Detection of Hypervisor Attacks in Virtual Machines

FNU Aayoush
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, USA
fnu.aayoush01@student.csulb.edu

Kashmira Lokhande
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, USA
kashmiraanant.lokhande01@student.csulb.edu

Mamta Pednekar
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, USA
mamta.pednekar@student.csulb.edu

Siddhesh Gawde
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, USA
siddheshshamsundar.gawde01@student.csulb.edu

***Abstract*—With the recent technology developments and hardware cost reduction, Virtualization gained significant percentage of share in every sector in the industry. Virtualization ensured the Information technology delivery in the easier manner. At the same time, it exposed number of vulnerabilities to breach their security. With every upgrade in the technology the intruders are also upgrading themselves to find the new ways to hack the systems. On the virtual machine, hypervisor is a native software that lies between virtual machine and hardware. Hypervisor is responsible for allocating resources to virtual machines and flow of information between them and hence, most prone area for the attackers to gain control of the virtual machine. This paper proposes a framework to detect the hypervisor attacks in virtual machines. We used a k-nearest neighbors and support vector machine classifies on the publicly available dataset. We used two virtual machines VMWare and XEN vulnerability datasets to develop based on real time attack.**

***Index Terms*—Hypervisor, Virtual Machine, Support Vector Machine, K-Nearest Neighbor, RapidMiner, XEN, VMWARE, Niave Bayes**

I. Introduction

A. Virtual Machines

Virtual Machines are the virtual devices that are installed on the one physical machine such that user gets facility of operating multiple operating systems without changing the underlying physical device [2]. It computes the resource that uses software instead of physical computer to run programs and deploy apps [3]. This results in multiple benefits like increasing resource utilization, saving the time and space, enable on demand resources for users and decrease the power consumption. To implement the virtual machines a software is used which lies between the hardware and Virtual Machine and called as Hypervisor. In this paper, we have discussed on the analysis of hypervisor attacks on two Virtual Machines XEN and VMWare. VMware is also known as the embedded hypervisor that runs on direct hardware. The hypervisor is known to run without the need for an additional underlying Operating System on the physical device. XEN, on the other hand, permits multiple guest Operating Systems to run on the same physical device at the same time. We used the K-Nearest Neighbor and Support Vector Machine Classifiers to classify the vulnerabilities on these VMs, and the Rapid Miner tool to detect the density of the hypervisor attack.

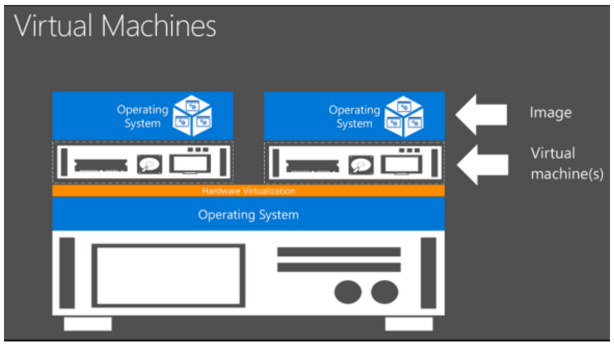


Fig. 1. Virtual Machine

B. Hypervisors

Hypervisor or virtual machine monitor (VMM) is a software that lies between virtual machine and hardware. Hypervisors create a virtualization layer that isolates the actual resources such as CPUs and processors from the virtual machines you construct. One host computer can support multiple guest VMs using hypervisors to virtually share its resources, such as memory and processing. It allocates system's available resources as the guest VM is independent of the host's hardware. The basic functions of hypervisors are to create, terminate, and move virtual machines, as well as to allocate resources [2]. Figure 1 depicts the two types of hypervisors:

1) Type 1 (also called bare metal or native): Type I hypervisors are software layers that are deployed directly on top of physical servers and their underlying hardware.

Furthermore, by interacting directly with the host system, resources are scheduled and allocated to the VMs. Examples of type 1 hypervisors are XEN VMWare ESX, and KVM.

2) Type 2 (also known as hosted hypervisors): This form of hypervisor runs inside a physical host machine's operating system. In addition, the built-in guest operating system allows the host system to modify the virtual machines. Examples of type 2 hypervisors are Oracle VM, VMWare Fusion.

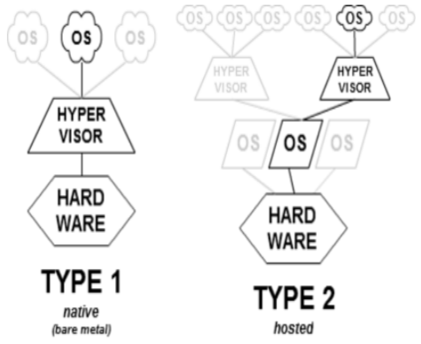


Fig. 2. Types of hypervisor

C. Hypervisor attacks

The hypervisor keeps track of important information (e.g., Domain list, EPT page table for VM, Shared pages between hypervisor and VM) that is used to administer and safeguard the virtual machines. There is no doubt that if the integrity of the hypervisor's code and critical data is hacked, the security of virtual machines will be jeopardized. An attacker initially launches an attack on a particular virtual machine that uses the hypervisor, which controls several virtual machines on a virtual host. Subsequently, malware then spread the attacks to all the virtual machines (VMs) that are running on the hacked hypervisor [2]. Consequently, the use of virtual machine resources rises, resulting in a denial of service for the entire host. Few famous examples of hypervisor attacks are:

