Rules Based Method For Software Vulnerabilities Levels Evaluation Using Machine Learning

DIAKO Doffou Jérôme, ACHIEPO Odilon Yapo M., MENSAH Edoete Patrice

Abstract- Cybercrime is on the increase with diversified attacks, particularly on software applications [1]. To combat software piracy, researchers have developed vulnerability scores using a standardized scoring mechanism to assess the level of vulnerability of software applications [2]. However, the calculation of this score is very complex in practice and difficult to use in industry and research [3]. In this paper we propose an approach to compute directely the software vulnerability level hat avoids the complexity of the existing method based on complex score calculation. Our method is developed from a modeling of international vulnerability data using decision tree approach and is based on rules expressed in first-order logic. Instead of generate vulnerability scores very close to those obtained by the complex classical method, our method is able to give directly the software vulnerability level, the advantage of being extremely simple to use both in research and in the industry.

Index Terms—Machine Learning, Software Vulnerabilities, CVSS Score, Vulnerability Levels, Decision Tree, R Programming language.

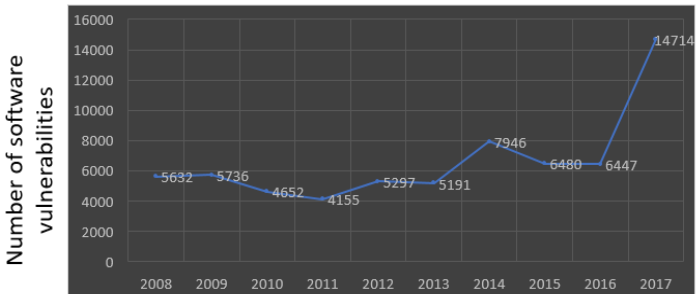
1 Introduction

In the field of computer security, a vulnerability is a weakness in a computer system that allows an attacker to compromise the integrity of that system, i.e. its normal functioning, confidentiality or integrity of the data it contains[4][5]. According to the research site [6] we see the number of existing software vulnerabilities and their trend from 2008 to 2017.

Fig. 1

Trends in software vulnerabilities

Source: https://www.cvedetails.com/vulnerabilities-by-types.php



Hackers can exploit software vulnerabilities to compromise the security of the computer systems. To remedy this problem, security service providers and non-profit organizations have expanded to implement procedures to classify vulnerabilities.

We have some organizations such as the National Vulnerability Database (NVD)[7], US-CERT, SANS[8], SECUNIA[9], ISS X-Force[10], VUPEN SECURITY[11] and many others institutions propose their own method for software vulnerability evaluation . Unfortunately, there has been no cohesion or interoperability between these systems. To effectively address this approach, researchers have developed vulnerability scores using a standardized scoring mechanism to assess the level of vulnerability of software applications [2].

However, this score is very complex in practice and difficult to use in industry and research because of the complexity of its calculation [3].

In this paper we propose an approach for evaluating software vulnerability levels that avoids the complexity of the initial software vulnerability score computing method based on scores calculation. Our method is developed from a modeling of international vulnerability data using decision tree technology. The developed method is reduce to some rules expressed in first-order logic and generated an assessment of vulnerability while having the advantage of being extremely easy to use both in industry and in research.

2 Related Works

In recent years, many IT security projects implemented by suppliers and non-governmental organizations have developed and implemented procedures to score vulnerable information systems. Companies such as Cisco System, Qualys, Symantec, Carnegie Mellon are promoting an evaluation system that will standardize the evaluation system called CVSS (Common Vulnerability Scoring System). In 2005, the National Infrastructure Advisory Council (NIAC) selected FIRST to be the custodian of the Common Vulnerability Assessment System (CVSS), the new standard for vulnerability scoring. This assessment system is designed to provide open and universal severity indices of software vulnerabilities. FIRST will work closely with CERT / CC and MITRE on this issue. The Common Vulnerability Scoring System (CVSS) provides a means of capturing the main characteristics of a vulnerability and producing a numerical score that reflects its severity [2].After using version 1 of the CVSS, more than a dozen CVSS-SIG members worked intensively in 2006 and 2007 to revise and improve CVSS v1 by testing hundreds of real vulnerabilities in order to implement CVSS v2[12].The CVSS v2 has been widely adopted by various organizations, it has been a help for

organizations in managing patches in measuring the severity of vulnerability.

