# Hypervisor Introspection Detection

*A dissertation report submitted in fulfilment of the requirements for the degree of Bachelor of Technology*

*by*

Shreya Jain(2016UCP1360)

Bhawana Sankhla(2016UCP1457) Birat Adhikari(2016UCP1033)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

APRIL 2020

# Certificate

We,

### Shreya Jain (2016ucp1360)

**Bhawana Sankhla (2016ucp1457) Birat Adhikari (2016ucp1033)**

Declare that this thesis titled, “Hypervisor Introspection Detection” and the work presented in it are our own. I confirm that:

* This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology

Jaipur (MNIT).

* Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.
* We have acknowledged all main sources of help.

Signed:

Date:

Dr. Pilli Emmanuel Shubhakar

H.O.D. (Associate Professor)

April 2020 Department of Computer Science & Engineering Malaviya National Institute of Technology Jaipur

# Abstract

The project aims at hypervisor level intrusion detection which helps in detecting the attack processes in order to secure the system. These processes facilitate the privilege escalation or corrupt its VMs, bypassing data protections and giving the adversary control over-processing. All these malicious processes can be stopped by checking their low-level information and required steps can be taken to abort such processes.

This is achieved by launching attacks on the hypervisor and capturing system call signature of the respective process. In this manner, the system calls of both malicious and normal processes are recorded. The system is further trained on a similar dataset which uses system calls for training and testing purposes. The dataset is preprocessed and trained on different ML Algorithms to check which model is highly beneficial. And then it will be further used to predict the attacks on the hypervisor lay

# Acknowledgement

We would like to express our deep gratitude to Dr Pilli Emmanuel Shubhakar, our project supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this work. A constant push and constructive feedback made this possible and helped us to have hands-on experience with the new technologies.

Shreya Jain Bhawana Sankhla Birat Adhikari

(2016UCP1360) (2016UCP1457) (2016UCP1033)

iii

# Contents

|  |  |
| --- | --- |
| **Certificate** | **i** |
| **Abstract** | **ii** |
| **Acknowledgements** | **iii** |
| **List of Figures** | **vi** |
| **List of Tables** | **vii** |
| **Introduction** | **1** |
| **Important Terms and Concepts** | **2** |
| 2.1 Hypervisor | 2 |
| 2.1.1 Hypervisor Types | 2 |
| 2.1.2 Xen | 3 |
| 2.1.3 KVM | 3 |
| 2.1.4 Virtual Machine | 3 |
| 2.2 Xen Vulnerability | 4 |
| 2.2.1 Vulnerability 1 | 4 |
| 2.2.2 Vulnerability 2 | 4 |
| 2.3 Torshammer DDOS | 4 |
| 2.4 Hping3 | 5 |
| 2.5 Trace | 5 |
| 2.5.1 Strace | 5 |
| 2.6 ADFA-LD Dataset | 5 |
| 2.7 Machine Learning Models | 6 |
| 2.7.1 Random Forest Random forests | 6 |
| 2.7.2 Logistic Model | 6 |
| 2.7.3 Naive-Bayes Classifier | 6 |
| 2.7.4 KNN Classifier | 6 |
| 2.7.5 Decision Tree | 7 |
| 2.7.6 Support Vector Machines | 7 |
| **Related Work** | **8** |
| **Proposed Method** | **9** |
| 4.1 LAUNCHING ATTACKS | 9 |
| 4.1.1 Xen-Vulnerability | 9 |
| 4.1.2 TCP-IP Hping3 Attack | 9 |
| 4.1.3. Torshammer DDOS | 9 |

* + 1. [Buffer Overflow 10](#_TOC_250010)
    2. [KeyLogger 10](#_TOC_250009)
  1. [TRACE 10](#_TOC_250008)
  2. [DETECTION 10](#_TOC_250007)

[Experimental Results 12](#_TOC_250006)

* 1. [TCP/IP 12](#_TOC_250005)
  2. [Torshammer DDOS 13](#_TOC_250004)
  3. [Rootkit Attack 13](#_TOC_250003)
  4. [ADFA-LD 13](#_TOC_250002)