1) BluePill: It's a piece of malware that starts a thin hypervisor and virtualizes the remainder of the machine to trap a running instance of the operating system.
2) DKSM: DKSM (Direct Kernel Structure Manipulation) attack is based on the observation that any introspection tool relies on the kernel data structures of a guest VM, which means a DKSM attack can change these kernel data structures in a variety of ways.
3) SubVirt: Attackers install a hypervisor underneath an existing operating system and use that VMM to host arbitrary malicious software. The resulting malware can perform general-purpose functions and can entirely disguise its state and activities from intrusion detection systems running on the target operating systems.

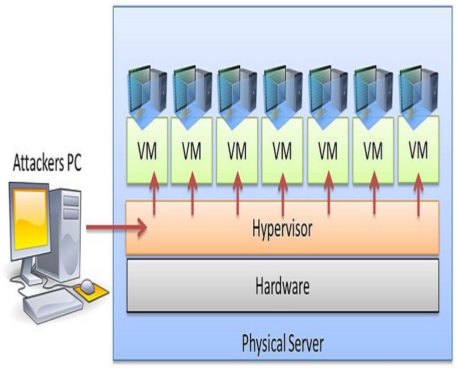


Fig. 3. Hypervisor attack

D. Parameters used for studying hypervisor vulnerability

In this paper the vulnerabilities of two hypervisors will be taken into consideration which are XEN and VMware. After considering few publicly available databases we choose the datasets on the CVE website for both XEN and VMWARE. This data has many features such as CVE ID, CWE ID, Type of Vulnerability, Score, Gained Access Level, Access, Complexity, Authentication, Confidentiality, Integrity and Availability. But the features that characterized the vulnerabilities of these hypervisors towards attack are as follows:

1) Confidentiality Impact: Confidentiality Impact was chosen as a characteristic in our study since data confidentiality is the most vulnerable attribute that leads to attacks. Measures must be taken to ensure privacy, with the goal of preventing sensitive data from reaching the wrong people.
2) Authentication: It is mechanism through which access to the resources is provided based on identity. It is a crucial attribute which is considered in vulnerability study of hypervisor because it important to achieve strong authentication, so that any malicious users can be restricted.
3) Integrity Impact: It is a critical attribute that has been chosen because data must not be altered, and actions must be made to ensure that information cannot be modified by unauthorized personnel.

E. Naïve Bayes model

One of the proposed frameworks for detecting the hypervisor attacks in virtual machines is Naïve Bayes model. We performed predictive analysis based on the Naive Bayes Theorem. After preprocessing and splitting the attacks into train and test data the model takes the three input parameters i.e., Integrity Impact, Confidentiality Impact, Authentication and makes use of Gaussian Naïve Bayes classifier to classify the vulnerabilities into Hypervisor Attacks and Non-hypervisor attacks. Model successfully predicted test data and was 80 percent(XEN) and 73 percent(VMware) accurate.

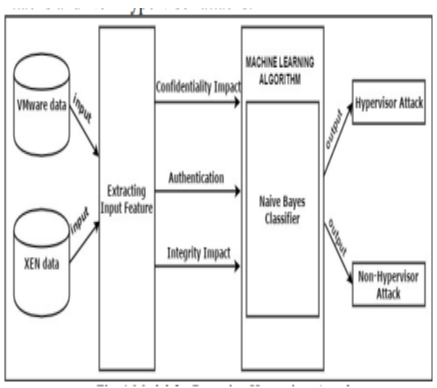


Fig. 4. Naive Bayes Model

F. k-Nearest Neighbors

k-Nearest Neighbors (KNN) is the non-parametric classification technique. In k-NN, the inputs consist of 'k' closest training samples. The output is class membership that means the output must be one of the values mentioned in the class. An object classification decided by a majority vote of its neighbors, with the object allocated to the most common class among its k closest neighbors. For e.g., if k =3, the object is assigned to the class occuring most among its 3 neighbors. In the above diagram, the green round is the candidate for

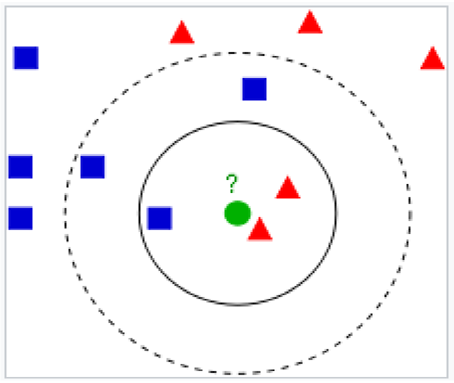


Fig. 5. k-Nearest Neighbor with k=3

classification. The three nearest neighbors are blue square and red triangle. The quantity of the red triangles are more i.e 2 hence, the green circle would be assigned to class red triangle. The function is only approximated locally in k-NN classification, and all computation is postponed until the function is evaluated. Because this method relies on distance for classification, normalizing the training data can greatly increase its performance if the features represent various physical units or come in wildly different scales.