3 Methods Of Vulnerability Assement

There are many vulnerability rating systems, which are supported by different organizations. In general, there are two ways to describe the severity of vulnerabilities. There is the qualitative evaluation system and the quantitative evaluation system. In this section, we will introduce two databases with different vulnerabilities.

3.1 Qualitative Method

IBM® X-Force® produces many leading security research assets to help customers, researchers and the general public better understand the latest security risks and anticipate new threats. Its database is one of the most comprehensive in the world on threats and vulnerabilities. It contains more than 40,000 vulnerabilities. IBM® X-Force® uses a qualitative vulnerability assessment method and assigns risk levels to each security problem to describe the extent of damage that could be caused by a specific security problem. In Table 1, there are three possible levels of risk of vulnerability [10].

TABLE 1
X-Force Vulnerability Assessment System.

Rating	Definition
High	Security issues that allow immediate remote or local access, or immediate execution of code or commands, with unauthorized privileges
Medium	Security issues that have the potential to grant access or enable code execution through complex or lengthy operating procedures, or low-risk issues applied to major internet components
Bass	Security problems that deny service or provide non-systemic information that could be used to formulate structured attacks against a target, but do not directly obtain unauthorized access.

The IBM® X-Force® Vulnerability Classification System method is a typical example of a qualitative method. In this method, security researchers directly give a level of threat (high, medium and low) for a vulnerability based on its attributes.

3.2 Quantitative Method

The NVD is the U.S. government's repository for standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data automates vulnerability management, security measures and compliance. The NVD includes databases of references to security checklists, security related software vulnerabilities, configuration errors, product names and impact indicators.
The NVD supports the Common Vulnerability Scoring System (CVSS) version 2 standard initiative for all CVE vulnerabilities. The NVD provides low, medium, and high severity scoring in addition to the numerical CVSS scores. CVSS is a well-known quantitative vulnerability rating system, as shown in Table 2.

TABLE 2
Vulnerability Assessment System By Cvss V2

Rating	Definition
High	Vulnerabilities will be classified as ''High severity'' if they have a base CVSS score of 7.0 to 10.0
Medium	Vulnerabilities will be classified as ''Medium severity'' if they have a base CVSS score of 4.0 to 6.9
Low	Vulnerabilities will be classified as ''Low'' if they have a base CVSS score of 0.0 to 3.9

The scores are calculated in order so that the base score is used to calculate the time score and the time score is used to calculate the environmental score.

4 Cvss Score Analysis

The CVSS is built from the base metric which gives us an evaluation of the base CVSS which will then be weighted with the time metric and then with the environmental metric.
These three metrics are defined as follows:

- ✓ The basic metric is unique and immutable; it is based on the intrinsic qualities of vulnerability.
- ✓ The time metric is unique but can change over time.
- ✓ Environmental metrics are multiple and evolve according to the IT environment. It depends on the computer system in which it is present.

These scores are calculated using a complex equation that is implemented in a calculator. Here is the presentation of this equation.
The equation is presented below CVSS Base Score Equation

$$BaseScore = (.6*\mathrm{Im}\,pact + .4*Exploitability - 1.5)*f(\mathrm{Im}\,pact) \quad (1)$$

$$Impact = 10.41*(1-(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))) \quad (2)$$

$$\mathbf{Exploitability} = 20*AccessComplexity*Authentication*AccessVector \quad (3)$$

$$\mathbf{f(Impact)} = 0 \ if \ Impact = 0;\ 1.176 \ otherwise \quad (.4)$$

- *DIAKO DOFFOU JEROME is a graduate student in computer science (EDP INPHB Yamoussoukro, Côte d'Ivoire), kingdjako@gmail.com*
- *ACHIEPO ODILON YAPO MELAINE is an Assistant Master in Artificial Intelligence (University of Korhogo, Côte d'Ivoire), kingodilon@gmail.com*
- *MENSAH EDOETE PATRICE is Professor in Theorical and Applied Mathematics (INPHB Yamoussoukro, Côte d'Ivoire),pemensah@hotmail.com*

To give the basic score of a vulnerability, this calculator uses six parameters which are

