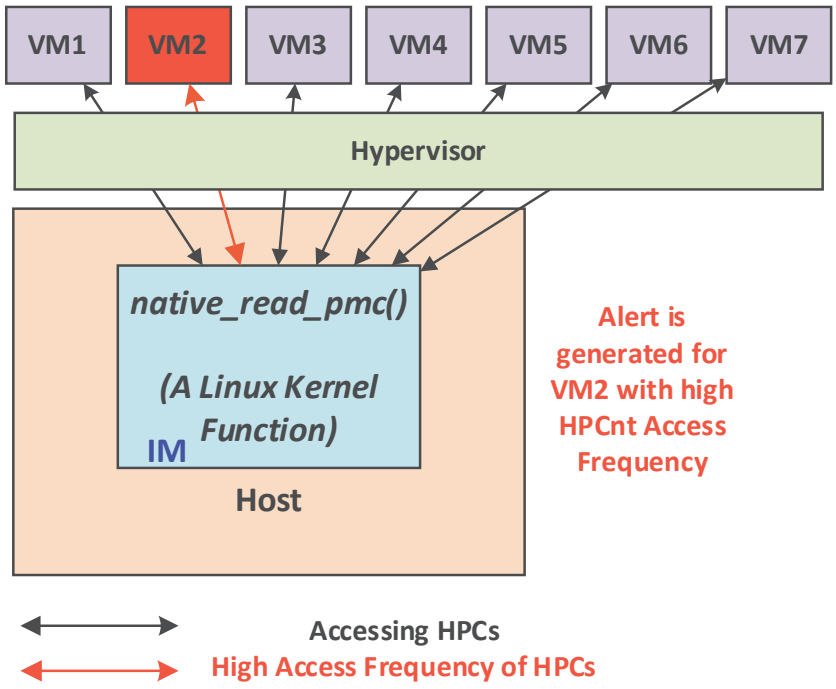


Figure 6.15: Interrupt Monitor : Functional Representation

When an alert is generated either from CAM or BM, along with an alert from IM also, the status of the VM is changed to *spy*. An advance trigger is sent to the Computer Emergency Response Team (CERT) on identifying the *spy* VM. However, the fourth monitor, Network Monitor, is turned on for this *spy* VM mainly for the confirmed detection of the BPA attack.

- Network Monitor (NM)

The primary function of NM is to monitor specific VM to track whether any packet trapping activities are carried out by it, as shown in Figure 6.16.

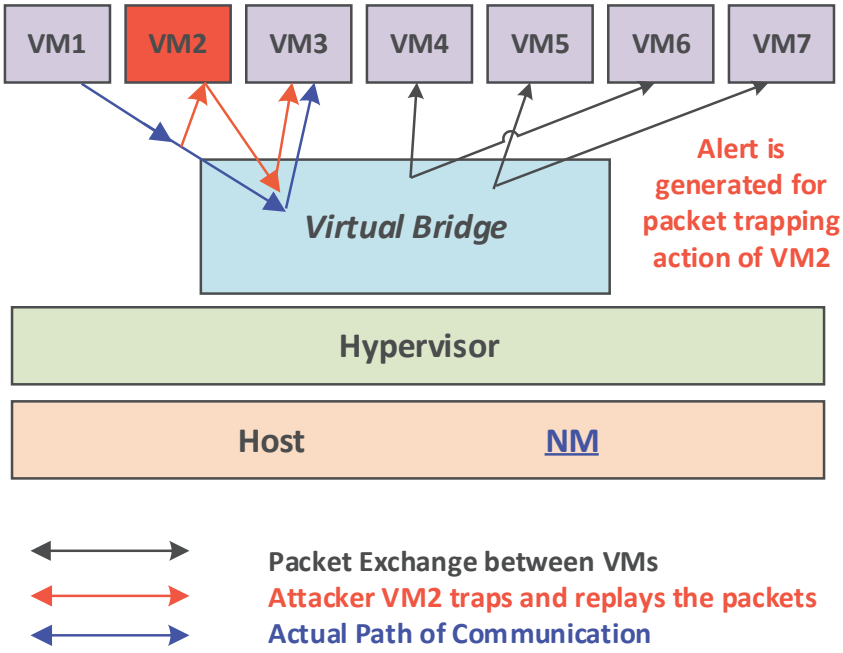


Figure 6.16: Network Monitor : Functional Description

The specific VM is the one for which an alert is generated by both CAM/BM and IM, i.e., *spy* VM. The way that is employed by the NM to trap packets also depends on the type of applications. Data of FTPS and HTTPS can be trapped over the network as well as from the disk, while the data of a secure Video Conferencing can be trapped only over the network. Detection of such activity can be carried out by manual, automatic, and semi-automatic methods. Among them, the semi-automatic model is the most popular as the manual methods are difficult to implement, and the automatic methods are resource hungry. The semi-automatic method waits for a trigger from external ways and means to start the required automatic method. The third eye of our architecture, NM, helps the security team to trigger for their semi-automatic method activated.

We have launched the ARP spoofing method to trap packets and have used the ARP spoofing detection module as a plug-in in the code of NM to prove its effectiveness. However, the security team of the system can replace it with the semi-automatic method in use, as shown in Figure 6.17.

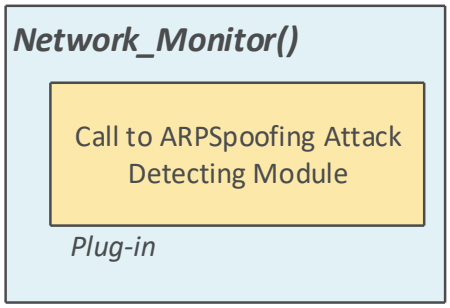


Figure 6.17: NM Implementation

Considering ARPSpoofing for trapping packets, there would be multiple entries of the MAC address in the MAC table for the *spy* VM. If so, then the *spy* VM has certainly performed packet trapping, and it is declared as *malicious*. It will raise alert to the Computer Emergency Response Team (CERT) about *malicious* activities for preventing leakage of secured data.

The logical flow of attack detection method employed by *Chaturdrashta* is presented in Figure 6.18. Additionally, *Chaturdrashta* is partitioned into two sub-solutions, *Trilochan* and *Trinetra*, for detecting the presence of DTA and TDA, respectively, as shown in Figure 6.19. A brief discussion on *Trilochan* and *Trinetra* is given in the following subsections.

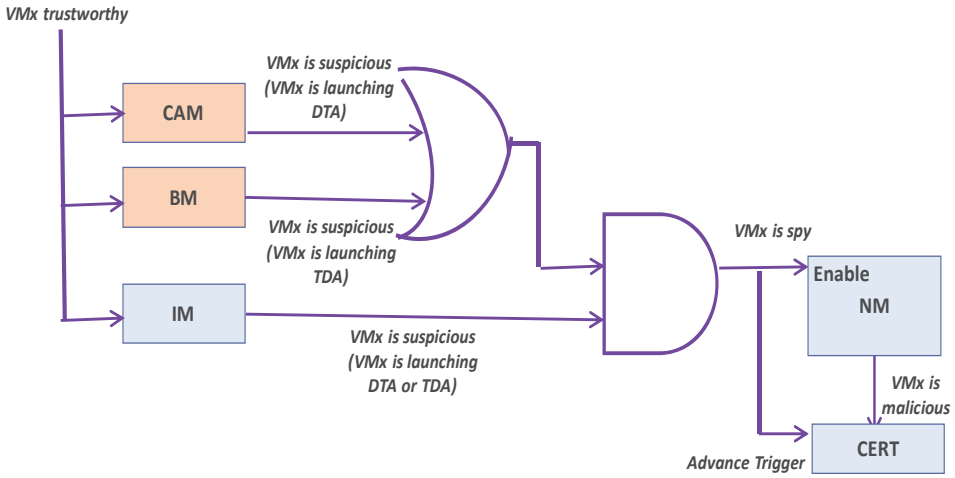


Figure 6.18: Logical Flow of *Chaturdrashta*

Like *Chaturdrashta*, *Trilochan* and *Trinetra* are also Sanskrit words, and the meaning of both the names is 'three eyes'. The names are given, considering that the approaches

require monitoring of three actions. The logical division of *Chaturdrashta* into *Trilochan* and *Trinetra* is shown in Figure 6.19. As per the performed actions during each of two attack processes, *Trilochan* is comprised of CAM, IM, and NM. *Trinetra* replaces CAM by BM, and thus BM, IM, and NM are the three primary components of *Trinetra*.

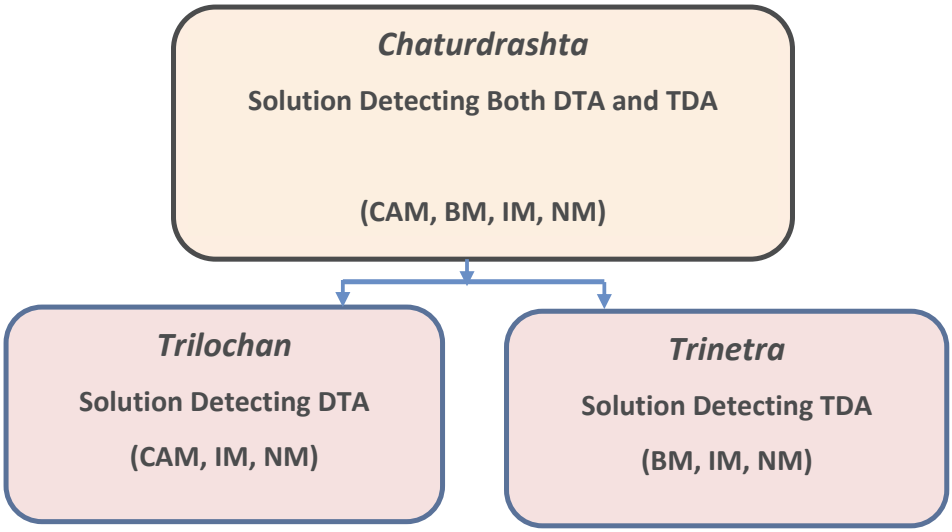


Figure 6.19: ***Chaturdrashta : Trilochan + Trinetra***

6.2.2.1 Trilochan: Solution to Detect Direct Timing Attack

The proposed solution *Trilochan* detects the presence of Direct Timing Attack on Cross-VM platform. As discussed previously, frequent access to cryptographic library and HPCnts are the primary actions performed during the launching process of DTA. Hence, CAM and IM are the parts of *Trilochan* to observe the above-mentions actions. Moreover, NM is also required to be included as the third component of *Trilochan* for observing the actions taken by the attacker after extracting the private key. Accordingly, it is comprised of CAM, IM, and NM, where the function of each of the monitors is discussed previously. The model and the flow of the proposed solution are shown in Figure 6.20 and Figure 6.21, respectively. As per the requisite for Cross-VM DTA, both the target and compromised VMs are assumed to share the cryptographic library loaded on a shared VM, as shown in Figure 6.20. CAM, IM, and NM are shown in the diagram to point to the area in the cloud environment they observe.

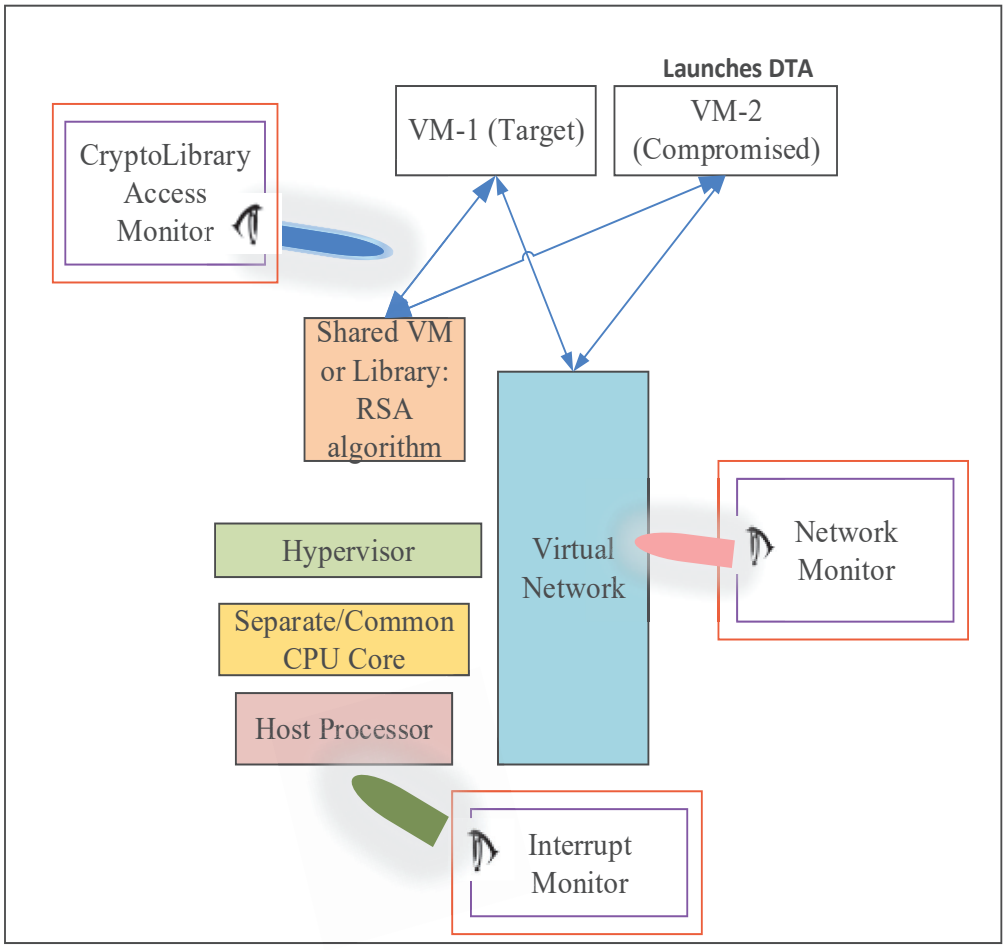


Figure 6.20: Model of *Trilochan*

The flow of *Trilochan*, shown in Figure 6.21, represents a step-by-step procedure that the *Trilochan* follows to detect the presence of a compromised VM initiating Cross-VM Direct Timing Attack.