G. Support Vector Machine

Support vector machine (SVM) is the supervised learning model. Like k-NN, SVM is also used for classification but there is a difference in its implementation and working. In SVM, the inputs are divided into two classes and the output is class membership. This model is created by dividing data into two hyperplanes. Hyperplanes can be divided using line, curve or an area. The output class would be decided on the basis of which side of the hyperplane the classification candidate is lying and that class would be assigned to the new member. The classification based on the SVM learning is shown in

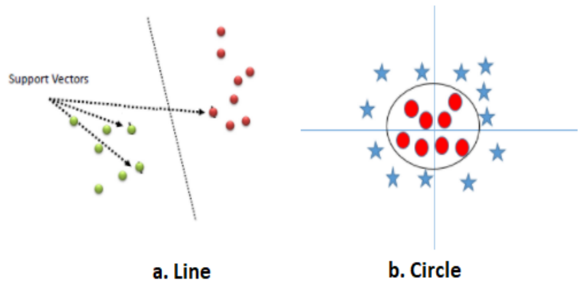


Fig. 6. Support Vector Machine

the Fig. 6. a.line figure illustrate that the line is used as the divider between left and right hyperplane whereas, In b.Circle, the are is used as the hyperplane seggregator, all the members which enclosed inside the circle would have same class and the members lying outside of the circumference of the circle would be assigned to other class.

II. Proposed Work

The Fig.1 illustrates the framework for detecting hypervisor attacks in the XEN and VMWARE virtual machines using K-Nearest Neighbors and Support Vector Machine. The input parameters primarily considered are Integrity, Confidentiality and Authentication respectively along with major attributes like Vulnerability type, Score, Access level, Complexity and Availability. Using the two machine learning model, we were able to classify the Hypervisor and the Non-Hypervisor attacks on the two virtual machines.

Steps:

1) Read the vulnerability dataset of the virtual machine.
2) Data Preparation involves various substeps like data cleansing, converting raw data into data primed for analysis and processing, parameter selection and lastly describing the training and the testing set of data. The process of data cleansing included removing extra information from the dataset, managing missing data using imputation, processing the attributes and following different cleansing and data processing for different machine learning models
3) Select the machine learning model.
4) Training data: In this phase, the model learns through the training set of data. How good the training data is, directly affects the quality of the model. It is the most important data which enables to model to learn and predict.

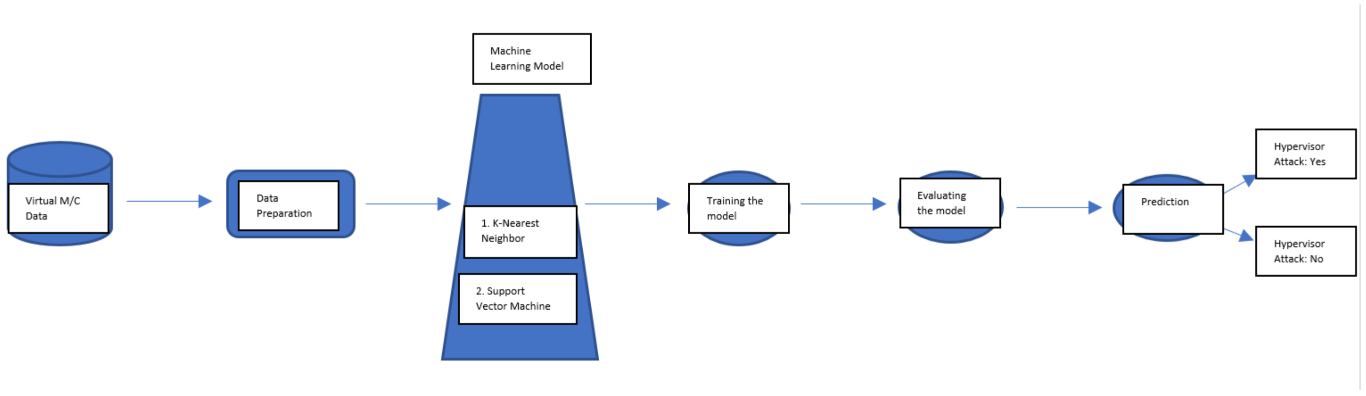
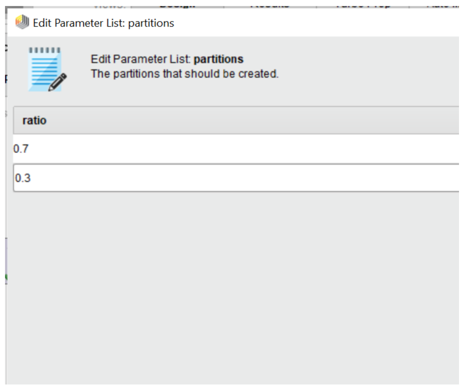


Fig. 7. Detection of Hypervisor Attacks in the Virtual Machine

5) Model is evaluated based on a set of metrics such as Accuracy, confusion matrix and F1 Score.
6) After training, the model is prepared to predict an output such as whether it is a hypervisor attack or not a hypervisor attack. It is applied on a new data and eventually it forecasts the outcome based on the new testing data supplied to the model.