[Conclusion 15](#_TOC_250001)

[References 16](#_TOC_250000)

**List of Figures**

1. Hypervisor. 2
2. Methods used for tracing 10
3. Hping3 attack launched. 11
4. Hping3 attack results. 11
5. Torshammer attack 12
6. Rootkit attack 12

# List of Tables

Table 1. Comparison of classification models 22

# Chapter 1

**Introduction**

Virtualization is a key technology that enables ubiquitous access to shared pools of system resources and high-level services provisioned with minimal management effort. An Operating System (OS) directly controls hardware resources in a non-virtualized system, but virtualization, typically performed by a hypervisor (also called a virtual machine monitor or VMM) provides a mechanism that abstracts the hardware and system resources from an OS. As a software layer that lies between the physical hardware and the Virtual Machines (VMs or guest machines), a hypervisor supports the guest machines by presenting the guest OSs with a virtual operating platform and managing their execution.[1]

With the increasing importance of hypervisors in today's era, the security of hypervisors is a major concern. The project aims at predicting the attempt of the various type of breach in the security at the hypervisor layer that includes the compromised hypervisor or leak of sensitive information of hypervisor. And a compromised hypervisor is able to introspect and corrupt its VMs, bypassing data protections and giving the adversary control over processing etc. Hypervisor Introspection can be done using guest VM or from other sources that lead to running malicious processes. HVI is at a higher privilege level which provides it to access the low-level information of those processes. So in order to detect it, check the low-level information of those processes running on the hypervisor as similar processes can have the similar property or attributes to launch an attack. Here, the attribute is the signature of the system calls . Each signature represents a sequence of the system calls with the ord

# Chapter 2 Important Terms and Concepts

### Hypervisor

A hypervisor is also known as a Virtual Machine Manager (VMM) and its sole purpose is to allow multiple “machines” to share a single hardware platform. Operating systems are designed so that they have a one-to-one relationship with the hardware they are running on, but with multi-core, multi-threaded processors and ludicrous amounts of RAM, running multiple at once is a breeze.

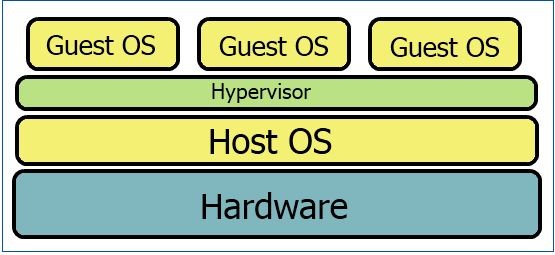


Figure-1 Hypervisor

The hypervisor separates the operating system (OS) from the hardware by taking the responsibility of allowing each running OS time with the underlying hardware. It acts as a traffic cop to allow time to use the CPU, memory, GPU, and other hardware. Each operating system controlled by the hypervisor is called a guest OS, and the hypervisor’s operating system, if any, is called the host OS. Because it stands between the guest OS and hardware you can have as many different guest OSs as your system can handle; you can even have different types (e.g. Windows, OS X, Linux).

### Hypervisor Types

**Type 1**, Bare Metal, is a hypervisor that installs directly onto a computer. There is no host OS and the hypervisor has direct access to all hardware and features. Bare metal is most often used for servers because of their security and portability to move from hardware to hardware in case of a crash.

**Type 2**, Hosted, is what most people are probably familiar with when it comes to virtualizing operating systems. Hosted hypervisors require a host OS and are often treated as installed software inside the host. Type 2 hypervisors are the ideal way to go when you need to test multiple OS. [2]

* + 1. **Xen**

The Xen hypervisor provides two types of domains – a single control domain (also called Domain0 or Dom0) and multiple guest domains (also called DomainU or DomU). Since the hypervisor supports two different virtualization modes, Paravirtualization (PV) and Hardware-assisted Virtualization (HVM), a total of three different types of VMs – Domain0 VM (Dom0-VM), DomainU--PVM (DomU-PVM) and DomainU-- HVM (DomU-HVM) can be hosted on the Xen platform. Dom0 is the initial domain started by the Xen hypervisor on booting up a privileged domain that plays the administrator role and supplies services for DomU VMs. For the two kinds of DomU guests, PV is a highly efficient and lightweight virtualization technology introduced by Xen in which Xen PV does not require virtualization extensions from the host hardware. Thus, PV enables virtualization on hardware architectures that do not support HVM, but it requires PV-enabled kernels and PV drivers to power a high-performance virtual server.[1]