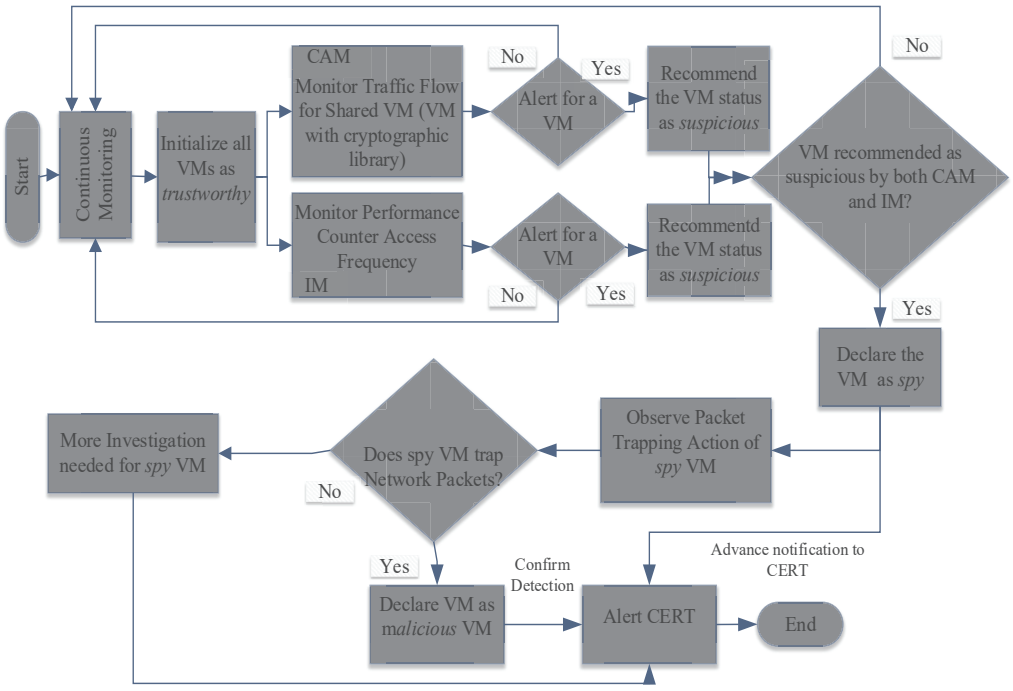


Figure 6.21: Flow of *Trilochan*

As shown in Figure 6.21, the initial status of all the VMs is set as trustworthy. Both CAM and IM are set to observe the activities of all the running VMs. When CAM generates an alert for a VM, the status of that VM is recommended as *suspicious*. Simultaneously, IM also keeps track of HPCnt access frequency of all the processes. If IM generates alert for a VM, then the status of that VM is also recommended to be *suspicious*. If both CAM and IM recommend the status change for the same VM as *suspicious*, then the status of that VM is set to *spy*. Although there are rare chances of a legitimate process to be declared as *spy*, NM is initiated mainly for the means of confirming detection. As soon as a VM is declared as *spy*, an advanced trigger is sent to the CERT. The NM is turned on at the same time. NM is turned on to check whether the *spy* VM is initiating any packet trapping action. If the *spy* VM is caught by NM also, then the status of that VM is changed to *malicious*. Further, a trigger for a confirmed detection is sent to CERT for the *malicious* VM. In case if the NM does not catch the *spy* VM, then more investigation is suggested.

Within a very short span of time, *Trilochan* can successfully detect a VM launching Cross-VM DTA. Where *Trilochan* is meant for DTA, *Trinetra* can detect the BPA attack launched with the TDA method, which is discussed in the next subsection.

6.2.2.2 Trinetra: Solution to Detect Trace-Driven Attack

The proposed solution *Trinetra* detects the presence of Trace-Driven Attack. A comparison between the three primary actions employed during the launching of BPA and TDA reveals that the behavior of the attack launching VM differs only in the first action. Where the VM accesses the RSA process with high frequency during DTA, the VM results in a high BTB occupancy ratio during the TDA. Hence, among the three components of *Trilochan*, only CAM is replaced by BM to track the BTB occupancy ratio. Accordingly, *Trinetra* is comprised of BM, IM, and NM, where the function of each of the monitors is discussed previously. Figure 6.22, presents the model of *Trinetra*, where the flow of the same is shown in Figure 6.23.

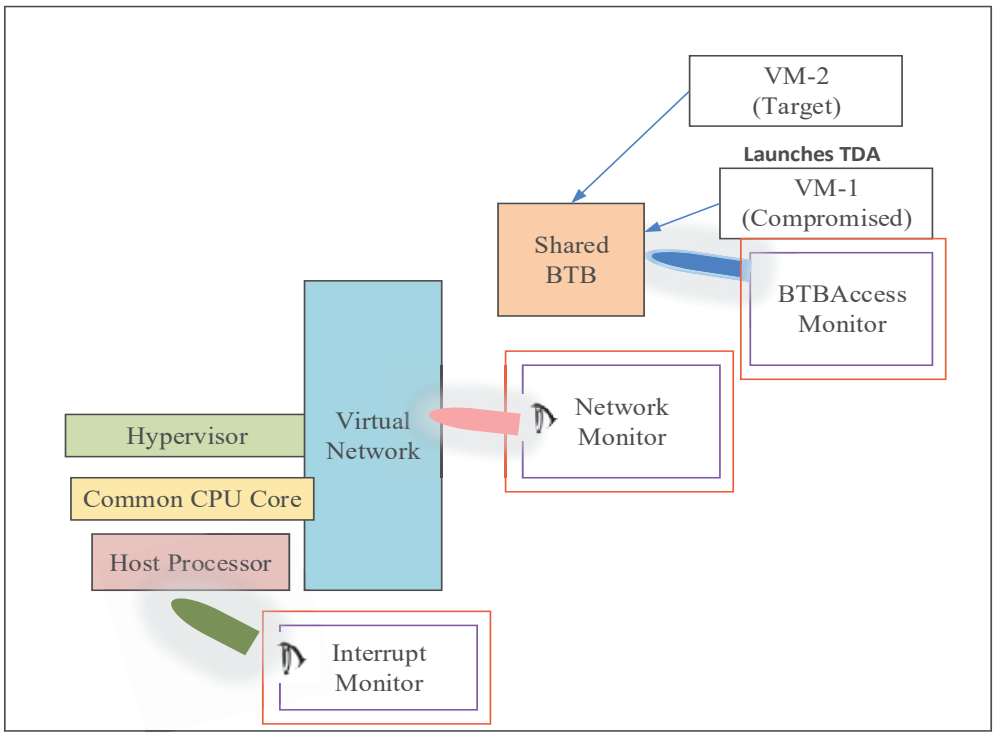


Figure 6.22: Model of *Trinetra*

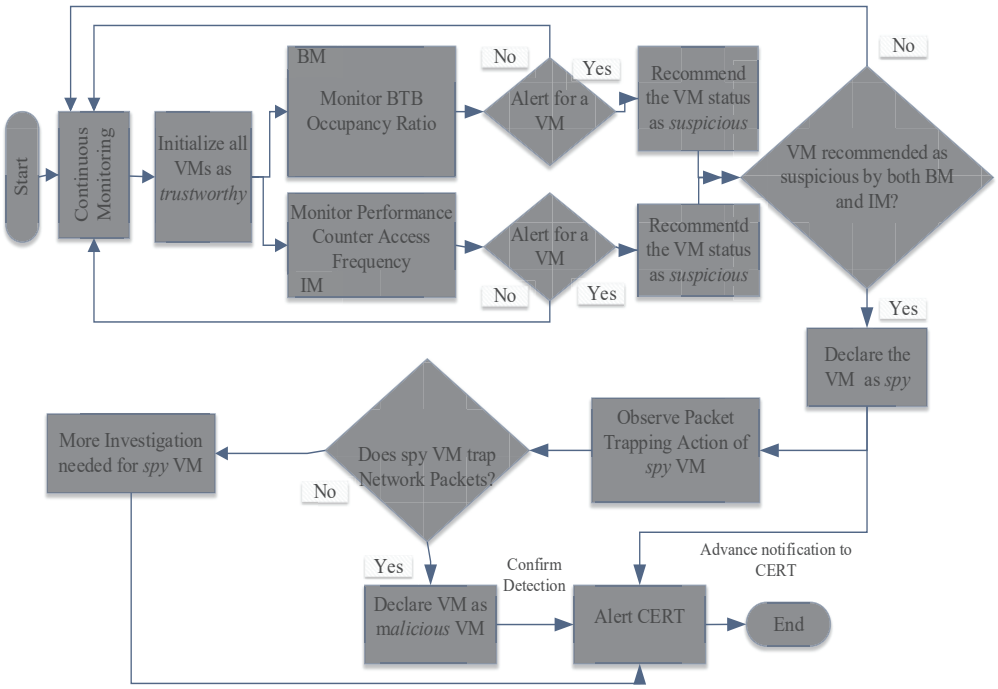


Figure 6.23: Flow of *Trinetra*

As shown in the diagram, the initial status of all the VMs is set as trustworthy. Both BM and IM are set to observe the activities of all the running VMs. When BM generates an alert for a VM, the status of that VM is recommended as *suspicious*. Simultaneously, IM also observes HPCnt access frequency of all the running VMs. Further, the status of the VM is changed to spy, if both BM and IM generate an alert for that VM. Like *Trilochan*, an advanced trigger is sent to CERT to inform about the spy process in addition to turning on the NM for monitoring the same spy VM. Further, the status of the spy VM is changed to *malicious* if NM finds it to perform packet sniffing. The *malicious* VM is finally blocked. Moreover, further investigation is suggested for the spy VM, for which the NM does not generate any alert. Even though any packet trapping activity is not detected from the spy VM, it should not be exempted as there are rare probabilities of a legitimate VM to be declared as a spy, i.e., to be caught by both BM and IM.

Chaturdrashta is implemented at the host level. Although the host-based attacks like VM Escape and host-based rootkits that may exploit the host-based module, it is difficult to compromise the code of Chatursrashta as it is incorporated in the kernel

source code. For overwriting or eliminating the Chatursrashta code, it is needed to recompile the kernel, which needs to reboot the system. A sudden system reboot without any means immediately generates an alert. Chatursdrshta does not apply the architectural or functional change, and hence it does not adversely affect the normal functionality of the system, and so the performance of the other processes does not get affected.

Combined approaches of *Trilochan* and *Trinetra*, i.e., *Chaturdrashta*, can detect the presence of a Cross-VM BPA attack launched with either DTA or TDA method. Moreover, *Chaturdrashta* was also found capable enough to detect the presence of Asynchronous and Synchronous BTB Eviction methods also as discussed in the next subsection.

6.2.2.3 *Trilochan* and *Trinetra*: Extended Scope to Detect of BTB Eviction Methods

Each of the sub-approaches of *Chaturdrashta* focuses on a specific type of method launching the BPA attack. As discussed in Sections 6.2.2.1 and 6.2.2.2, both *Trilochan* and *Trinetra* are comprised of three monitors to observe the frequency of the actions performed by DTA and TDA spy processes, respectively. However, identification of the primary actions of the BTB Eviction methods and their resemblance with that of DTA and TDA infers the extended scope of *Trilochan* and *Trinetra* in detecting these Methods.

As represented in Subsection 6.2.1, the primary actions of the Asynchronous BTB Eviction method are the union of the primary actions of both DTA and TDA. This resemblance reveals that, although *Trilochan* and *Trinetra* are designed to detect the Cross-VM DTA and TDA, respectively, they both can also detect the presence of the Cross-VM Asynchronous BTB Eviction method in their direct form.

As discussed in Subsection 6.1.4, it is difficult to implement the Cross-VM Synchronous attack practically. However, the proposed approach will detect its presence even if it is successfully launched. The resemblance of the primary actions of this Synchronous BTB Eviction attack with that of TDA clearly represents that *Trinetra* can detect the presence of this Synchronous BTB Eviction method in addition to the detection of TDA.

The behavioral description of *Chaturdrashta* theoretically proves its efficiency in detecting the BPA attack irrespective of the underlying attack methodology. However, we also prove its functional-practical efficiency by carrying out experimental analysis of all the four different components. Among the four attack methods, we consider the three methods except for the Synchronous BTB Eviction method for the experimental analysis as we have taken its simulation as the future work, as discussed in Subsection 6.1.4. The Implementation details and observed results are provided in the next chapter.

6.3 Summary

Attack simulation plays a vital role in revealing the practical issues related to the attack launching methods. We found simulation of BPA attack in [120][125] and [128] where a non-virtualization environment was considered. We have simulated the BPA attack with the DTA and TDA in virtualization as per the environment targeted in our work. It is proved that attack is possible in the commonly used and defacto standard of the Hybrid virtualization environment. Additionally, partial simulation of the Asynchronous BTB Eviction method was also performed, but we identified a need to study the BTB Eviction methods from two aspects: (1) to simulate Synchronous BTB Eviction method, which is a bit complex, and (2) to analyze the scope of both the methods in the presence of Montgomery Ladder algorithm. Considering time constraints, the identified work related to the BTB Eviction methods is taken as the future work of our research. However, we consider them while designing the new approach in detecting their presence.

The attack detection approach needs a detailed analysis of attack launching methods for the behavior study. Careful observation and analysis of procedures launching DTA and TDA highlight the primary actions carried out by the spy process. It was found that the DTA process access the shared cryptographic process (here, RSA) with a very high frequency. On the other hand, the TDA process results in very high BTB occupancy. Both the attack launching procedures also employ a common action of reading of Hardware Performance Counter with considerably high frequency. Behavior analysis of BTB Eviction methods in a similar fashion revealed an absolute resemblance to the primary actions of the Asynchronous method with those of DTA as well as TDA. The resemblance of the primary actions was also found between the Synchronous attack and TDA. It is with reference to the similarity in simulation

steps between DTA and Asynchronous BTB Eviction method and TDA-like dummy process in both the methods.

Although a process or VM performing the above-discussed activities cannot be normal, we found it necessary to consider the post-attack actions mainly for a confirm detection. A fruitful BPA attack needs to trap packets by launching attacks like MITM [134] for extracting the confidential messages that are encrypted with the extracted key. In the case of non-availability of the packets, the extracted private key does not provide any useful information. Thus, packet trapping can be considered as an essential post-attack action to be performed by the compromised VM. Identified four primary actions (three during DTA/TDA/BTB Eviction Methods + one post-attack) are considered in designing the new approach to detect the presence of a Cross-VM BPA attack.

As the main contribution of our work, a four-eyed approach *Chaturdrashta* was proposed to observe the total of four primary actions of DTA and TDA. CryptoLibrary Access Monitor (CAM), BTBAccess Monitor (BM), Interrupt Monitor (IM), and Network Monitor (NM) are the four monitors that form *Chatursrashta*. CAM, BM, IM, and NM are designed to observe the four primary actions performed during DTA and TDA, i.e., packet traversal frequency, BTB occupancy, HPCnt access frequency, and initiation of packet trapping activity, respectively. *Chaturdrashta* observes the combined actions performed during both DTA and TDA, which is further logically divided into two sub-approaches, *Trilochan* and *Trinetra,* to detect the presence of DTA and TDA, respectively. As per the underlying actions performed during the respective attack process, *Trilochan* is comprised of CAM, IM, and NM, and *Trinetra* is the composition of BM, IM, and NM. The operating mode of both approaches is almost similar. CAM (in the case of *Trilochan*), or BM (in the case of *Trinetra*) is started with the initial status *trustworthy* for all the VMs. IM also runs simultaneously with CAM/BM. If CAM/BM generates an alert for a VM, then the status of that VM is recommended as *suspicious*. If the IM, running simultaneously, also generates alert for the same VM, then the status of that VM is changed to *spy*. With an advance trigger to CERT regarding the *spy* VM, NM is also turned on to check whether any packet trapping activity is initiated by the *spy* VM. A confirm detection signal is raised on finding the packet trapping activity from the *spy* VM where it is declared as the *malicious* VM. *Malicious* VM is then blocked.

The resemblance of primary actions performed during BTB Eviction methods with that of DTA and TDA methods implicitly represent the broader scope of *Chaturdrashta*. It reveals that, although designed to detect DTA and TDA, *Chaturdrashta* can detect the presence of the BPA attack even when the Asynchronous or Synchronous BTB Eviction methods are employed to launch the attack.

Both *Trilochan* and *Trinetra* generate alert only for the processes that access HPCnt with very high frequency as well as either result in high packet traversal with the shared VM or occupies BTB with a high ratio. There are rare chances where a legitimate process performs such action with that much high frequency. Still, it is detected as *malicious* only when it is found to perform packet trapping action, leading to the elimination of the false positive rate. The proposed attack detection mechanism does not manipulate any architecture component as well as it is independent of the type of victim cryptographic algorithm.