Edit Parameter List: partitions

Edit Parameter List: **partitions**
The partitions that should be created.

ratio
0.7
0.3

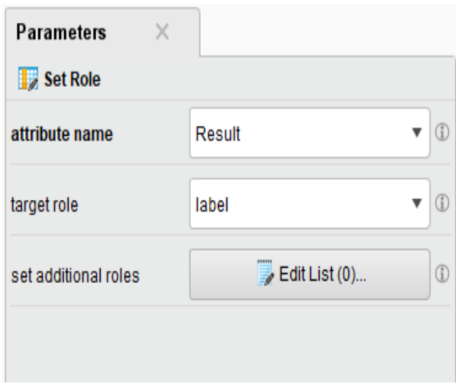
Fig. 8. Split Data

III. Experimental Work

The proposed framework was implemented on two virtual machines namely the XEN and the VMWARE. It was conducted using both Python and RapidMiner. The XEN and the VMWARE vulnerability dataset from the CVE website were used. Firstly, we applied the K-Nearest Neighbors machine learning model to detect the vulnerabilities. Secondly, we used the Support Vector Machine model to detect it and later compared the performance with the Naive Bayes Model. Fig.2 and Fig.3 showcases the process for XEN and VMWARE using K-Nearest Neighbor in RapidMiner. Fig.4 and Fig.5 showcases the process for XEN and VMWARE using Support Vector Machine.

Important terms while generating process in RapidMiner

1) Set Role: Setting a target attribute for the data i.e whether a hypervisor or a non-hypervisor attack is detected.

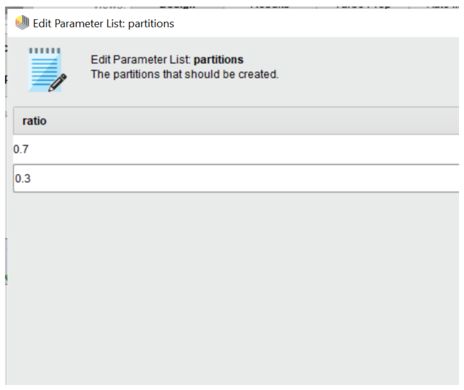

Parameters

Set Role

attribute name	Result
target role	label
set additional roles	Edit List (0)...

Fig. 9. Set Role

2) Split Data: Setting the ratios for training and testing data i.e 0.7 for training and 0.3 for testing.


Edit Parameter List: partitions

Edit Parameter List: **partitions**
The partitions that should be created.

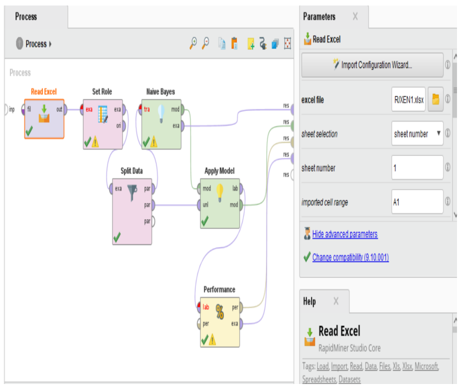
ratio
0.7
0.3

Fig. 10. Split Data

3) Performance: Various measures such as Accuracy, Precision, Recall and a graphical representation is observed.

A. Naive Bayes

Fig.4 and Fig.5 represents the process generated for the XEN and the VMWARE dataset. The operators involved are Read CSV, Set role, Naive Bayes Model, Split data, Apply model and performance. After the model is applied, we were able to classify the attacks and analyze the performance.



Read Excel

Import Configuration Wizard...

excel file: R/XEN1.xlsx

sheet selection: sheet number

sheet number: 1

imported cell range: A1

Hide advanced parameters

Change compatibility (9.10.001)

Read Excel

RapidMiner Studio Core

Tags: Load, Import, Read, Data, Files, Xls, Xlsx, Microsoft, Spreadsheets, Datasets

Fig. 11. Process for XEN dataset using Naive Bayes

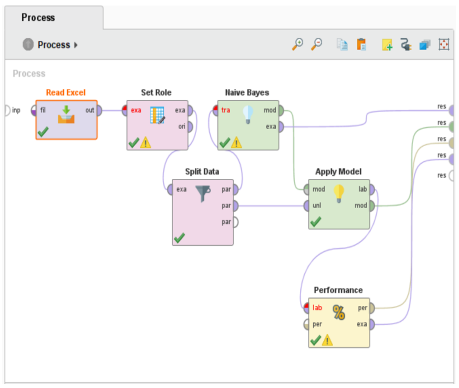


Fig. 12. Process for XEN dataset using Naive Bayes

B. K-Nearest Neighbor

Fig.6 and Fig.7 represents the process generated for the XEN and the VMWARE dataset. The operators involved are Read CSV, Set role, K-Nearest Neighbor Model, Split data, Apply model and performance. After the model is applied, we can analyze the performance, get a graphical representation for it and get a class predicted.