## KVM

In the open-source hypervisor projects, the Kernel-based Virtual Machine (KVM) is a relatively new product which was first introduced in 2006 and soon merged into the Linux kernel (2.6.20). KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V) where VMs run as normal Linux processes. As KVM is installed on top of the host OS, it is considered a Type 2 hypervisor. However, the KVM kernel module turns the Linux kernel into a Type 1 bare-metal hypervisor, providing the power and functionality of even the most complex and powerful Type 1 hypervisors.[1]

## Virtual Machine

Virtual machines are software computers that provide the same functionality as physical computers. Like physical computers, they run applications and an operating system. However, virtual machines are

computer files that run on a physical computer and behave like a physical computer. In other words, virtual machines behave as separate computer systems.[3]

## Xen Vulnerability

We have tried to exploit two vulnerabilities of XEN. The description of both of them is given below.

## Vulnerability 1

Xen allows page tables of the same level to map each other as read-only in PV domains. This is useful if a guest wants to use the self-referential page table trick for easy access to page tables by mapped virtual address. Xen will recursively drop the typed recounts of pages referenced by the page table, potentially recursively cleaning them up as well. For normal page tables, the recursion depth is bounded by the number of paging levels the architecture supports. However, no such depth limit exists for page tables of the same depth that map each other.[4]

## Vulnerability 2

This is a vulnerability in memory\_exchange() that permits PV guest kernels to write to an arbitrary virtual address with hypervisor privileges. Access\_ok() only checks the address, not the size, if the address points to guest memory, based on the assumption that any caller of access\_ok() will access guest memory linearly, starting at the supplied address. .[5]

## Torshammer DDOS

Torshammer is a slow-rate HTTP POST. Torshammer executes a [DoS attack](https://security.radware.com/ddos-knowledge-center/ddospedia/dos-attack/) by using a classic slow POST attack, where HTML POST fields are transmitted in slow rates under the same session (actual rates are randomly chosen within the limit of 0.5-3 seconds).

The slow POST attack causes the webserver application threads to await the end of boundless posts in order to process them. This causes the exhaustion of the web server resources and causes it to enter a denial-of-service state for any legitimate traffic.[6]

## Hping3

Hping3 allows the transmission of manipulated packets. This tool allows us to control the size, quantity and fragmentation of packets in order to overload the target and bypass or attack firewalls. Hping3 can be useful for security or capability testing purposes, using it you can test firewalls effectivity and if a server can handle a big amount of packets.[7]

## Trace

To check the behaviour of the processes we need some information and to do so and here we are using a system call. In order to get system calls, we are using strace.

## Strace

Strace is a diagnostic, debugging [and instructional userspace](https://en.wikipedia.org/wiki/Userspace) utility for [Linux](https://en.wikipedia.org/wiki/Linux). It is used to monitor and tamper with interactions between [processes](https://en.wikipedia.org/wiki/Process_(computing)) and the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel), which include [system calls](https://en.wikipedia.org/wiki/System_call), [signal](https://en.wikipedia.org/wiki/Unix_signal) deliveries, and changes of process state. The operation of strace is made possible by the kernel feature known as [ptrace](https://en.wikipedia.org/wiki/Ptrace). The most common use is to start a program using strace, which prints a list of system calls made by the program. This is useful if the program continually crashes, or does not behave as expected; for example, using strace may reveal that the program is attempting to access a file which does not exist or cannot be read.[8]

## ADFA-LD Dataset

ADFA Linux data set (ADFA-LD) is released recently for substituting the existing benchmark data sets in the area of host-based anomaly detection which has lost most of their relevance to modern computer systems. ADFA-LD is composed of thousands of system call traces collected from a contemporary Linux local server, with six types of up-to-date cyberattack involved. Australian Defence Force Academy Linux Dataset (ADFA-LD) and Australian Defence Force Academy Windows Dataset (ADFA-WD) are new generation system calls datasets that contain labelled system call traces for modern exploits and attacks on various applications.[9]

## Machine Learning Models

We have applied various Machine Learning Algorithms for training the Dataset.