CHAPTER

7 Experimental Analysis

CryptoAccess Library Monitor, BTBAccess Monitor, and Interrupt Monitor are designed to generate an alert message when they find that the actions they observe cross a certain threshold. For an improperly identified threshold value, CAM, BM and/or IM may generate an alert message for a legitimate process even though it is performing the respective action(s) in a normal fashion. In such circumstances, the identification of an appropriate threshold value is quite significant for reducing the scope of false positive. A repetitive practical analysis is required to carefully choose the threshold values of the three components, CAM, BM, and IM.

In this chapter, we have described the results obtained from the experimental analysis carried out for setting the threshold values for all the three monitors. Further, simulation of NM is done to test how the *spy* VM can be monitored to check whether it initiates any packet trapping actions. As discussed in Subsection 6.2.2, CAM is one of the components involved in the detection of DTA and Asynchronous BTB Eviction Attack (ABEA), where BM is involved in the detection of TDA, ABEA as well as in the detection of the Synchronous BTB Eviction Attack (SBEA). IM and NM, play an active role in the detection of all the four attack launching methods. In this regard, discussion on the experimental analysis is carried out on each of the four components with reference to the type of attack(s) detection approach they are parts of. Although *Chaturdrashta* is capable of detecting the presence of BTB attack irrespective of the underlying method, we do not include Synchronous BTB Eviction Attack in our present research. It is regarding the fact that the simulation plays a vital role in performing the experimental analysis, where the simulation of the SBEA is the future work of the research.

In Section 7.2, the performance of the proposed solution is evaluated for comparing its scope with other existing solutions as well as to perform an overhead calculation

7.1 *Chaturdrashta* : Simulation and Result Analysis

Chaturdrashta is a four-eyed approach with four components that work independently but also integrated for a specific means of BPA attack detection. The simulation of each of the four components is carried out to analyze the obtained results mainly for defining the appropriate threshold value, as discussed previously. The simulation setup and result analysis of CAM, BM, IM, and NM are discussed in subsequent subsections of this section.

7.1.1 Simulation Setup

Practical implementation and result analysis of *Chaturdrashta* needs to be carried out in accordance with the simulation of DTA, TDA, and ABEA. Hence, the simulation setup employed during the attack simulation (Section 6.1) is considered while carrying out an experimental analysis of *Chaturdrashta* also. Section 6.1 describes the entire scenario of the BPA attack. It provides a step-wise description of how and when the attack is launched. The experiments were performed with VMs configured with KVM Hypervisor, where the host Operating System (OS) is Ubuntu 15.10. The same version of OS was used as a guest OS also. The simulation was performed mainly with two VMs: one victim and one compromised VM. An additional VM was required for DTA and ABEA that stores the shared cryptographic library.

Experiments on each of the three components (CAM, BM, and IM) were performed in two steps. During the first step, each component was simulated in the absence of the BPA attack launching the spy process. Observations collected from the experiments were applied to define the threshold value for each of the components. Further, the experiments were repeated in the presence of the attack launching spy process as the second step. Defined threshold values were revised in accordance with the results obtained in the second step for eliminating the scope of the false positives. Experimental analysis of CAM, BM, and IM as per the above-described flow is discussed in Sections 7.1.2, 7.1.3 and 7.1.4 respectively following by the implementation details of NM in Section 7.1.5.

7.1.2 CryptoLibrary Access Monitor

As discussed in section 6.2.2, CAM was set to monitor packet traversal between the shared VM and all other running VM. In our experiment, we had initiated secure VC, secure FTP, and an application profiler on VMs. We measured the total number of packets traversed for these applications at regular intervals using *tcpdump* tool. The obtained results are plotted in Figure 7.1. The x-axis of the plot represents the duration for which monitoring took place, where the y-axis represents the obtained packet-count. Results in Figure 7.1 reveal that the application profiler does not access the cryptographic library available on the shared VM. Even for secure VC and SFTP also the maximum observed packets are below 130, i.e., SFTP results into 128 packet traversals in the period of 50 sec, where only 112 packets are traversed within the time duration of 70 sec for the secure VC. Traversal of 150 packets within the time duration of 65 sec can be considered as the threshold for the CAM considering the packet traversal rates of various processes. CAM is set to generate an alert when it finds a process resulting in packet traversal with values much more than the defined threshold.

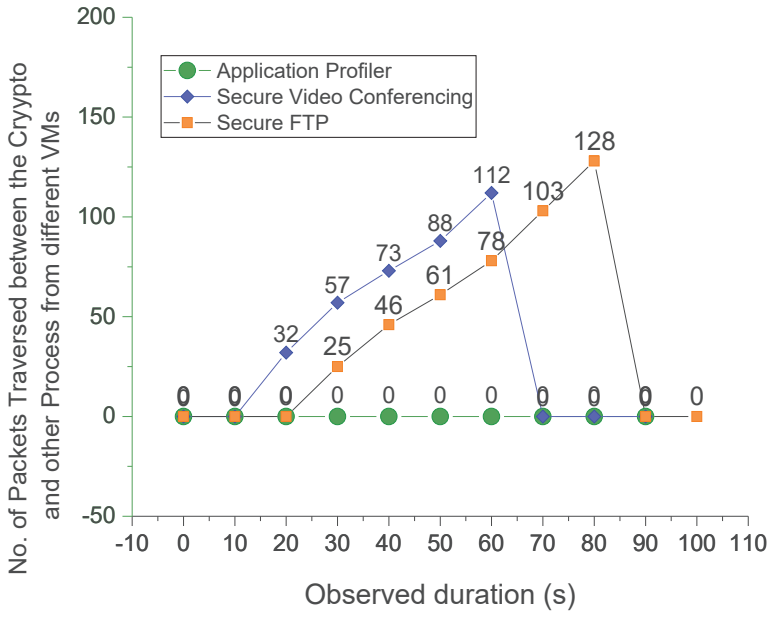


Figure 7.1: Results of Packet Monitoring

Further, the same experiment was done by running a DTA process in VM2 in addition to the previously defined processes. The results observed by CAM are shown in Figure 7.2. The results represent very clearly that the packet traversal due to the DTA process is around 15-20 times more than that of the other running processes.

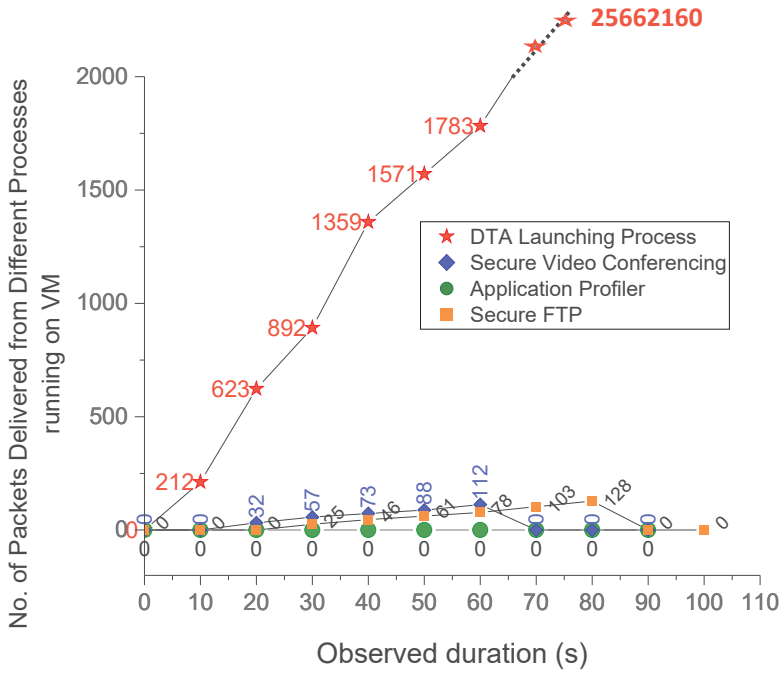


Figure 7.2: Results for Packet Monitoring along with DTA process

The performance of CAM was tested even by initiating the DTA-like Online phase of ABEA, where the experiments were carried out around 50 times. The obtained results are plotted in Figure 7.3, which is almost similar to the results of DTA. The observed packet traversal frequency is shown in Fig. 7.3 reveals that a considerable difference between the normal Linux processes and the ABEA process exis as in the case of DTA. As observed in Figure 7.2 and Figure 7.3, we get almost similar results for DTA and ABEA, and hence, we consider only DTA for further experimental analysis related to CAM.

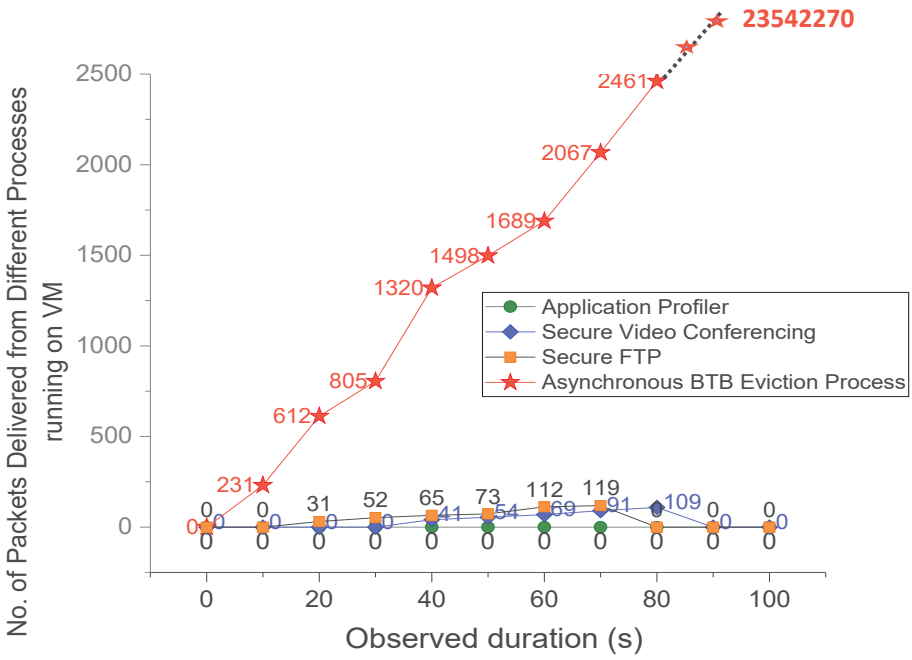


Figure 7.3: Results for Packet Monitoring along with ABEA Process

In another experiment, a VC was initiated with varying speeds, where results obtained for packet traversal during an interval of 10 minutes are shown in Figure 7.4. We are using *Trueconf* software for VC, and the maximum speed offered by it is 10Mbps. Hence, we took different communication speeds as 1,2,4,8 and 10 Mbps for our experiment. As shown in the figure, the observed results of the packet traversal for varying communication speed of VC are compared with that of DTA. As shown in the plot, the packets generated by VC depend on the allotted bandwidth for the required quality. Even in case of the highest quality standard of 10 Mbps, it generated 380553 packets in 10 minutes. On the other hand, DTA found generating more than 10 million packets within that much time duration.

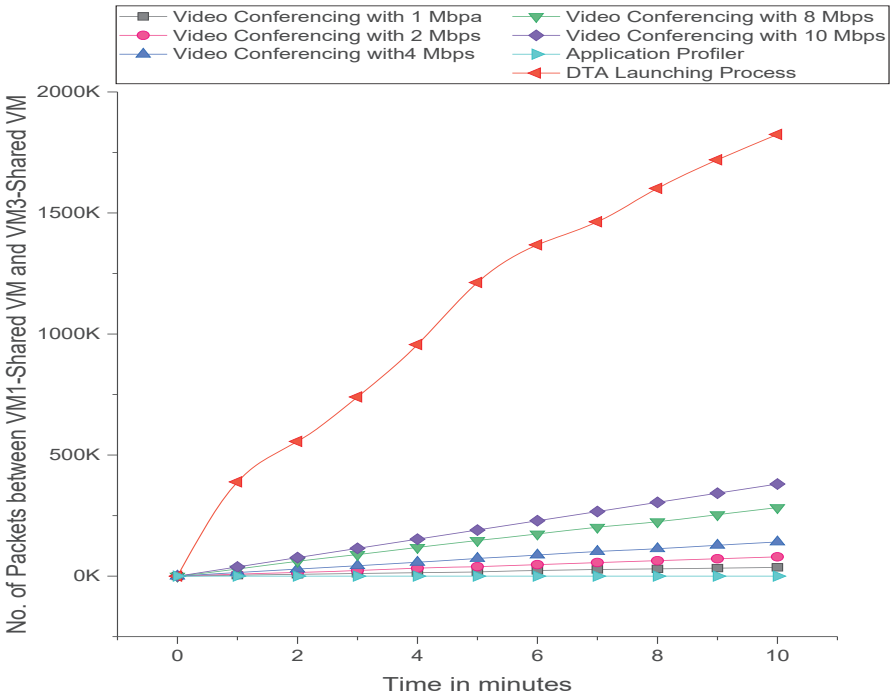
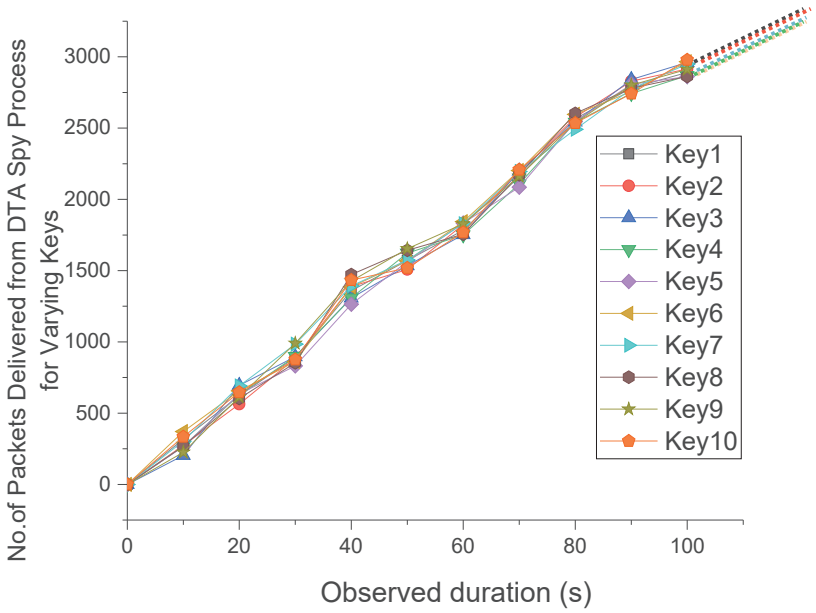


Figure 7.4: Results for Packet Monitoring for Video Conferencing with varying speed

Obtained results clearly reveal that the packet traversal rate between the VM running DTA spy process and the shared VM is very high compare to that of the other processes. As the next step, the packet traversal frequency of the spy process was also measured for around 30 key values. The result of randomly selected ten keys is plotted in Figure 7.5. The plot reveals that the packet traversal frequency is high in each case, irrespective of its key value.



,

Figure 7.5: Results for Packet Monitoring from DTA Spy Process for Varying Keys

The threshold value of CAM was set to 150 packets within the time duration of 65 seconds, as discussed previously. The threshold was decided as per the results observed for the three processes we executed during our implementation, i.e., Secure VC, Application Profiler, and Secure FTP. Other benign processes may result in even more amount of packet traversal where the value would be obviously less than that of the DTA process. To prevent such benign process from wrongly getting alert from CAM, the threshold value can be increased to the value where only the DTA/ABES process can fall beyond. That is, we have increased the threshold to 1800 packets per 65 seconds (i.e., around 30 packets per second). All the processes crossing the criteria would get alert, and their status would be recommended to be *suspicious*.