C. Support Vector Machine

Fig.8 and Fig.9 represents the process generated for the XEN and the VMWARE dataset. The operators involved are Read CSV, Set role, Support Vector Machine Model, Split data, Apply model and performance. After applying the model, a classification of the data is obtained.

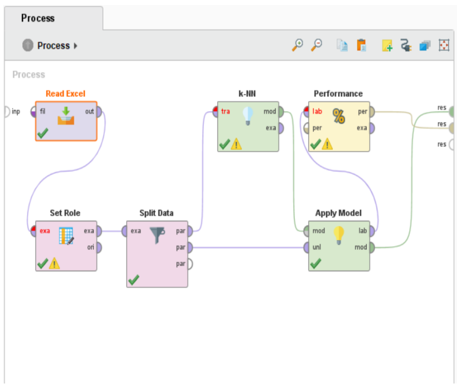
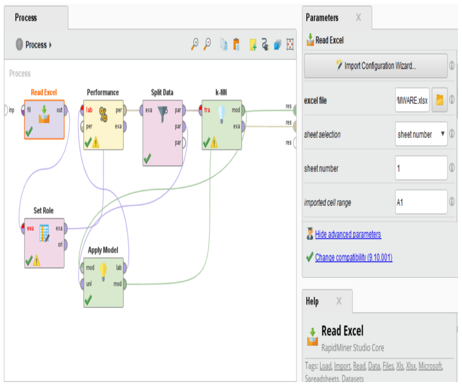


Fig. 13. Process for XEN dataset using K-Nearest Neighbor



Read Excel

Import Configuration Wizard...

excel file: 'MWARE.xlsx

sheet selection: sheet number

sheet number: 1

imported cell range: A1

Hide advanced parameters

Change compatibility (9.10.001)

Read Excel

RapidMiner Studio Core

Tags: Load, Import, Read, Data, Files, Xls, Xlsx, Microsoft, Spreadsheets, Datasets

Fig. 14. Process for VMWARE dataset using K-Nearest Neighbor

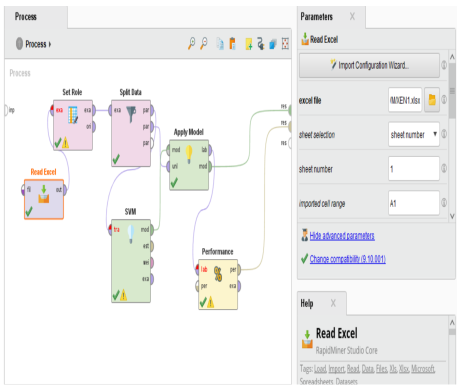
IV. Results

1) Confusion Matrix: A confusion matrix is a table that shows how well a classification model performs on a set of test data for which the true values are known.
2) Accuracy: It's the proportion of accurately anticipated observations to total observations. The model is the best if it has a high level of accuracy.
3) Precision: The ratio of accurately predicted positive observations to total expected positive observations is known as precision.
4) Recall: Recall is defined as the proportion of accurately predicted positive observations to all observations in the class.

A. XEN Virtual Machine Vulnerability Results

Fig. 10 shows the Naive Bayes model results for the XEN dataset. We observed an accuracy of 80 percent. In Fig. 11, we can observe the predicted and the actual values for the XEN dataset. We observe incorrect predicted results of 20 percent. In Fig.12, we observe a bar plot for confidentiality, integrity and Authentication with respect to the target variable. Target variable is the label which is reponsible for predicting whether an attack is a Hypervisor Attack or a Non-Hypervisor Attack.

Fig. 13, Fig.14 and Fig.15 represents an accuracy of 93.33 percent, a precision of 83.33 percent and a recall of 100 percent.



Parameters
Read Excel
Import Configuration Wizard...

excel file	/M/XEN1.xlsx
sheet selection	sheet number
sheet number	1
imported cell range	A1

Hide advanced parameters
Change compatibility (9.10.001)

Help
Read Excel
RapidMiner Studio Core
Tags: Load, Import, Read, Data, Files, Xls, Xlsx, Microsoft, Spreadsheets, Datasets