### Random Forest Random forests

Random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.[10]

### Logistic Model

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model . Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labelled "0" and "1".[11]

### Naive-Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.[12]

### KNN Classifier

The basic logic behind KNN is to explore your neighbourhood, assume the test datapoint to be similar to them and derive the output. In KNN, we look for k neighbours and come up with the prediction. In

case of KNN classification, a majority voting is applied over the k nearest data points whereas, in KNN regression, the mean of k nearest data points is calculated as the output. [13]

### Decision Tree

A Decision tree is a flowchart like a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification. [14]

### Support Vector Machines

The objective of the support vector machine algorithm is to find a hyperplane in N-Dimensional space (N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, many possible hyperplanes could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features.[15]

# Chapter 3

**Related Work**

Hypervisor attacks are categorized as external attacks and defined as exploits of the hypervisor's vulnerabilities which allow attackers to gain accessibility and authorization over the hypervisors [19]. In support of hypervisor defence, Perez-Botero et al. characterized Xen and KVM vulnerabilities based on hypervisor functionalities in 2012 [15]. However, these vulnerabilities cannot be used as the basis for characterizing many recent attacks. Using the NIST 800-115 security testing framework, Thongtua et al [16] assessed the vulnerabilities of widely used hypervisors, including VMware ESXi, Citrix XenServer, and KVM then performed some sample experiments in order to derive severity scores, and attack impacts. In an effort to develop hypervisor forensic methods, researchers discussed the attacks on hypervisors, their forensic mechanisms and challenges [18], and leveraged existing memory forensic techniques to perform forensic analysis on hypervisor attacks [17].

# Chapter 4

**Proposed Method**

Project is divided into three tasks:-

## LAUNCHING ATTACKS

Exploiting the vulnerabilities of Hypervisor and launching an attack to check the behaviour of hypervisor under malicious process. The work associated with task-1 is described below:-

## Xen-Vulnerability

To record the system calls of two of the XEN vulnerabilities. We tried to execute them but each time the attack was launched the VM crashed. So the work is still under progress.

The description of both the vulnerabilities is as follows:- This vulnerability leads to a stack overflow in XEN.

It permits PV guest kernels to write to an arbitrary virtual address with hypervisor privileges.

## TCP-IP Hping3 Attack

Flood DOS attack on the VM causing the loss of further receiving packets.

## Torshammer DDOS

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic. We implemented a Torshammer DDOS attack from VM as an attacker to HOST as a victim. We also tried the attack from VM to MNIT’s site which led to the webpage being unresponsive.

## Buffer Overflow

## KeyLogger

* + 1. Meanwhile, we also tried to implement a **rootkit attack on docker** which was capable of escaping from Guest OS to Host OS.

## TRACE

Tracing of the information/data collection using strace.To check the behaviour of the processes we need some information to do so and here we are using system call. In order to get system calls we have used some methods:-

1. **Strace** command.
2. Trace using **python code.**

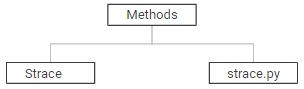


Figure 2 Methods used for tracing

## DETECTION

In order to predict hypervisor introspection, we have assumed that system call signature can give us some lead or can be used to predict the attacks. So we used signatures for system calls. And to get traces (system calls) of processes to make a dataset of system calls ,we have to run some attacks and normal processes and gather their traces . We have got the traces of the normal processes but if we talk about attacks there are some attacks we have performed but those are not sufficient for making a dataset . After a lot of research we found a dataset ADFA-LD which uses signature of system calls to determine whether it is attack or not .Our methodology of using system calls as parameters for deciding whether a process is a attack or not is same as used by them.So we used ADFA-LD ,

assuming that we would get the same type of dataset of system calls. We preprocessed the data and applied some ML algorithms. So assuming this dataset we applied ML Algorithm on this dataset.The result obtained is given in table.