7.1.3 BTBAccess Monitor

Observation of BTB occupancy is the main role of BM. Unlike CAM, BM does not require a shared cryptographic library but requires a common CPU core (in turn, common BTB) between the victim and the spy processes.

BM is meant to observe BTB occupancy of running processes, which is one of the three primary actions of TDA. As the attack simulation of TDA was done with the Gem5 simulator as discussed in Section 6.1.2, we used the same for performing experiments of BM for integrity. Gem5 is a modular platform for research based on computer-architecture, including system-level and processor architecture.

The experimental setup for BM is comprised of two VMs with shared BTB. Parameters related to branch prediction were set, as mentioned in Table 6.2. Gem5 simulator provides output in *stats.txt* file. Observed values of all the performance parameters can be obtained from this file. Details of parameters like number of BTB entries, number of branch instructions, branch hits and miss and number of misprediction BTB occupancy value of the executing process, can be obtained for an executing process from the stat.txt file. Among them, total BTB entries, number of branches, and number of hits and misses can help in calculating the BTB occupancy ratio of a process.

An experiment was carried out to calculate the BTB occupancy ratio of different MiBench [135] programs for around 50 times. The plot in Figure 7.6 represents the minimum, average, and maximum of the observed rates. The plot reveals that among all the MiBench programs, jpeg has the highest average occupancy, which is 0.4. Accordingly, the threshold value for the BM was set to 0.4.

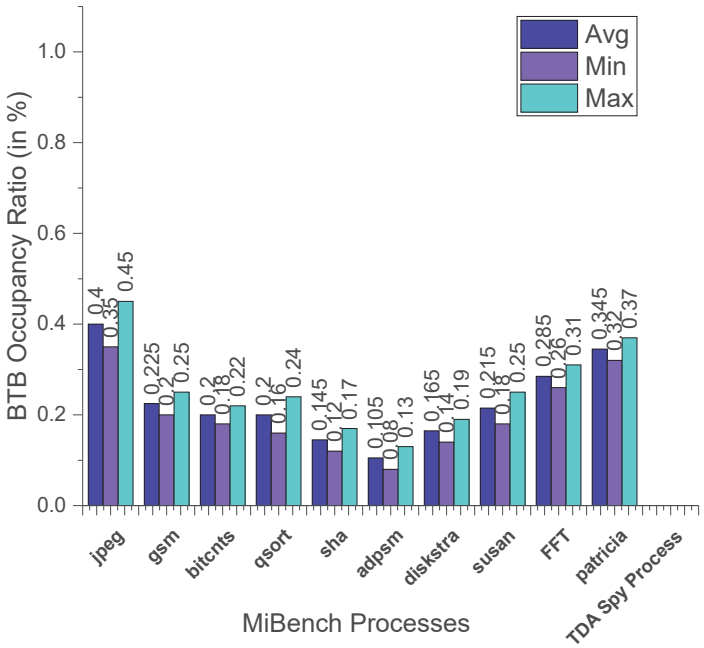


Figure 7.6: BTB Occupancy Ratio observed by BTBAccess Monitor

As discussed in Subsection 6.2.2, the spy process of TDA can be caught by the BM. Moreover, the dummy process of the ABEA also works like the spy process of the TDA. Hence, BM should be able to detect the presence of the ABEA dummy process also in addition to TDA. For observing the behavior of the dummy process of ABEA, it was separately executed on the Gem5 simulator, where the partial simulation of the entire ABEA process was performed by writing code in Java. The experiments to calculate the BTB occupancy were further performed on these two processes in addition to the MiBench processes.

The results observed for BM are shown in Figure 7.7. As discussed previously, the threshold value for BM was set to 0.4, considering the highest value of the BTB occupancy ratio observed in Figure 7.6. However, we increase that value to 0.8 for a safer side. The value is decided based on the average occupancy ratio observed for the TDA spy process (which is 0.87) and that of the ABEA dummy process (which is 0.92) as per Figure 7.7. Accordingly, 0.8 is set as the threshold value for generating alert by the BM. As soon as the BTB occupancy of a process crosses value 0.8, BM generates an alert. The VM running this malicious process is declared as a *suspicious* VM.

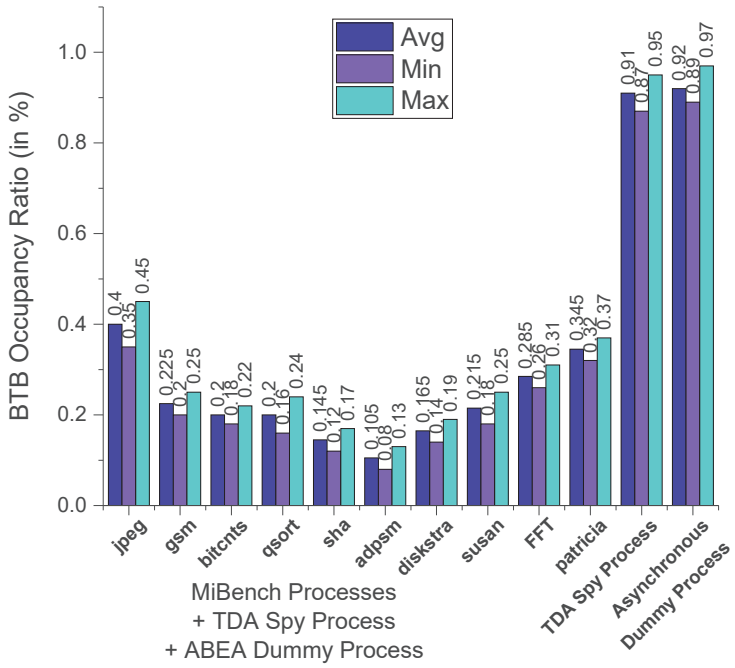


Figure 7.7: BTB Occupancy Ratio along with TDA Spy Process and ABEA Dummy Process

For the above experiments, the total BTB entries were set to 4096 (4KB) size. We repeated the experiments of BM with varying sizes of BTB. For each of the MiBench program and the TDA spy process, average values of BTB occupancy ratio was calculated for a range of BTB size varying from 3KB to 8KB. A plot is shown in Figure 7.8 to represent the observed values. It was found as the size of BTB increases, the occupancy ratio of MiBench processes decrease, which is obvious as the size of the programs does not get changed. The spy process launching TDA as well ABEA dummy process is written carefully based on the size of BTB so that they can fill it entirely. Hence, no major change in the occupancy ratio was observed by setting BTBentries to different values. As shown in the diagram, the difference between the result obtained from the spy/dummy processes and that of other MiBench programs increase as the size of BTB increase.

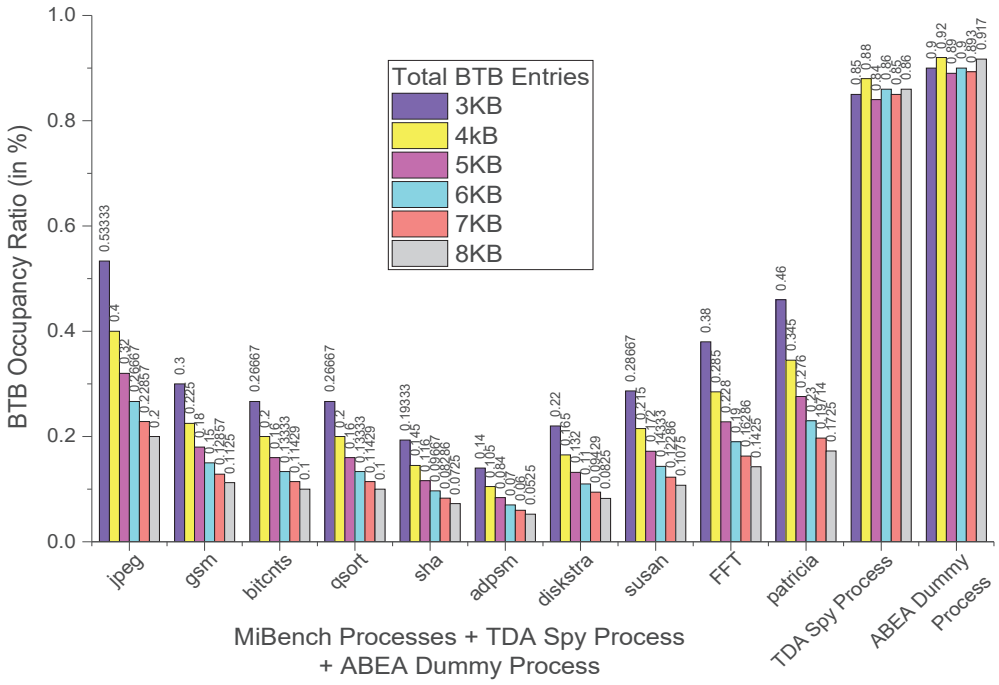


Figure 7.8: BTB Occupancy Ratio for Varying BTB size

The above representation reveals that the difference in BTB occupancy ratio between MiBech processes and that of the TDA spy process decreases as the size of BTB decreases. In such circumstances, any of the benign processes may cross the threshold value, and BM generates alert for it. Such a situation can become possible, but as discussed previously, that process would be declared as a *spy* only if IM also generates alert for the same process. There are effectively rare chances that exist for such a situation to arise. Still, even if they arise in the worst case, the process will never be caught by NM, which gives the confirmed signal of detection.

7.1.4 Interrupt Monitor

In concurrence to the BTBAccess Monitor, Interrupt Monitor keeps a continuous watch on performance counter access frequency. The code added to the kernel function *native_read_pmc()* makes a kernel log when the counter for a process hits the predefined threshold.

Initially, the threshold was set to value 10. Experimental analysis was initially carried out on sample programs of MiBench, but we did not find kernel logs for any of the programs for threshold value 10. However, we found logs for other Linux processes running in the foreground/background and so results were collected for them for analysis. For threshold value 10, we found repeated kernel logs for some of the running processes where the time difference between two successive logs was considered as a parameter for plotting the results. Figure 7.9 plots the results where the x-axis represents the timestamp in seconds when a process hits the threshold value, and the corresponding value in the y-axis represents the exponential cumulative frequency with which the HPCnts are accessed by various running processes. Frequency is calculated by taking the inverse of the time difference between the last two successive logs. As shown in the diagram, we found logs for various benign activities like software-updater, web-content, and Xorg. The results reveal that the exponential cumulative frequency is between 0.01 to 1 for benign Linux processes. We carried out the experiment for around 100 times, and we obtained similar results for all the runs. Accordingly, we found that 1 can be set as the threshold value for the Interrupt Monitor.

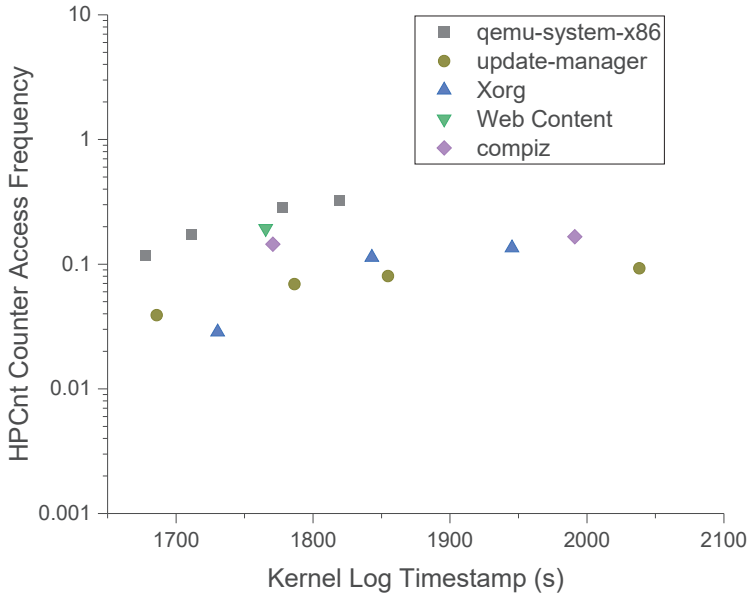


Figure 7.9: Observation by IM for Counter Threshold 10

As the reading of the HPCnt is one of the core actions of all the BPA attack launching methods, the results of IM were observed in the presence of the VM launching DTA, TDA,

and ABEA launching processes. Obtained results shown in Figure 7.10, Figure 7.11, and Figure 7.12 reveal that where none of the legitimate processes cross the value 1 for the exponential cumulative frequency, the same is found to be more than 1000 for the VM running DTA and ABEA. The measured frequency was found more than 100 for the TDA spy process. By taking the results of the thee attack launching processes into account, the previously decided threshold value 1 for IM is increased to 100 (minimum of 1000 and 100) for the means of eliminating the scope of false positive.

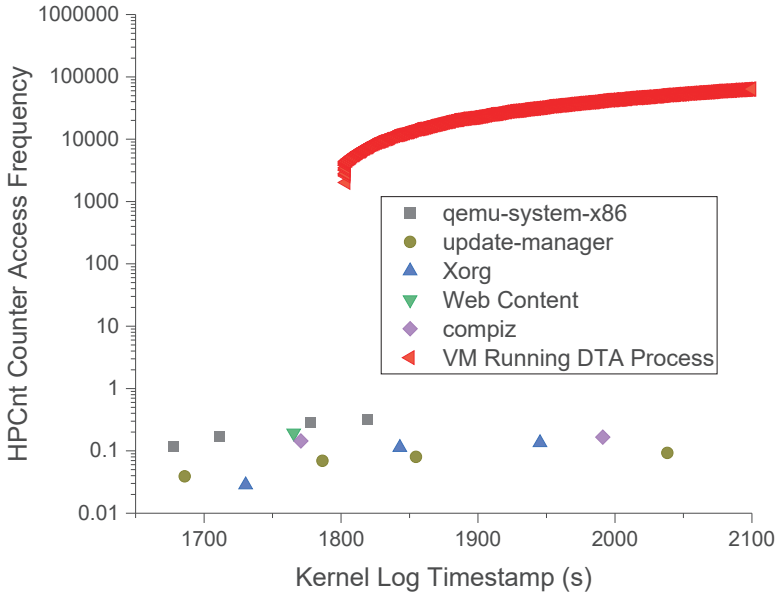


Figure 7.10: Observation by IM for Counter Threshold 10 (along with DTA launching process)

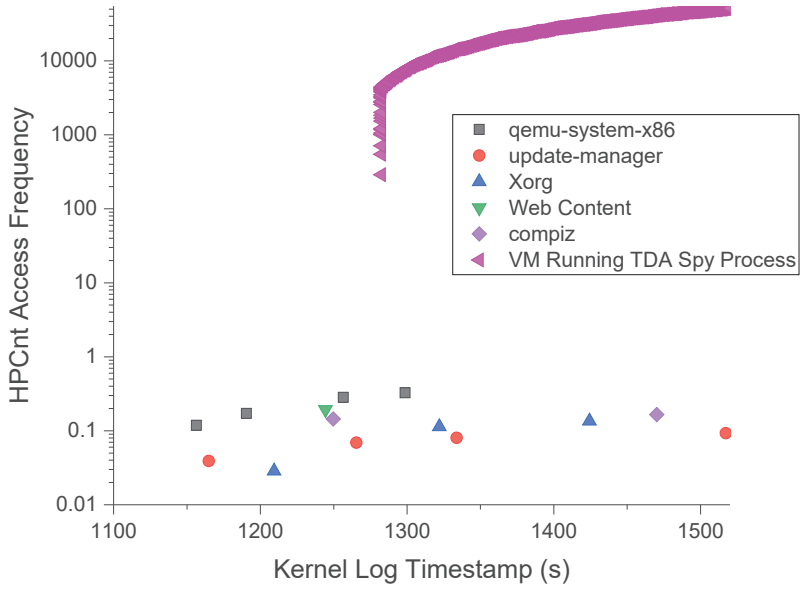


Figure 7.11: Observation by IM for Counter Threshold 10 (along with TDA launching process)