Fig. 15. Process for XEN dataset using Support Vector Machine

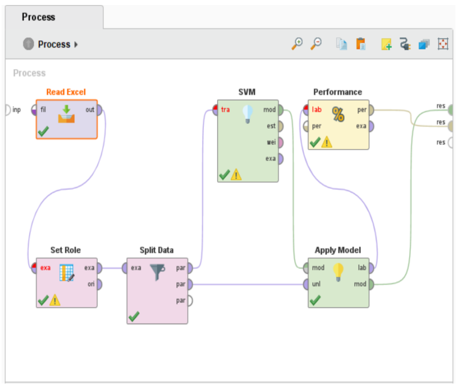
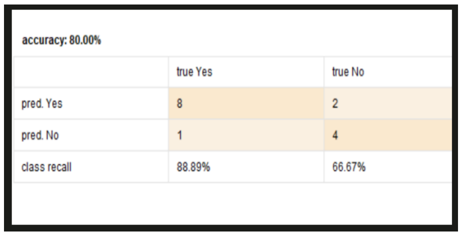
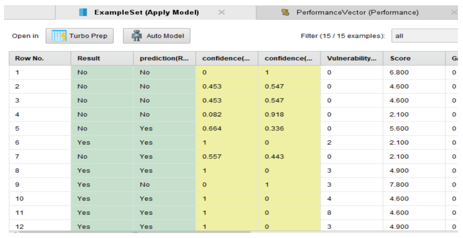


Fig. 16. Process for VMWARE dataset using Support Vector Machine


accuracy: 80.00%

	true Yes	true No
pred. Yes	8	2
pred. No	1	4
class recall	88.89%	66.67%

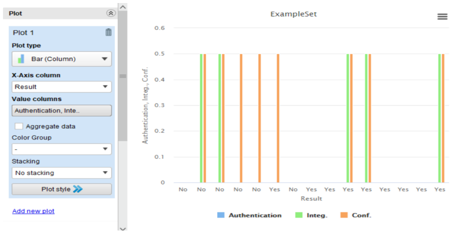
Fig. 17. Naive Bayes for XEN: Confusion Matrix and Accuracy


ExampleSet (Apply Model) PerformanceVector (Performance)

Open in Turbo Prep Auto Model Filter (15 / 15 examples): all

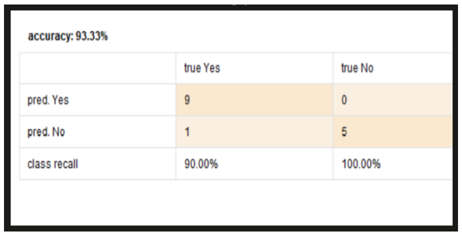
Row No.	Result	prediction(R...	confidence(...	confidence(...	Vulnerability...	Score	G
1	No	No	0	1	0	6.800	0
2	No	No	0.453	0.547	0	4.600	0
3	No	No	0.453	0.547	0	4.600	0
4	No	No	0.082	0.918	0	2.100	0
5	No	Yes	0.664	0.336	0	5.600	0
6	Yes	Yes	1	0	2	2.100	0
7	No	Yes	0.557	0.443	0	2.100	0
8	Yes	Yes	1	0	3	4.900	0
9	Yes	No	0	1	3	7.800	0
10	Yes	Yes	1	0	4	4.600	0
11	Yes	Yes	1	0	8	4.600	0
12	Yes	Yes	1	0	3	4.900	0

Fig. 18. Naive Bayes for XEN: ExampleSet



Plot type: Bar (Column)
X-Axis column: Result
Value columns: Authentication, Inte...
Aggregate data
Color Group: -
Stacking: No stacking
Plot style

Fig. 19. Naive Bayes for XEN: Graphical Representation


accuracy: 93.33%

	true Yes	true No
pred. Yes	9	0
pred. No	1	5
class recall	90.00%	100.00%

Fig. 20. K-Nearest Neighbor for XEN: Confusion Matrix and Accuracy

Fig. 16 shows an accuracy of 73.33 percent using Support Vector Machine.
Amongst the three machine learning models used, the best performance for the XEN Virtual machine was using K-Nearest Neighbor where we obtained an accuracy of 93.33 percent.

B. VMWARE Virtual Machine Vulnerability Results

We observed an accuracy of 73.33 percent using Naive Bayes in Fig. 17. In the following Fig.18, we can actually compare and analyse the predicted and the actual values for the VMWARE Vulnerabilities. We notice that around 73 percent of the predicted data is correct that is it matches the actual result of whether an attack is a hypervisor or a non-hypervisor attack. Secondly, an accuracy of 86.67 and 66.67 percents is observed for K-Nearest Neighbor and Support Vector Machine in Fig.19 and 20. We conclude that again, the KNN performed better than the other machine learning models for VMWARE virtual machine vulnerability dataset.