# Chapter 5

# Experimental Results

The Attacks launched till now are Torshammer-DDOS , Hping3 attack , etc. The screenshots of the results are attached below:-

## TCP/IP

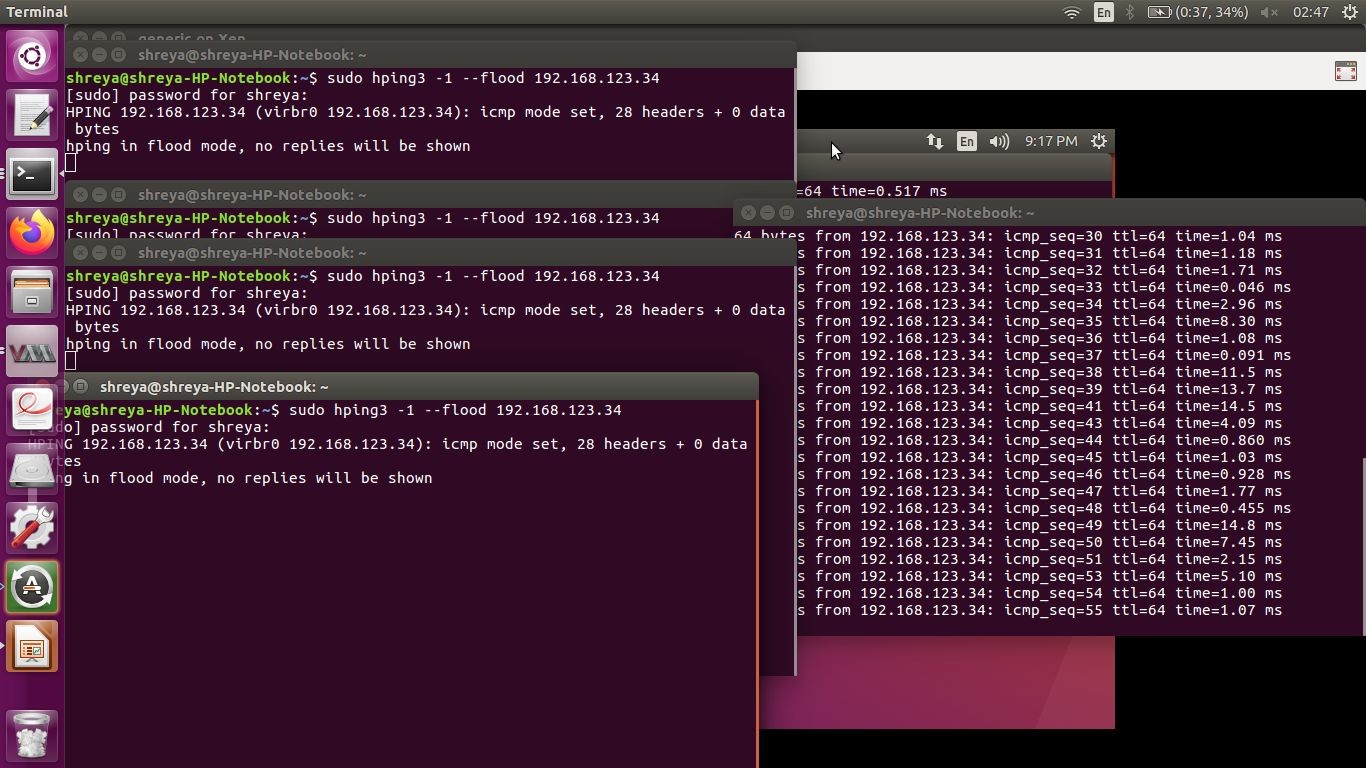


Figure 3 Hping3 attack launched

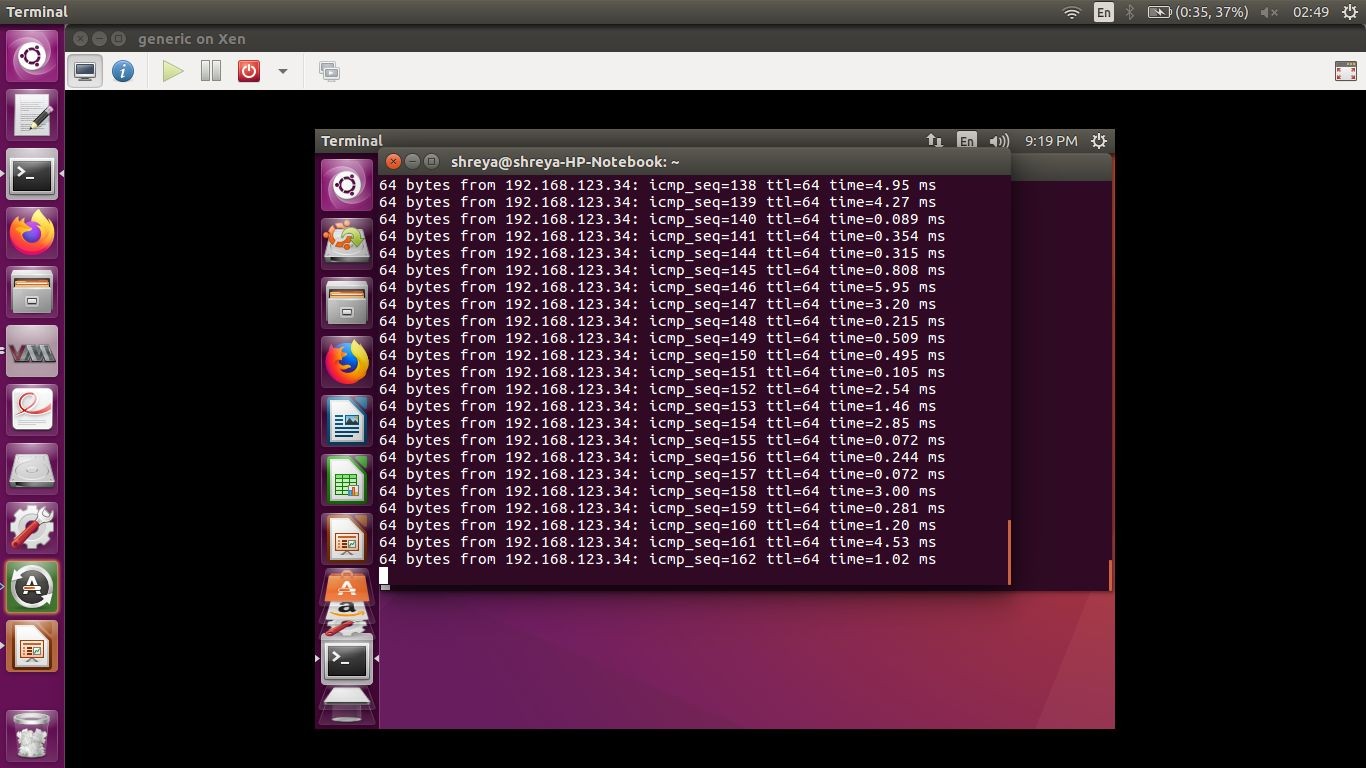


Figure 4 Hping3 attack results

## Torshammer DDOS

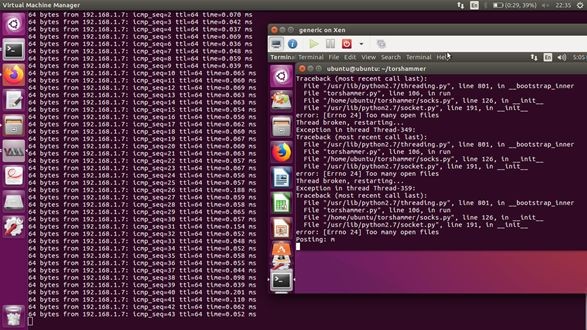


Figure 5 Torshammer attack

## Rootkit Attack

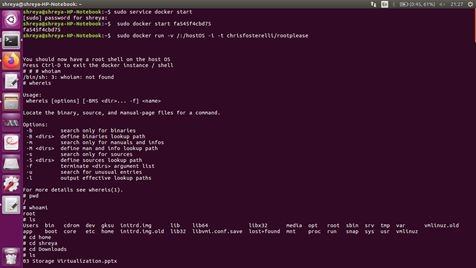


Figure 6 Rootkit attack

## ADFA-LD

In the detection part we applied various ML algorithms on the ADFA-LD Dataset and the results obtained are depicted in the table.The best results were obtained from Gaussian Naive Base with an accuracy of (93%).