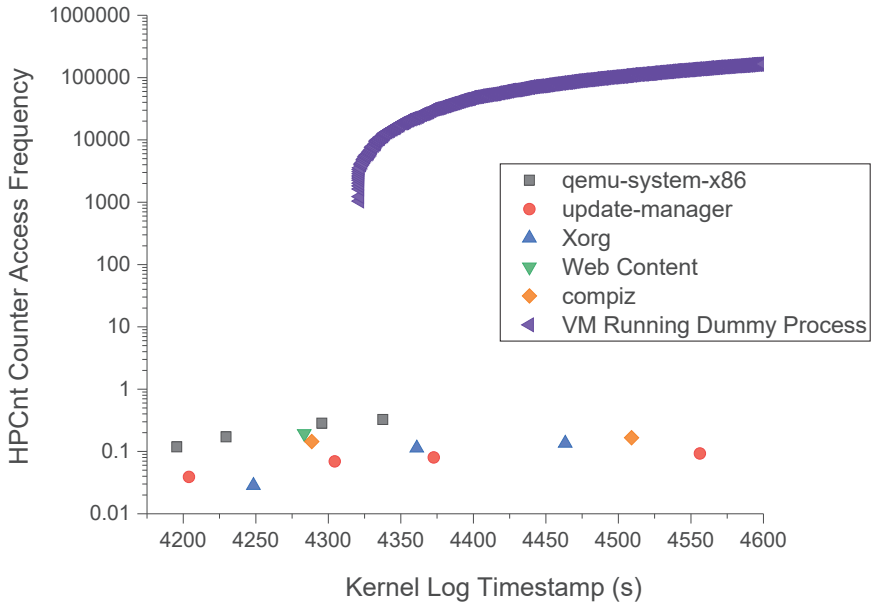


Figure 7.12: Observation by IM for Counter Threshold 10 (along with ABEA Process)

Obtained results are also plotted to represent the time taken in hitting the counter on every hit. For the counter threshold value 10, time is taken at every hit, i.e., when the counter hits values 20, 40, 60, etc., are plotted in Figure 7.13. As shown in the diagram, the time taken to

reach the threshold value is much more for the processes other than the DTA, TDA, and ADEA spy/dummy processes. The spy process of DTA and ABEA hit the threshold with even faster than that of the TDA process. A prominent difference can be observed between the behavior of the spy processes and that of other normal processes.

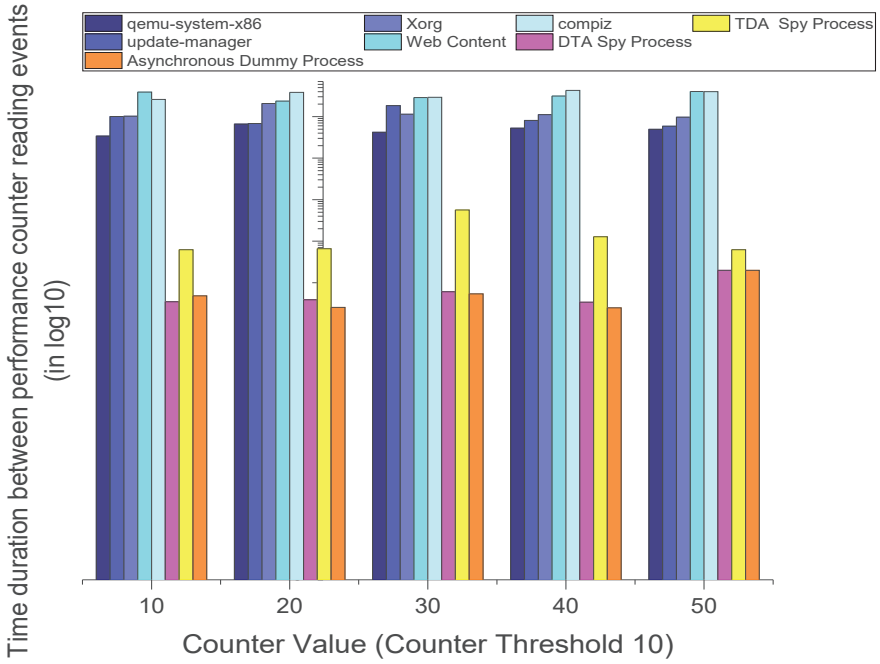


Figure 7.13: HPCnt Access Frequency for Threshold 10

Above mentioned experiment was repeated by setting the threshold with varying values, i.e., 20, 30, and 40. Respective obtained results are plotted in Figure 7.14, Figure 7.15, and Figure 7.16.

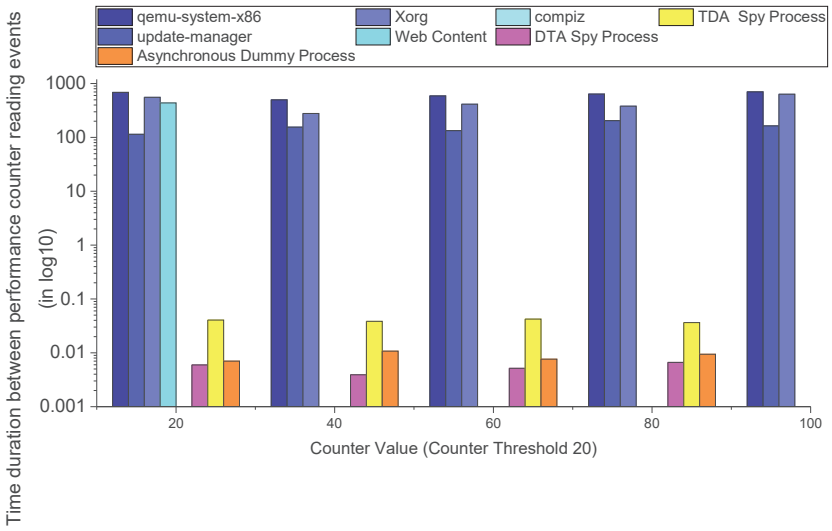


Figure 7.14: HPCnt Access Frequency for Threshold 20

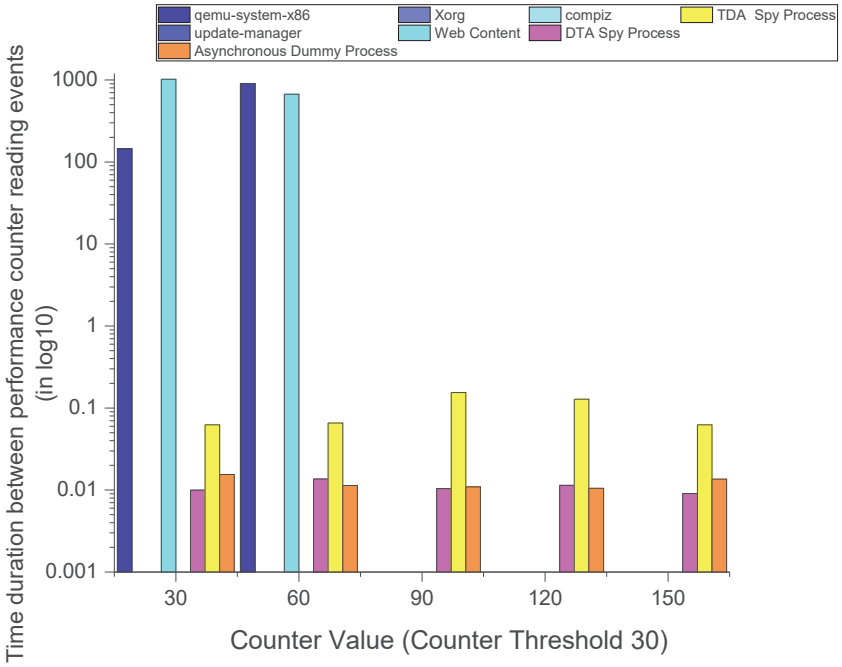


Figure 7.15: HPCnt Access Frequency for Threshold 30

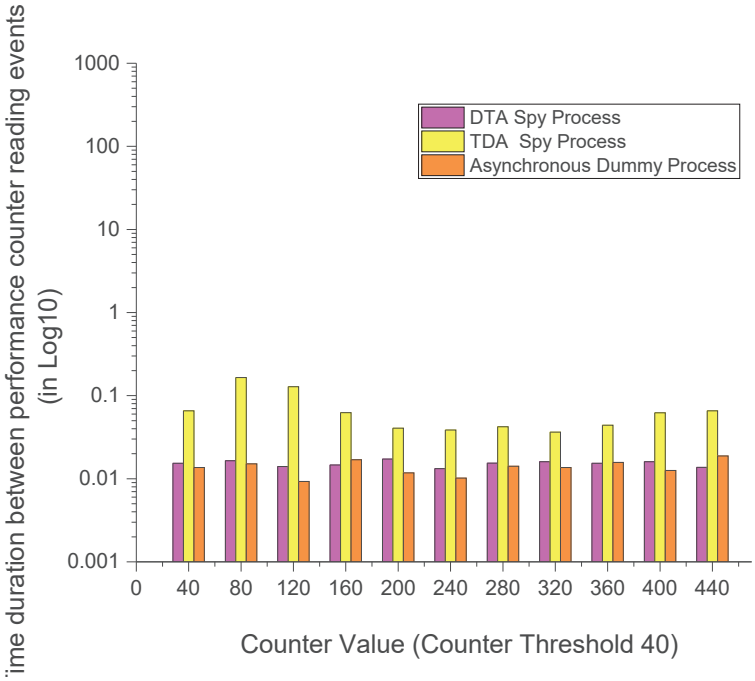


Figure 7.16: HPCnt Access Frequency for Threshold 40

Excessive kernel logs were found for that VM running spy process even when the counter was set to value 40, as represented in Figure 7.16.

We observed from the obtained results that as the counter threshold value was increased, the number of processes generating kernel log was decreased. Moreover, we did not find any kernel log for any of the legitimate processes when the counter threshold was set to value 40. Logs for only DTA, TDA, and ABEA processes were found with threshold value 40. Figure 7.17 summarizes the above discussion in graphical form.

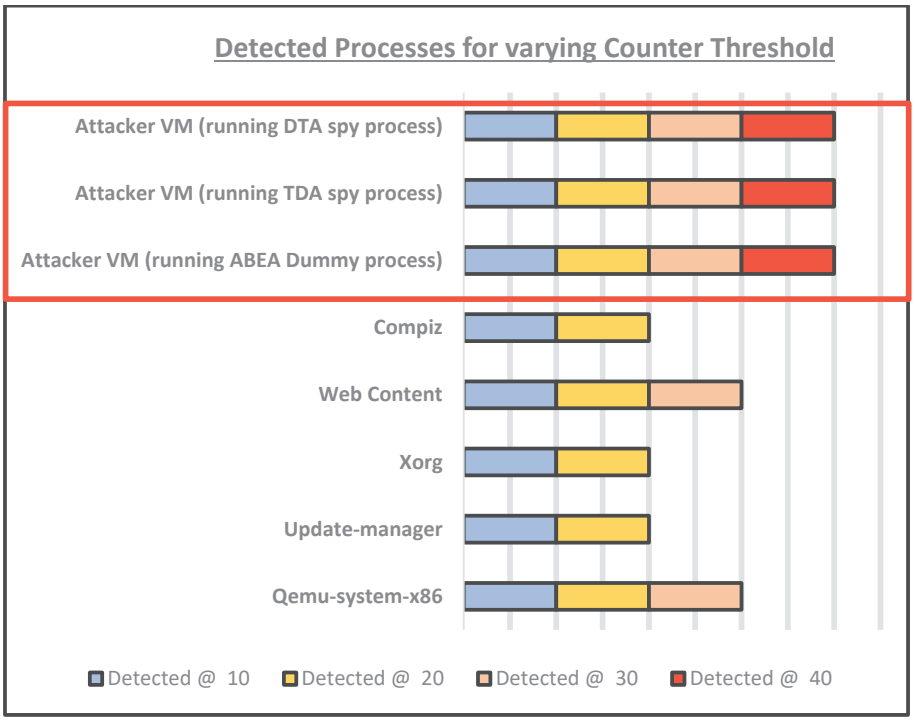


Figure 7.17: Processes caught by IM for varying counter threshold

As discussed previously, IM was set to generate an alert on finding the HPCnt access frequency more than 100. Another option for the generation of alerts can be added from the above discussion. We found that kernel logs for only DTA, TDA, and ABEA spy processes were obtained by setting the threshold value 40. In summary, there are two approaches to generate alert from IM for the means of eliminating the scope of false-positive : (1) On finding the cumulative HPCnt access frequency of a process/VM crossing the limit 100 (2) On finding kernel logs for a process/VM by setting the threshold value 40. In either of the ways, IM will generate an alert the spy process within 1 second time period. Additionally, no other legitimate process will be caught by the IM. The VM for which alert is generated by IM is recommended for status change to *suspicious*.

In turn, the status of VM for which either CAM or BM, as well as IM, generates an alert is changed to *spy*. Obtained results reveal that neither of CAM, BM, or IM generates alert for any legitimate process with the finalized values of the threshold. Even if either of CAM and BM catches a process, there are quite rare chances that the same process is caught by IM also and vice versa. In such circumstances, the probability of false-positive is almost nullified. As

discussed in the previous chapter, an advance trigger is sent to CERT when a VM status becomes *spy*. However, NM is initiated for this *spy* VM for the confirmed detection of the attack.

7.1.5 Network Monitor

The Network Monitor is launched for the VM for which the status is changed to *spy*. The primary action of NM is to trap encrypted confidential messages. Although there may be many ways of trapping network packets, we consider ARPspoofing for our implementation. Accordingly, Network Monitor targets VM2 and searches for double entries of its MAC address. If the Network Monitor finds such an entry, then it changes the status of *spy* VM to *malicious* and generates alert to CERT for further actions.

The result analysis reveals that no legitimate VM crossed the set threshold value for any of the two monitors. Moreover, a VM is declared as *spy* only if both BM and IM generate alert for it. In this scenario, even if an alert is generated for a legitimate VM by either BM or IM, there are extremely low chances where the VM gets caught by both the monitors together.

Further, NM is also set as an additional eye to observe if the attacker generates some packet trapping activity. Although packet trapping is not a part of the TDA attack, the attacker needs to carry it out for a fruitful BPA attack, as discussed in Section 5. In this regard, if the NM generates an alert for the *spy* VM, then the process/VM is certainly *malicious*. Accordingly, we can say that the false-positive rate is extremely low in *Chaturdrashta*.

7.2 Performance Evaluation

There is a limited work done in the area of BPA attack, as discussed in 3.5. The performance comparison of the limited number of existing solutions was carried out in Table 5.2. The result of comparison led to a summary that there is a need to propose a solution to detect the presence of a Cross-VM BPA attack that can overcome the identified limitations of existing solutions. The proposed approach *Chaturdrashta* is the outcome of that summary, where its performance evaluation should be carried out with reference to the present approaches. The table is revised and presented as Table 7.1 that evaluates the performance of *Chaturdrashta*

considering various factors like applicability in virtualization to handle Cross-VM BPA attack, Focused Mechanism, Effect on Legitimate Processes and Dependency on Cryptographic Algorithm. The presented analysis reveals that Chaturdashta can detect the presence of a Cross-VM BPA attack launched with DTA and TDA methods. It is also stated that it is possible for *Chaturdrashta* to detect the other two BTB Eviction attacks in addition to the DTA and TDA. As discussed previously, the new solutions should fulfill two significant criteria: No effect on normal system performance and No dependency on the cryptographic algorithm. As shown in Table 7.1, *Chaturdrashta* fulfills both the criteria.