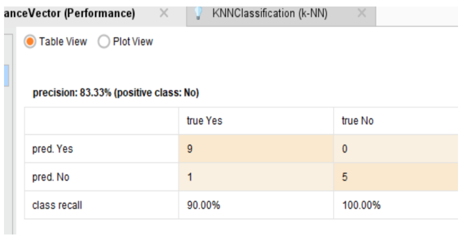

anceVector (Performance) | KNNClassification (k-NN)

Table View [✓] Plot View

precision: 83.33% (positive class: No)

	true Yes	true No
pred. Yes	9	0
pred. No	1	5
class recall	90.00%	100.00%

Fig. 21. K-Nearest Neighbor for XEN: Precision

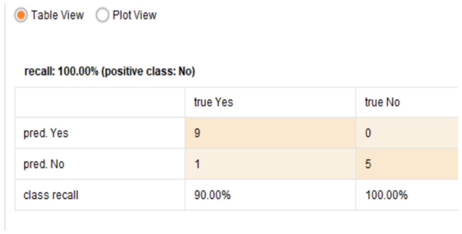
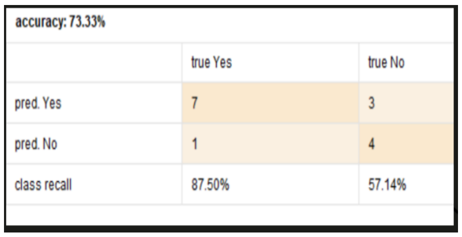


Table View [✓] Plot View

recall: 100.00% (positive class: No)

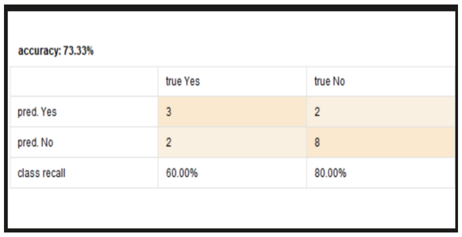
	true Yes	true No
pred. Yes	9	0
pred. No	1	5
class recall	90.00%	100.00%

Fig. 22. K-Nearest Neighbor for XEN: Recall


accuracy: 73.33%

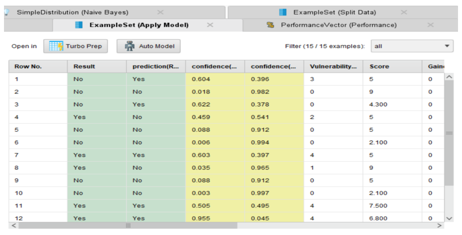
	true Yes	true No
pred. Yes	7	3
pred. No	1	4
class recall	87.50%	57.14%

Fig. 23. Support Vector Machine for XEN: Confusion Matrix and Accuracy


accuracy: 73.33%

	true Yes	true No
pred. Yes	3	2
pred. No	2	8
class recall	60.00%	80.00%

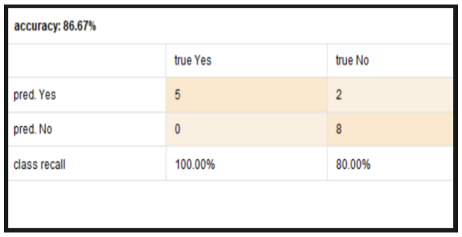
Fig. 24. Naive Bayes for VMWARE: Confusion Matrix and Accuracy


SimpleDistribution (Naive Bayes) | ExampleSet (Split Data) | ExampleSet (Apply Model) | PerformanceVector (Performance)

Open in Turbo Prep Auto Model Filter (15 / 15 examples): all

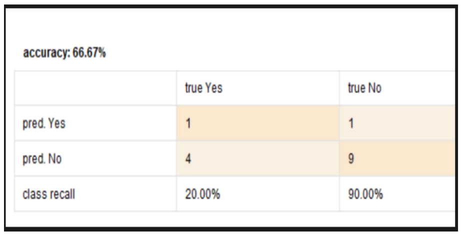
Row No.	Result	prediction(R...	confidence(...	confidence(...	Vulnerability...	Score	Gain
1	No	Yes	0.604	0.396	3	5	0
2	No	No	0.018	0.982	0	9	0
3	No	Yes	0.622	0.378	0	4.300	0
4	Yes	No	0.459	0.541	2	5	0
5	No	No	0.088	0.912	0	5	0
6	No	No	0.006	0.994	0	2.100	0
7	Yes	Yes	0.603	0.397	4	5	0
8	Yes	No	0.035	0.965	1	9	0
9	No	No	0.088	0.912	0	5	0
10	No	No	0.003	0.997	0	2.100	0
11	Yes	Yes	0.505	0.495	4	7.500	0
12	Yes	Yes	0.955	0.045	4	6.800	0

Fig. 25. Naive Bayes for VMWARE: ExampleSet


accuracy: 86.67%

	true Yes	true No
pred. Yes	5	2
pred. No	0	8
class recall	100.00%	80.00%

Fig. 26. K-Nearest Neighbor for VMWARE: Confusion Matrix and Accuracy


accuracy: 66.67%

	true Yes	true No
pred. Yes	1	1
pred. No	4	9
class recall	20.00%	90.00%

Fig. 27. Support Vector Machine for VMWARE: Confusion Matrix and Accuracy

C. *Virtual Machine Predicted Results using Python*

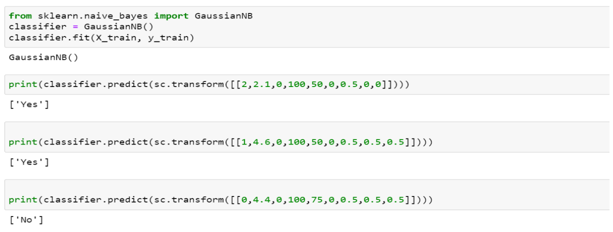

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

GaussianNB()

print(classifier.predict(sc.transform([[2,2.1,0,100,50,0,0.5,0,0]])))