Comparison of classification model

# Chapter 6

# Conclusion

In the era of cloud computing , the existence of a hypervisor is commonplace and hence its role is very significant. Thus a compromise in hypervisor could pose to be a very big threat . It has also been proposed to utilize even lower layers for detection, such as the System Management Mode as the similar types of processes emulate the same behaviour in terms of low level information like CPU cycles , memory bound and usages , system calls , hypercalls etc. Detection of attacks on the basis of system-calls is possible and is an efficient method. We perform Hypervisor Introspection by launching hypervisor attacks and recording their respective system calls signature followed by training the system on the dataset obtained from the collection of such system calls.

Our future works include the generation of a dataset of different processes consisting of both malicious and normal processes. We are also finding new methods for preprocessing the dataset and more

algorithms of classification models .

# References

1. https://csrc.nist.gov/publications/detail/nistir/8221/final
2. [https://www.howtogeek.com/66734/htg-explains-what-is-a-hypervisor/](http://www.howtogeek.com/66734/htg-explains-what-is-a-hypervisor/)
3. [https://www.vmware.com/topics/glossary/content/virtual-machine](http://www.vmware.com/topics/glossary/content/virtual-machine)
4. [https://www.exploit-db.com/exploits/43014](http://www.exploit-db.com/exploits/43014)
5. [https://www.exploit-db.com/exploits/41870](http://www.exploit-db.com/exploits/41870)
6. https://security.radware.com/ddos-knowledge-center/ddospedia/tors-hammer/
7. https://linuxhint.com/hping3/
8. https://en.wikipedia.org/wiki/Strace

[9] https://link.springer.com/chapter/10.1007/978-3-319-11698-3\_44

1. https://en..org/wiki/wikipediaRandom\_forest
2. https://en.wikipedia.org/wiki/Logistic\_regression
3. [https://www.geeksforgeeks.org/naive-bayes-classifiers/](http://www.geeksforgeeks.org/naive-bayes-classifiers/)
4. https://towardsdatascience.com/comparative-study-on-classic-machine-learningalgorithms-24f9 ff6ab222
5. [https://www.geeksforgeeks.org/decision-tree/](http://www.geeksforgeeks.org/decision-tree/)
6. https://towardsdatascience.com/support-vector-machine-introduction-to-machinelearning-algorit hms-934a444fca47
7. Perez-Botero D, Szefer J, Lee RB (2013) Characterizing hypervisor vulnerabilities in cloud computing servers. Proceedings of the 2013 International Workshop on Security in Cloud Computing, (ACM, Hangzhou, China), pp 3-10. https://doi.org/10.1145/2484402.2484406
8. Thongthua A, Ngamsuriyaroj S (2016) Assessment of hypervisor vulnerabilities. Proceedings of the 2016 International Conference Cloud Computing Research and Innovations (ICCCRI), (IEEE,

Singapore, Singapore), pp 71-77. https://doi.org/10.1109/ICCCRI.2016.19

1. Graziano M, Lanzi A, Balzarotti D (2013) Hypervisor memory forensics. Proceedings of the International Workshop on Recent Advances in Intrusion Detection, (Springer, Rodney Bay, St. Lucia), pp 21-40. https://doi.org/10.1007/978-3-642-41284-4\_2
2. Joshi LM, Kumar M, Bharti R (2015) Understanding threats in hypervisor, its forensics mechanism and its research challenges. International Journal of Computer Applications 119(1):1-5. https://doi.org/10.5120/21028-2755
3. Shi J, Yang Y, Tang C (2016) Hardware assisted hypervisor introspection. SpringerPlus, 5:647. https://doi.org/10.1186/s40064-016-2257-7