Table 7.1: Performance Analysis of *Chaturdrashta* against Existing Approaches

Approach	Targets Virtualization?	Applicable to handle Cross-VM BPA Attack?	Focused BPA Attack mechanism	Affects Legitimate Process?	Dependent on the Cryptographic Algorithm?
Agosta et al. [127]	No	Yes	Independent of Attack Mechanism	No	Yes. requires modification in each vulnerable algorithm
Tan et al. [128]	No	Only for VMs with common core	Trace-Driven Attack	Yes	No
Julien et al. [129]	Yes	Yes	Independent of Attack Mechanism	Yes	No
S. Bhattacharya [130]	No	Yes	Direct Timing Attack	Yes	No
Chaturdrashta	**Yes**	**Yes**	**Independent of Attack Mechanism**	**No**	**No**

The above analysis reveals that the proposed *Chaturdrashta* performs as per the defined objectives. At the same time, the overhead of the solution is also a very significant parameter to be considered while evaluating its performance.

Overhead resulted from *Chaturdrashta* is calculated in two ways: (1) Resource Usage of each individual component among four components of *Chaturdrashta*. (2) Overall resource usage of the approach, *Chaturdrashta*

Chaturdrashta is a host-based approach that runs as a background process. Resource usage of *Chaturdrashta* is carried out by measuring resource usage of individual components. As discussed in Subsection 6.2.2, *Chaturdrashta* is comprised of CAM, BM, IM, and NM. Parameters like occupied virtual memory, resident memory, time consumption in hundredths of a second, and CPU utilization are measured for each of the four components using *top* command of Linux Operating System. The obtained results are shown in Figure 7.18. Results plotted in the diagram reveals that CAM occupies the maximum resources that are followed by NM. IM and BM occupy comparatively less amount of memory and CPU. As CAM is the part of *Trilochan,* which is replaced by BM in *Trinetra*, the resource consumption of *Trilochan* is more than that of *Trilochan*.

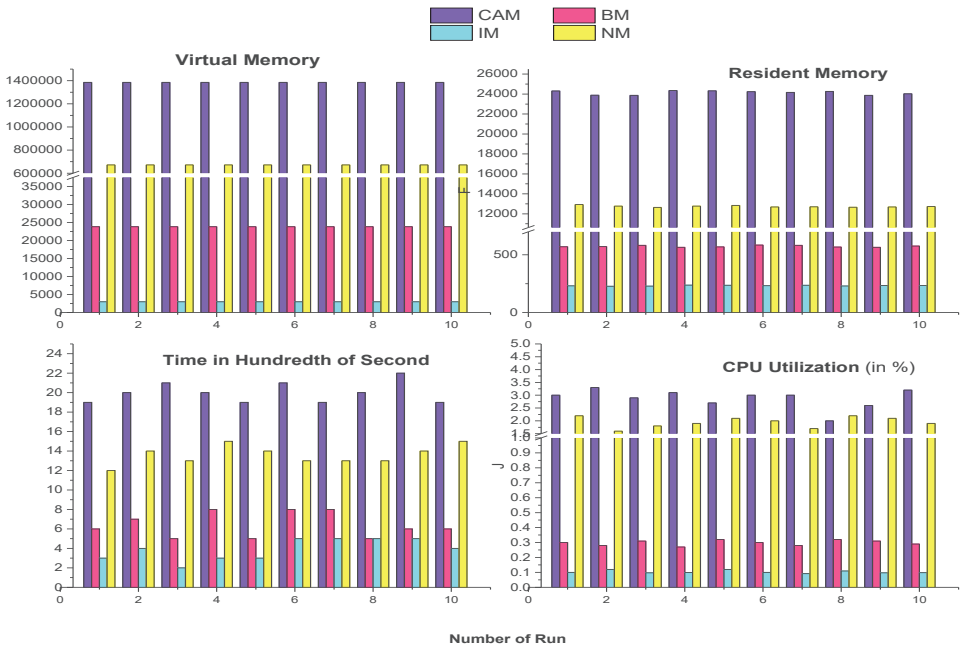


Figure 7.18: Resource Usage of *Chaturdrashta* Components

Performance analysis from another way is carried out for the means of calculating the exact overhead generated by executing *Trilochan* and *Trinetra*. Resource usage was measured using *vmstat* command of the Linux Operating System. The *vmstat* command displays a list of parameter values related to different resources of the overall system. Resource utilization of the proposed solutions can be measured by comparing the results obtained of the overall system in the presence and absence of the running module of the approach. Accordingly, the

performance is measured for three different scenarios : (1) Without running *Trilochan* and *Trinetra* (2) With *Trilochan* (3) With *Trinetra*.

Parameters like memory utilization, CPU utilization, and the total number of interrupt occurrences are measured for all three cases, as reflected in Figure 7.19. It can be started from the observed values of different performance parameters that the overhead of both *Trilochan* and *Trinetra* is within 1%.

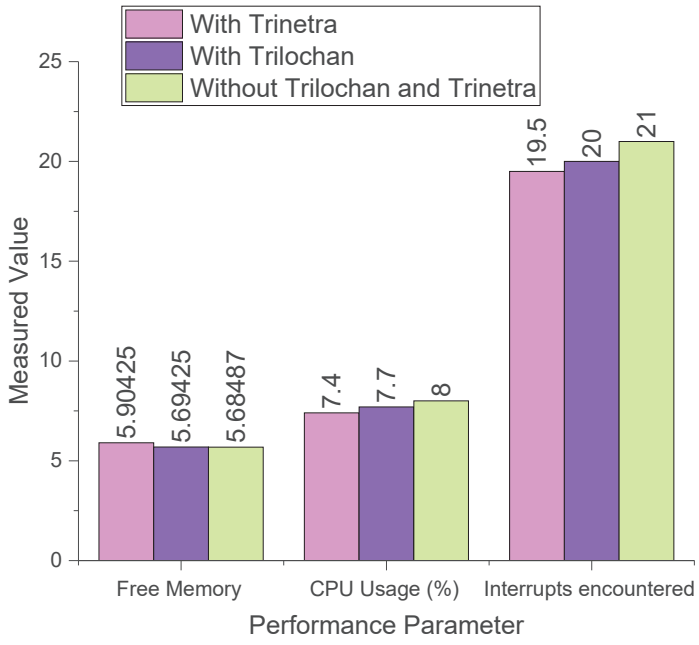


Figure 7.19: Performance Monitoring of *Trilochan* and *Trinetra*

The performance of the proposed solution is also evaluated, considering the time taken in detecting the presence of the BPA attack. Alerts generated by CAM and IM together signals the presence of DTA, where the presence of the TDA is indicated when both BM and IM generate an alert. When ABEA is launched, both CAM and BM generate an alert along with IM, but the alert that is generated first is considered for declaring the VM as *spy*. A plot is shown in Figure 7.20 to represent the timings when the CAM and IM generate an alert.

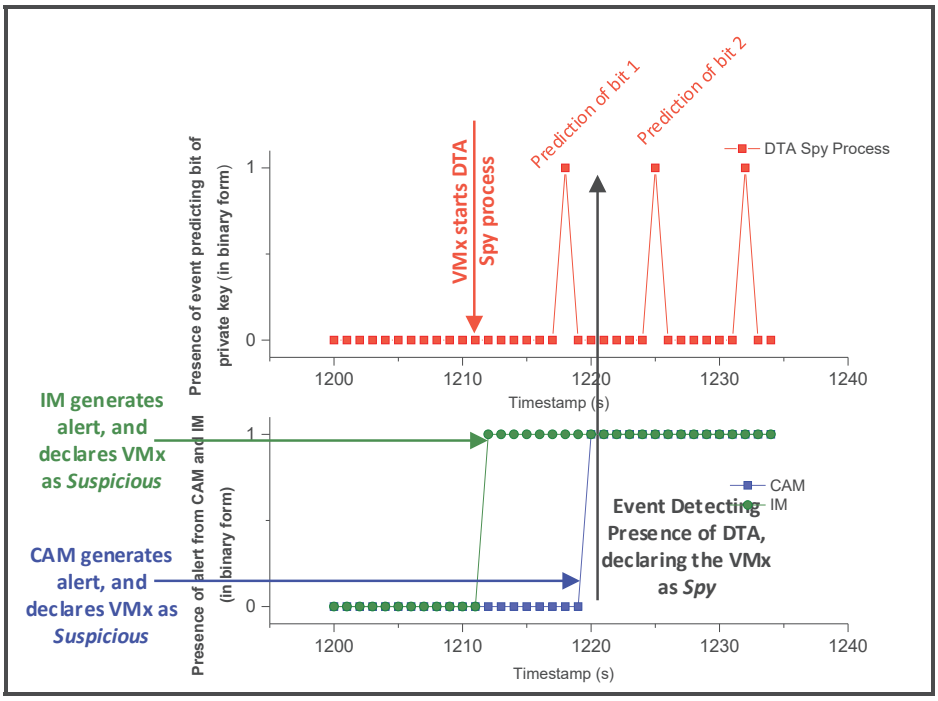


Figure 7.20: Time Taken in the detection of DTA

As shown in Figure 7.20, IM is much faster than CAM. Even with the threshold value 40, IM generates alert at every second. CAM is set to generate alert when the total number of packet traversal crosses the limit of 300 packets per 10 seconds. For the defined threshold values, the DTA is detected during the prediction of the very first bit. Even if we configure CAM and IM to generate alert when they hit the threshold value for multiple times (say 8-10), the DTA would be detected by the time when hardly a few bits are predicted. Similar representations of TDA and ABEA are shown in Figure 7.21 and Figure 7.22, respectively.

It can be observed from Figure 7.21 that, like in the case of DTA, the TDA can also be detected by the time a few bits are predicted. The detection of ABEA differs from DTA and TDA, where alerts are generated from all the three monitors, CAM, BM, and IM. As shown in the diagram, Figure 7.22, alert from IM comes first among the three monitors. Further, as soon as an alert comes either from CAM or BM for the same VM, the VM is declared as *spy*.

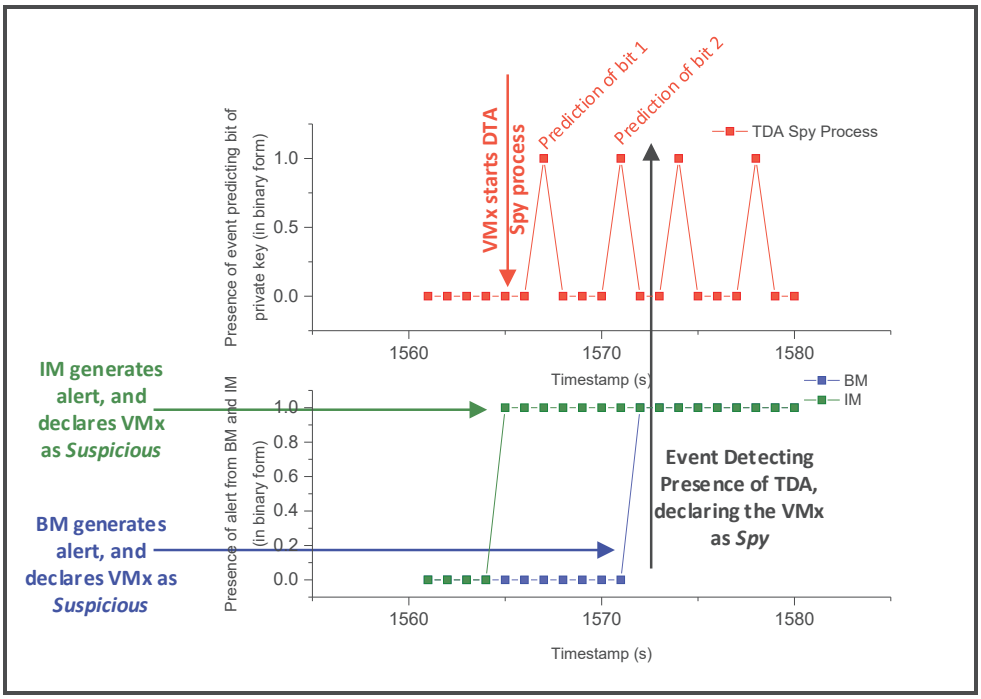


Figure 7.21: Time Taken in the detection of TDA

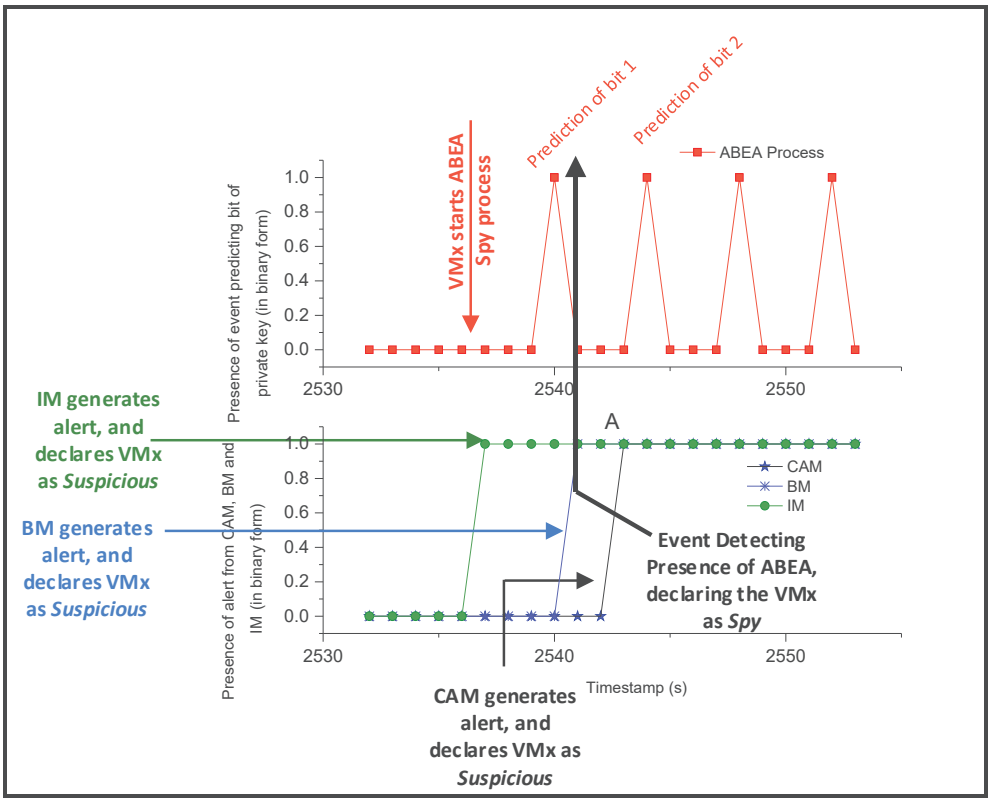


Figure 7.22: Time Taken in the detection of ABEA

The above discussion clearly represents that the presence of DTA, TDA, and ABEA can be detected by the time when a very small part of the key is predicted.

7.3 Summary

Experimental evaluation plays a significant role in proving the effectiveness of a proposed system. Implementation and analysis are carried out to visualize the efficiency of *Chaturdrashta* in this chapter. *Chaturdrashta* is comprised of four monitors, and so, experimental analysis of each individual component is essential to analyze the performance of the entire system. With a simulation setup similar to the one used in the attack simulation, the experiments are carried out for CAM, BM, IM, and NM. CAM, BM, and IM are designed to generate an alert on a particular threshold value, and so the value is required to be chosen carefully to avoid any false positives.