['Yes']

print(classifier.predict(sc.transform([[1,4.6,0,100,50,0,0.5,0.5,0.5]])))

['Yes']

print(classifier.predict(sc.transform([[0,4.4,0,100,75,0,0.5,0.5,0.5]])))

['No']
```

Fig. 28. Naive Bayes Prediction Results

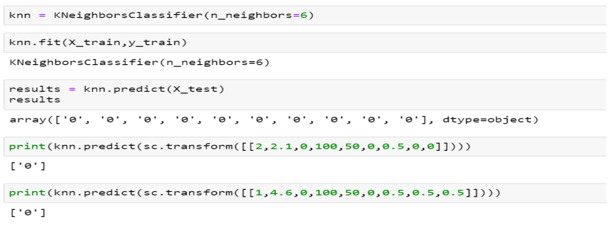
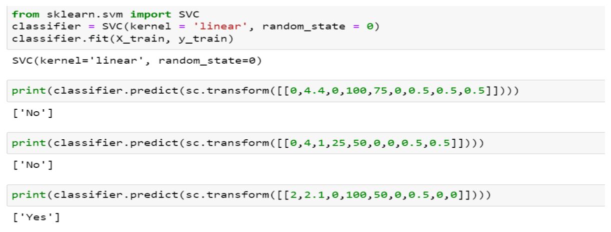

```
knn = KNeighborsClassifier(n_neighbors=6)

knn.fit(X_train,y_train)

KNeighborsClassifier(n_neighbors=6)

results = knn.predict(X_test)
results

array(['0', '0', '0', '0', '0', '0', '0', '0', '0', '0'], dtype=object)

print(knn.predict(sc.transform([[2,2.1,0,100,50,0,0.5,0,0]])))

['0']

print(knn.predict(sc.transform([[1,4.6,0,100,50,0,0.5,0.5,0.5]])))

['0']
```

Fig. 29. KNN Prediction Results

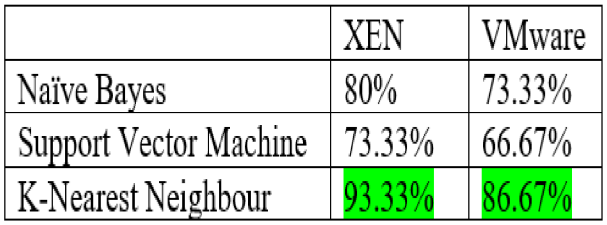


```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

SVC(kernel='linear', random_state=0)

print(classifier.predict(sc.transform([[0,4.4,0,100,75,0,0.5,0.5,0.5]])))

['No']

print(classifier.predict(sc.transform([[0,4,1,25,50,0,0,0.5,0.5]])))

['No']

print(classifier.predict(sc.transform([[2,2.1,0,100,50,0,0.5,0,0]])))

['Yes']
```

Fig. 30. SVM Prediction Results

V. Conclusion

Hypervisor is prone to attacks in the virtual machine. This issue has been addressed by identifying the attacks on hypervisor using machine learning in the research. So our experiment and work has concluded that the K-Nearest Neighbor has outperformed the other machine learning models used with a performance increase of about 16.5 percent. Table. 1 gives an acuuracy comparison for all the models on both the virtual machiness.



	XEN	VMware
Naïve Bayes	80%	73.33%
Support Vector Machine	73.33%	66.67%
K-Nearest Neighbour	93.33%	86.67%

Fig. 31. Comparison of Accuracies for Naive Bayes, KNN and SVM

References

[1] Sadia Ansar,Kanchan Hans,Sunil Kumar Khatri,"A Naive Bayes Classifier Approach for Detecting Hypervisor Attacks in Virtual Machines ", 2017 2nd International Conference on Telecommunication and Networks (TEL-NET 2017)

[2] Shreyal Gajare , Prof.Shilpa Sonawani, "Safeguard Hypervisor attacks in cloud computing" International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-3, Issue-12, December 2017.

[3] Gagandeep Kaur,Varsha Grover, "Detection and Prevention of Hypervisor and VM Attacks" , 2021 IEEE International Conference on Nanoelectronics, Nanophotonics, Nanomaterials, Nanobioscience Nanotechnology (5NANO)

[4] Louis Turnbull;Jordan Shropshire, "Breakpoints: An analysis of potential hypervisor attack vectors", 2013 Proceedings of IEEE Southeastcon

[5] Pezhman Sheinidashtegol,Michael Galloway, "Performance Impact of DDoS Attacks on Three Virtual Machine Hypervisors",2017 IEEE International Conference on Cloud Engineering (IC2E)