A two-step procedure is implemented to select an appropriate threshold value for CAM, BM, and IM. In the first step, a tentative threshold value is derived based on the results achieved from the respective component in the absence of the BPA attack. The process is repeated in the presence of the BPA attack to derive the final value of the threshold, to eliminate the scope of false positives. It is found that IM responds faster than CAM and BM when the attack is launched.

As discussed in the previous chapter, although CAM/BM and IM are capable of identifying the presence of the BPA attack (DTA/TDA/ABEA), NM is initiated for confirmed attack detection. There are many ways to implement the packet trapping activity, among which ARPSpoofing is chosen for simulation in the present work. NM is meant to catch the packet trapping activity of the *spy* VM for which alert is generated from CAM/BM and IM. The implementation of NM in the present work is carried out to identify multiple MAC addresses of the caught VM, as it considers that the *spy* VM uses ARPSpoofing to trap the packets. As discussed in the previous chapter, a plug-in can be provided for NM so that any other option can replace the present implementation with ARPSpoofing.

The performance of *Chaturdrashta* was compared with the other solutions considering the parameters like applicability in virtualization, the method under consideration, the effect on normal system functioning, and dependency on the cryptographic algorithm. Effectiveness

of *Chaturdrashta* in detecting the presence of DTA, ABEA, and TDA requires an overhead calculation also for assessing the impact on the overall system. Analysis of the resource utilization of all the four monitors reveals that the proposed system is light-weight. Additionally, when the statistical analysis of the entire system was carried out in the absence and presence of *Trilochan* as well as *Trinetra*, it was found that both the approaches take the overhead of only 1%. It was also shown that the detection of DTA, ABEA, and TDA could take place by the time when a few bits are predicted.

CHAPTER

8 Conclusions and Future Scope

The cloud security issue becomes more challenging when the virtualization layer is attacked, which is an abstraction layer protecting underlying resources. The significance of virtualization and related security issues has motivated us to do the research work in that area. A literature survey on the cloud security issues led us to define a taxonomy classifying different generic cloud security issues with the main focus on virtualization. A detailed study on existing work done in each classified type of security issue has revealed a research gap in the area of Branch Prediction Analysis (BPA) attack. BPA attack is a type of Side-Channel Attack (SCA) where resources shared among different processes are exploited to extract confidential information. Among the shared resources like memory bus, cache memory, network queue, etc., the components of Branch Prediction Unit (BPU) like branch predictor and Branch Target Buffer (BTB) are the main target of BPA attack.

The study of BPA attack launching methods revealed a need to assess the scope of the BPA attack in virtualization. A thorough study of the BPA attack revealed the required type the resource sharing configuration for successfully launching BPA on Cross-VM platform by each of the four BPA attack launching methods. Although the analysis of the existing solutions revealed their applicability to handle the Cross-VM BPA attack, a need for a new solution was also identified to overcome their limitations.

The BPA attack with the DTA method, one of the four methods, was simulated by taking different 30 keys where repeated simulation was performed for around 50 times for each key. The TDA, another type of BPA attack, was simulated with different 20 keys where repeated simulation was carried out for around 30-35 times. The simulation of the other two BTB Eviction methods is the future work of our research, considering the complexity in simulating the Synchronous attack. Additionally, the scope of both the attacks is also

required to be assessed in the presence of the Montgomery Ladder algorithm. However, we have done a partial simulation of the Asynchronous BTB Eviction method to observe the actions taken during the event of launching the attack.

Attack simulation highlighted that absolutely legal kind of activities to perform illegal function requires careful observation of primary actions. A behavior analysis of the BPA attack methods was carried out to differentiate a malicious process from the benign ones. As an outcome of the behavior analysis and as the main contribution of our work, *Chaturdrashta*, a four-eyed approach was proposed, comprised of four components designed to observe the primary actions of the attack. With four monitors, and their logical mapping to two sub-approaches, *Trilochan* and *Trinetra*, DTA and TDA were successfully detected on the Cross-VM platform. Additional capabilities of *Trilochand* and *Trinetra* were also represented in detecting the BPA attack even if it is launched with the BTB Eviction methods.

An extensive experimental analysis was carried out to observe the behavior of CAM by analyzing the packet traversal frequency of the DTA and ABEA spy processes with reference to that of the other Linux processes. With a repetitive experiment performed for more than 50 times with ten keys, a threshold value for CAM was identified to reduce the false-positive rate. Similarly, the performance of BM was also analyzed with the Gem5 simulator by measuring the BTB occupancy ratio of various MiBench programs along with the TDA spy process and ABEA dummy process. Results obtained from the experiments carried out for more than 50 times were averaged out as per which the threshold value of BM was identified. It was also shown with the experimental analysis that BTB occupancy of the spy/dummy processes remains high irrespective of the total number of BTB entries. On the other hand, the occupancy of the MiBemch programs reduces as the BTB size increases.

The experimental analysis of IM was done for around 50 times with reference to DTA, TDA, and ABEA to observe the HPCnt access frequency of the spy processes of both the attacks. Results observed by setting different threshold values like 10, 20, 30, and 40 reveal that the spy processes access HPCnt with much more high frequency than all the other Linux processes like Update-Manager, Web-Content, Xorg and Compiz. Moreover, no process other than the spy processes were found to hit the threshold when its value was set to 40.

An intersection was taken between the results of CAM and IM as well as between that of BM and IM. It was observed that both the intersections resulted in empty sets, which ultimately represents almost nullified false-positive rates of *Trilochan* and *Trinetra*. We also observed that the NM was required to be initiated only for the BPA attack launching processes. The above observation gives support to the claim that NM is just an additional monitor designed for the confirmed detection of the attack.

The resource utilization of CAM, BM, and IM was measured to observe the memory, CPU usage, and elapsed time. The performance of both *Trilochan* and *Trinetra* was also evaluated to observe their overhead in terms of memory occupancy, CPU utilization, and interrupt frequency. Obtained results revealed that the overhead of both the approaches is within 1%. Additionally, it is also represented that the detection of the BPA attack can be accomplished by the time when a few bits are extracted. No manipulation of the architectural components of the proposed approach reveals that it does not affect the normal system functioning.

Although *Chatursrashta* is efficient in detecting the presence of the BPA attack irrespective of the underlying attack methods, more exploration is needed for the Asynchronous and Synchronous BTB Eviction methods. The complex simulation of the Syncrrhonous attack and the requirement of the scope assessment of both the methods in the presence of the Ladder algorithm are very crucial tasks, which are identified as the future scope.

Chaturdrashta can work efficiently for the Cross-VM BPA attack. With a minor change in the design of CAM, its applicability can get extended for an environment where both compromised and victim processes reside on the same system. *Chaturdrashta* is already independent of the victim cryptographic algorithm and the BPA attack launching method. With the modified CAM, it can also become platform-independent also.

List of References

[1] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[2] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O’Reilly Media, Inc., 2009.

[3] B. Halpert, *Auditing Cloud Computing: A Security and Privacy Guide*. Wiley, 2011.

[4] D. Shackleford, *Virtualization Security: Protecting Virtualized Environments*. Wiley, 2012.

[5] Z. Nan, “Virtualization safety problem analysis,” in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011, pp. 195–197.

[6] J. Brodkin, “Gartner: Seven cloud-computing security risks,” *Infoworld*, vol. 2008, pp. 1–3, 2008.

[7] M. A. Khan, “A survey of security issues for cloud computing,” *J. Netw. Comput. Appl.*, vol. 71, pp. 11–29, Aug. 2016.

[8] N. Gonzalez *et al.*, “A quantitative analysis of current security concerns and solutions for cloud computing,” *J. Cloud Comput. Adv. Syst. Appl.*, vol. 1, no. 1, p. 11, 2012.

[9] N. Gruschka and M. Jensen, “Attack Surfaces: A Taxonomy for Attacks on Cloud Services,” in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 276–279.

[10] M. K. Srinivasan, K. Sarukesi, P. Rodrigues, M. S. Manoj, and P. Revathy, “State-of-the-Art Cloud Computing Security Taxonomies: A Classification of Security Challenges in the Present Cloud Computing Environment,” in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, 2012, pp. 470–476.

[11] L. Coppolino, S. D’Antonio, G. Mazzeo, and L. Romano, “Cloud security: Emerging threats and current solutions,” *Comput. Electr. Eng.*, vol. 59, pp. 126–140, Apr. 2017.

[12] S. Singh, Y.-S. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” *J. Netw. Comput. Appl.*, vol. 75, pp. 200–222, Nov. 2016.

[13] S. M. Hashemi and M. R. M. Ardakani, “Article: Taxonomy of the Security Aspects of Cloud Computing Systems - A Survey,” *Int. J. Appl. Inf. Syst.*, vol. 4, no. 1, pp. 21–28, Sep. 2012.

[14] G. Pék, L. Buttyán, and B. Bencsáth, “A Survey of Security Issues in Hardware Virtualization,” *ACM Comput. Surv.*, vol. 45, no. 3, Jul. 2013.

[15] C. N. Modi and K. Acha, “Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review,” *J. Supercomput.*, vol. 73, no. 3, pp. 1192–1234, 2017.

[16] D. Sgandurra and E. Lupu, “Evolution of Attacks, Threat Models, and Solutions for Virtualized Systems,” *ACM Comput. Surv.*, vol. 48, no. 3, Feb. 2016.

[17] A. R. Riddle and S. M. Chung, “A Survey on the Security of Hypervisors in Cloud Computing,” in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, 2015, pp. 100–104.

[18] M. Ali, S. U. Khan, and A. V. Vasilakos, “Security in cloud computing: Opportunities and challenges,” *Inf. Sci. (Ny).*, vol. 305, pp. 357–383, Jun. 2015.

[19] M. Almorsy, J. Grundy, and I. Müller, “An analysis of the cloud computing security problem,” *arXiv Prepr. arXiv1609.01107*, 2016.

[20] S. Iqbal *et al.*, “On cloud security attacks: A taxonomy and intrusion detection and prevention as a service,” *J. Netw. Comput. Appl.*, vol. 74, pp. 98–120, Oct. 2016.

[21] W. Dawoud, I. Takouna, and C. Meinel, “Infrastructure as a service security: Challenges and solutions,” in *2010 The 7th International Conference on Informatics and Systems (INFOS)*, 2010, pp. 1–8.

[22] T. Takahashi, G. Blanc, Y. Kadobayashi, D. Fall, H. Hazeyama, and S. Matsuo, “Enabling secure multitenancy in cloud computing: Challenges and approaches,” in *2012 2nd Baltic Congress on Future Internet Communications*, 2012, pp. 72–79.

[23] G. Zhao, C. Rong, M. G. Jaatun, and F. E. Sandnes, “Deployment models: Towards eliminating security concerns from cloud computing,” in *2010 International Conference on High Performance Computing Simulation*, 2010, pp. 189–195.

[24] Aishwarya C.S and Revathy S, “Insight into Cloud Security issues,” *Int. J. Adv. Comput. Sci. Its Appl.*, vol. 1, no. 1, pp. 30–33, 2011.

[25] M. Sudha, D. B. R. K. Rao, and M. Monica, “A comprehensive approach to ensure secure data communication in cloud environment,” *Int. J. Comput. Appl.*, vol. 12, no. 8, pp. 19–23.

[26] Gobi M. and Sridevi R., “An Approach for Secure Data Storage in Cloud Environment,” *Int. J. Comput. Commun. Eng.*, vol. 2, no. 2, pp. 206–209, 2013.

[27] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in Cloud Computing,” in *2009 17th International Workshop on Quality of Service*, 2009, pp. 1–9.

[28] D. Purushothaman and S. Abburu, “An Approach for Data Storage Security in Cloud Computing,” *Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 100–105, 2012.

[29] S. REDDY and M. BALARAJU, “An Integrated Approach of Data storage and Security in Cloud Computing,” *Int. J. Appl. or Innov. Eng. Manag.*, vol. 1, no. 4, pp. 72–78, 2012.

[30] A. Sangroya, S. Kumar, J. Dhok, and V. Varma, “Towards Analyzing Data Security Risks in Cloud Computing Environments,” in *Information Systems, Technology and Management*, 2010, pp. 255–265.

[31] R. G. Reddy, M. Raju, and R. Naik, “A Novel Approach for Multi-Cloud Storage

security in Cloud Computing," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 5, pp. 5043–5045, 2012.

[32] A. Juels and A. Oprea, "New Approaches to Security and Availability for Cloud Data," *Commun. ACM*, vol. 56, no. 2, pp. 64–73, Feb. 2013.

[33] D. Smith, Q. Guan, and S. Fu, "An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems," in *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*, 2010, pp. 376–381.

[34] K. Hamlen, M. Kantarcioglu, L. Khan, and B. Thuraisingham, "Security Issues for Cloud Computing," *Int. J. Inf. Secur. Priv.*, vol. 4, no. 2, pp. 39–51, 2010.

[35] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Collaboration-Based Cloud Computing Security Management Framework," in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 364–371.

[36] Q. Su, F. Wang, and Q. Hang, "Study of Cloud Computing Security Service Model," in *2012 Spring Congress on Engineering and Technology*, 2012, pp. 1–4.

[37] U. Tupakula, V. Varadharajan, and N. Akku, "Intrusion Detection Techniques for Infrastructure as a Service Cloud," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 744–751.

[38] L. M. Vaquero, L. Rodero-Merino, and D. Morán, "Locking the sky: a survey on IaaS cloud security," *Computing*, vol. 91, no. 1, pp. 93–118, 2011.

[39] M. Yildiz, J. Abawajy, T. Ercan, and A. Bernoth, "A Layered Security Approach for Cloud Computing Infrastructure," in *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009, pp. 763–767.

[40] P. Arora, R. C. Wadhawan, and S. P. Ahuja, "Cloud Computing Security Issues in Infrastructure as a Service," *Int. J. Adv. Res. inComputer Sci. Softw. Eng.*, vol. 2, no. 1, 2012.

[41] T. Alharkan and P. Martin, "IDSaaS: Intrusion Detection System as a Service in Public Clouds," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, 2012, pp. 686–687.

[42] G. Nascimento and M. Correia, "Anomaly-based intrusion detection in software as a service," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2011, pp. 19–24.

[43] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, Jan. 2011.

[44] 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*, 2009, pp. 109–116.

[45] D. Kusnetzky, *Virtualization: A Manager's Guide*. O'Reilly Media, Inc., 2011.

[46] T. Garfinkel and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments," in *Proceedings of*

the 10th Conference on Hot Topics in Operating Systems - Volume 10, 2005, p. 20.

[47] J. Kirch, "Virtual Machine Security Guidelines Version 1.0," 2006.

[48] J. S. Reuben, "A survey on virtual machine Security," 2007.

[49] R. Anand, S. Sarswathi, and R. Regan, "Security issues in virtualization environment," in *2012 International Conference on Radar, Communication and Computing (ICRCC)*, 2012, pp. 254–256.

[50] T. T. Brooks, C. Caicedo, and J. S. Park, "Security Vulnerability Analysis in Virtualized Computing Environments," *Int. J. Intell. Comput. Res.*, vol. 3, no. 4, pp. 263–277, 2012.

[51] J. Li *et al.*, "CyberGuarder: A virtualization security assurance architecture for green cloud computing," *Futur. Gener. Comput. Syst.*, vol. 28, no. 2, pp. 379–390, Feb. 2012.

[52] K. Kortchinsky, "CLOUDBURST," *BlackHat USA*, 2009. .

[53] N. Elhage, "Virtunoid: A KVM Guest→ Host privilege escalation exploit," *Black Hat USA*, vol. 2011, 2011.

[54] A. and G. A. Wang Jiangand Stavrou, "HyperCheck: A Hardware-Assisted Integrity Monitor," in *Recent Advances in Intrusion Detection*, 2010, pp. 158–177.

[55] Z. Wang and X. Jiang, "HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 380–395.

[56] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the Hypervisor Attack Surface for a More Secure Cloud," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011, pp. 401–412.

[57] N. L. Petroni and M. Hicks, "Automated Detection of Persistent Kernel Control-Flow Attacks," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 103–115.

[58] X. and X. D. Riley Ryanand Jiang, "Guest-Transparent Prevention of Kernel Rootkits with VMM-Based Memory Shadowing," in *Recent Advances in Intrusion Detection*, 2008, pp. 1–20.

[59] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes," in *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, vol. 41, no. 6, pp. 335–350.

[60] L. Litty, H. A. Lagar-Cavilla, and D. Lie, "Hypervisor Support for Identifying Covertly Executing Binaries," in *Proceedings of the 17th Conference on Security Symposium*, 2008, pp. 243–258.

[61] A. Srivastava, K. Singh, and J. Giffin, "Secure Observation of Kernel Behavior," 2008.

[62] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “VMM-Based Hidden Process Detection and Identification Using Lycosid,” in *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2008, pp. 91–100.

[63] S. and L. P. and J. J. Zhang Lingchenand Shetty, “RootkitDet: Practical End-to-End Defense against Kernel Rootkits in a Cloud Environment,” in *Computer Security - ESORICS 2014*, 2014, pp. 475–493.

[64] R. Hund, T. Holz, and F. C. Freiling, “Return-Oriented Rootkits: Bypassing Kernel Code Integrity Protection Mechanisms,” in *Proceedings of the 18th Conference on USENIX Security Symposium*, 2009, pp. 383–398.

[65] Z. Wang, X. Jiang, W. Cui, and P. Ning, “Countering Kernel Rootkits with Lightweight Hook Protection,” in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009, pp. 545–554.

[66] S. Bharadwaja, W. Sun, M. Niamat, and F. Shen, “Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System,” in *2011 Eighth International Conference on Information Technology: New Generations*, 2011, pp. 695–700.

[67] Y. Du, R. Zhang, and M. Li, “Research on a security mechanism for cloud computing based on virtualization,” *Telecommun. Syst.*, vol. 53, no. 1, pp. 19–24, 2013.

[68] S. T. King, P. M. Chen, Y.-M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch, “SubVirt: Implementing malware with virtual machines,” in *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24May 2006, Berkeley, California, USA*, 2006, pp. 314–327.

[69] J. Rutkowska, “Subverting Vista Kernel for Fun and Profit.” [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2008/08/20084218/BH-US-06-Rutkowska.pdf.

[70] E. Barbosa, “Detecting of Hardware Virtualization Rootkits,” in *Symposium on Security for Asia Network (SyScan), Singapore*, 2007.

[71] T. Ristenpart, E. Tromer, H. Shacham, and S. 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*, 2009, pp. 199–212.

[72] F. Hao, T. V Lakshman, S. Mukherjee, and H. Song, “Secure Cloud Computing with a Virtualized Network Infrastructure,” in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010, p. 16.

[73] F. Zhao, W. Yang, H. Jin, and S. Wu, “VNIDS: A virtual machine-based network intrusion detection system,” in *2008 2nd International Conference on Anti-counterfeiting, Security and Identification*, 2008, pp. 254–259.

[74] K. Kourai and S. Chiba, “HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection,” in *Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments*, 2005, pp. 197–207.

[75] A. Srivastava and J. Giffin, “Tamper-Resistant, Application-Aware Blocking of Malicious Network Connections,” in *Recent Advances in Intrusion Detection*, 2008, pp. 39–58.

[76] U. Tupakula and V. Varadharajan, “On the design of Virtual machine Intrusion detection system,” in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, 2011, pp. 682–685.

[77] Y. Zhang, Y. Gu, H. Wang, and D. Wang, “Virtual-Machine-based Intrusion Detection on File-aware Block Level Storage,” in *2006 18th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD’06)*, 2006, pp. 185–192.

[78] H. Jin *et al.*, “A VMM-based intrusion prevention system in cloud computing environment,” *J. Supercomput.*, vol. 66, no. 3, pp. 1133–1151, 2013.

[79] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen, “Detecting Past and Present Intrusions through Vulnerability-Specific Predicates,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, pp. 91–104, Oct. 2005.

[80] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, “ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 211–224, Dec. 2003.

[81] T. Garfinkel and M. Rosenblum, “A virtual machine introspection based architecture for intrusion detection,” in *Net. and Distributed Sys. Sec. Symp.*, 2003.

[82] M. Noura, S. Mohammadalian, L. Fathi, and M. Torabi, “Secure Virtualization for Cloud Environment Using Guest OS and VMM-based Technology,” *Int. J. Comput. Networks Commun. Secur.*, vol. 1, no. 2, pp. 61–67, 2013.

[83] B. D. Payne, M. Carbone, M. Sharif, and W. Lee, “Lares: An Architecture for Secure Active Monitoring Using Virtualization,” in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008, pp. 233–247.

[84] X. Jiang, X. Wang, and D. Xu, “Stealthy Malware Detection and Monitoring through VMM-Based ‘out-of-the-Box’ Semantic View Reconstruction,” *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, Mar. 2010.

[85] D. Patidar, P. S. Patheja, and A. A. Waoo, “An efficient approach for cloud computing based on hierarchical secure paravirtualization system resource model,” *Int. J. Appl. Eng. Res.*, vol. 7, no. 11, pp. 1527–1534, 2012.

[86] R. Kadam and M. Bangare, “A Survey on Security Issues and Solutions in Live Virtual Machine Migration,” *Int. J. Adv. Found. Res. Comput.* , vol. 1, no. 12, pp. 131–137, 2014.

[87] M. Aiash, G. Mapp, and O. Gemikonakli, “Secure Live Virtual Machines Migration: Issues and Solutions,” in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, 2014, pp. 160–165.

[88] W. Wang, Y. Zhang, B. Lin, X. Wu, and K. Miao, “Secured and reliable VM migration in personal cloud,” in *2010 2nd International Conference on Computer Engineering and Technology*, 2010, vol. 1, pp. V1-705-V1-709.

[89] K. Sammy, R. Shengbing, and C. Wilson, “Energy efficient security preserving vm live migration in data centers for cloud computing,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 1694–1814, 2012.

[90] G. J. Jeincy, R. S. Shaji, and J. P. Jayan, “A secure virtual machine migration using memory space prediction for cloud computing,” in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2016, pp. 1–5.

[91] P. J. Reeba, R. S. Shaji, and J. P. Jayan, “A secure virtual machine migration using processor workload prediction method for cloud environment,” in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2016, pp. 1–6.

[92] J. Oberheide, E. Cooke, and F. Jahanian, “Empirical exploitation of Live virtual machine migration,” in *Black Hat Security Conference*, 2008.

[93] A. Burtsev, K. Srinivasan, P. Radhakrishnan, L. N. Bairavasundaram, K. Voruganti, and G. R. Goodson, “Fido: Fast Inter-Virtual-Machine Communication for Enterprise Appliances,” in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, 2009, p. 25.

[94] W. Huang, M. J. Koop, Q. Gao, and D. K. Panda, “Virtual Machine Aware Communication Libraries for High Performance Computing,” in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, 2007.

[95] K. Kim, C. Kim, S.-I. Jung, H.-S. Shin, and J.-S. Kim, “Inter-Domain Socket Communications Supporting High Performance and Full Binary Compatibility on Xen,” in *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2008, pp. 11–20.

[96] D. Li, H. Jin, Y. Shao, and X. Liao, “A High-Efficient Inter-Domain Data Transferring System for Virtual Machines,” in *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, 2009, pp. 385–390.

[97] J. Wang, K.-L. Wright, and K. Gopalan, “XenLoop: a transparent high performance inter-VM network loopback,” *Cluster Comput.*, vol. 12, no. 2, pp. 141–152, 2009.

[98] L. Youseff, D. Zagorodnov, and R. Wolski, “Inter-OS Communication on Highly Parallel Multi-Core Architectures,” Tech. rep. University of California, Santa Barbara, 2008.

[99] S. and R. P. and G. J. L. Zhang Xiaolanand McIntosh, “XenSocket: A High-Throughput Interdomain Transport for Virtual Machines,” in *Middleware 2007*, 2007, pp. 184–203.

[100] C. Gebhardt and A. Tomlinson, “Challenges for inter virtual machine communication,” *R. Holloway, Univ. London, Tech. Rep.*, 2010.

[101] J. Wang, “Survey of State-of-the-art in Inter-VM Communication Mechanisms,” *Res. Profic. Rep.*, 2009.

[102] S. Yang, W. Chen, and Y. Wang, “ICAS: An inter-VM IDS Log Cloud Analysis System,” in *2011 IEEE International Conference on Cloud Computing and*

Intelligence Systems, 2011, pp. 285–289.

[103] S. Khoudali, K. Benzidane, and A. Sekkaki, "Inter-VM packet inspection in Cloud Computing," in *The 5th International Conference on Communications, Computers and Applications (MIC-CCA2012)*, 2012, pp. 84–89.

[104] K. Benzidane, S. Khoudali, F. Leila, and A. Sekkaki, "Toward inter-VM visibility in a Cloud environment using packet inspection," in *ICT 2013*, 2013, pp. 1–5.

[105] K. Benzidane, S. Khoudali, and A. Sekkaki, "Secured architecture for inter-VM traffic in a Cloud environment," in *2nd IEEE Latin American Conference on Cloud Computing and Communications*, 2013, pp. 23–28.

[106] Q. Zhang *et al.*, "Residency Aware Inter-VM Communication in Virtualized Cloud: Performance Measurement and Analysis," in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 204–211.

[107] C. Luo, Y. Bai, T. Chen, C. Xu, and L. Zhang, "A Functional Classification Based Inter-VM Communication Mechanism with Multi-core Platform," in *2009 15th International Conference on Parallel and Distributed Systems*, 2009, pp. 332–339.

[108] Y. Fan, J. Xin, W. Lina, and Y. Rongwei, "Secure data sharing between domains in Xen," in *International Conference on Cyberspace Technology (CCT 2014)*, 2014, pp. 1–4.

[109] J. Betz, D. Westhoff, and G. Müller, "Survey on covert channels in virtual machines and cloud computing," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 6, p. e3134, Jun. 2017.

[110] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM Side Channels and Their Use to Extract Private Keys," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012, pp. 305–316.

[111] B. Sevak, "Security against Side Channel Attack in Cloud Computing," *Int. J. Eng. Adv. Technol.* , vol. 2, no. 2, pp. 183–186, 2012.

[112] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen, and R. Schlichting, "An Exploration of L2 Cache Covert Channels in Virtualized Environments," in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, 2011, pp. 29–40.

[113] C. Percival, "Cache missing for fun and profit," 2005.

[114] D. J. Bernstein, "Cache-timing attacks on AES," 2005.

[115] D. Page, "Partitioned Cache Architecture as a Side-Channel Defence Mechanism," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 280, 2005.

[116] 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)*, 2013, pp. 186–191.

[117] B. K. Mughal, A. Syed, and A. Khan, "Information leakage in the cloud," in *2014 International Conference on Open Source Systems & Technologies*, 2014, pp. 56–61.

[118] A. and T. E. Osvik Dag Arneand Shamir, “Cache Attacks and Countermeasures: The Case of AES,” in *Topics in Cryptology – CT-RSA 2006*, 2006, pp. 1–20.

[119] Y. Kulah, B. Dincer, C. Yilmaz, and E. Savas, “SpyDetector: An approach for detecting side-channel attacks at runtime,” *Int. J. Inf. Secur.*, vol. 18, no. 4, pp. 393–422, 2019.

[120] O. Acundefinediçmez, Ç. K. Koç, and J.-P. Seifert, “Predicting Secret Keys via Branch Prediction,” in *Proceedings of the 7th Cryptographers’ Track at the RSA Conference on Topics in Cryptology*, 2007, pp. 225–242.

[121] O. Aciiçmez, Ç. K. Koç, and J.-P. Seifert, “On the Power of Simple Branch Prediction Analysis,” in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, 2007, pp. 312–320.

[122] S.-M. Joye Marcand Yen, “The Montgomery Powering Ladder,” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, 2003, pp. 291–302.

[123] S. and S. J.-P. Aciçmez Onurand Gueron, “New Branch Prediction Vulnerabilities in OpenSSL and Necessary Software Countermeasures,” in *Cryptography and Coding*, 2007, pp. 185–203.

[124] S. Bhattacharya and D. Mukhopadhyay, “Fault Attack revealing Secret Keys of Exponentiation Algorithms fromBranch Prediction Misses,” *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 790, 2014.

[125] S. Bhattacharya and D. Mukhopadhyay, “Who watches the watchmen? : Utilizing Performance Monitors for Compromisingkeys of RSA on Intel Platforms,” *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 621, 2015.

[126] S. Bhattacharya and D. Mukhopadhyay, “Formal fault analysis of branch predictors: attacking countermeasures of asymmetric key ciphers,” *J. Cryptogr. Eng.*, vol. 7, no. 4, pp. 299–310, 2017.

[127] G. Agosta, L. Breveglieri, G. Pelosi, and I. Koren, “Countermeasures Against Branch Target Buffer Attacks,” in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*, 2007, pp. 75–79.

[128] Y. Tan, J. Wei, and W. Guo, “The Micro-architectural Support Countermeasures against the Branch Prediction Analysis Attack,” in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 2014, pp. 276–283.

[129] J. Sebot and S. Gueron, “Mitigating branch prediction and other timing based side channel attacks,” US8869294B2, 2014.

[130] S. Bhattacharya, S. Bhasin, and D. Mukhopadhyay, “Online Detection and Reactive Countermeasure for Leakage from BPU Using TVLA,” in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, 2018, pp. 155–160.

[131] S. Kumar and B. Indrani, “A Study on Web Hijacking Techniques and Browser Attacks,” *Int. J. Appl. Eng. Res.*, vol. 13, no. 5, pp. 2614–2618, 2018.

[132] A. C. de Melo, “The New Linux ’perf’ tools.” In slides from Linux Kongress, 2010.

[133] N. Binkert *et al.*, “The Gem5 Simulator,” *SIGARCH Comput. Arch. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[134] A. R. Chordiya, S. Majumder, and A. Y. Javaid, “Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools,” in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 438–443.

[135] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, 2001, pp. 3–14.

List of Publications

1 Published

a. D. H. Buch and H. S. Bhatt, “Taxonomy on Cloud Computing Security Issues at Virtualization Layer,” International Journal of Advanced Research in Engineering and Technology. (IJARET), vol. 9, no. 4, pp. 50–76, 2018.

b. D. H. Buch and H. S. Bhatt, “Cross-VM Branch Prediction Analysis Attack : Scope Assessment and Simulation,” International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 2, pp. 4868–4873, 2019.

c. D. H. Buch and H. S. Bhatt. "*Trinetra*: a solution to handle cross-VM time-driven attack." SN Applied Sciences, vol. 2, no. 524, pp. 1-12, 2020.

2 Under Review

D. H. Buch and H. S. Bhatt, “*Trilochan*: A Solution to detect the presence of Cross-VM Direct Timing Attack” in Journal of Software: Practice and Experience