TABLE 3
CHARACTERISTICS OF SOFTWARE VULNERABILITY

CHARACTERISTICS	VALUES	CVSS WEIGHTS
Access vector	Local(L) Adjacent Network(AN) Network (N)	0.395 0.646 1
Authentication	None (N) Single(S) Multiple(M)	0.704 0.56 0.45
Access complexity	High(H) Medium(M) Low (L)	0.35 0.61 0.71
Integrity impact	None(N) Partial(P) Complete(C)	0.0 0.275 0.660
Confidentiality impact	None(N) Partial(P) Complete(C)	0 0.275 0.660
Availability impact	None(N) Partial(P) Complete(C)	0 0.275 0.660

Given the complexity, CVSS-SIG researchers implemented this equation as a calculator. As an illustration we have

Example of vulnerability score with CVSS v2 for CVE-2003-0062 vulnerability

TABLE 4
CVSS V2 CALCULATOR

BASE SCORE METRICS	
Exploitability Metrics	Impact Metrics
ATTACK VECTOR (AV) Local (AV: L) Adjacent Network (AV: A) Network (AV: N)	**CONFIDENTIALITY** IMPACT (C) None (C: N) Partial (C: P) Complete (C:C)
ACCESS COMPLEXITY (AC) High (AC:H) Medium (AC:M) Low (AC: L)	**INTEGRITY IMPACT (I)** None (I: N) Partial (I: P) Complete (I:C)
AUTHENTICATION (AU) Multiple (Au:M) Single (Au: S) None (Au: N)	**AVAILABILITY IMPACT (A)** None (A: N) Partial (A: P) Complete (A:C)

BASE METRIC EVALUATION SCORE

Access Vector [Local] (0.395)
Access Complexity [High] (0.35)
Authentication [None] (0.704)
Confidentiality Impact [Complete] (0.66)
Integrity Impact [Complete] (0.66)
Availability Impact [Complete] (0.66)

FORMULA BASE SCORE

Impact = 10.41*(1-(0.34*0.34*0.34)) == 10.0
Exploitability = 20*0.35*0.704*0.395 == 1.9
f(Impact) = 1.176
BaseScore =((0.6*10)+(0.4*1.9)-1.5)*1.1 ==(6.2)
(AV: L/AC:H/Au:N/C:C/I:C/A:C)

TABLE 5
CALCULATOR CVSS V2 ENABLED

BASE SCORE METRICS	
Exploitability Metrics	Impact Metrics
ATTACK VECTOR (AV) Local (AV: L) Adjacent Network (AV: A) Network (AV: N)	CONFIDENTIALITY IMPACT (C) None (C: N) Partial (C: P) Complete (C:C)
ACCESS COMPLEXITY (AC) High (AC:H) Medium (AC:M) Low (AC: L)	INTEGRITY IMPACT (I) None (I: N) Partial (I: P) Complete (I:C)
AUTHENTICATION (AU) Multiple (Au:M) Single (Au: S) None (Au: N)	AVAILABILITY IMPACT (A) None (A: N) Partial (A: P) Complete (A:C)

This gives a **score of 6.2** which is equivalent to an **average vulnerability**, see Table 2.

The difficulties of this approach is that you have to be a safety specialist to understand these six parameters that are already complex to understand, which leads to the complex use of the calculator.

Then the CVSS score cannot be calculated if a required attribute is not defined. In this case, the severity of the problem is categorized as Undetermined.

5 PROPOSED APPROACH

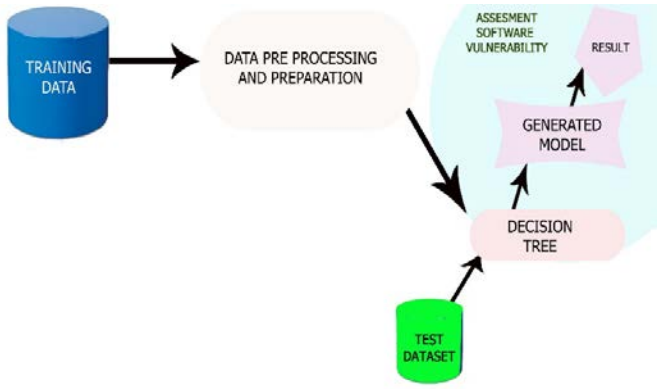
Our approach is to propose a learning model that allows us to assess the level of software vulnerabilities.
To achieve this, we will:

- ✓ Develop a vulnerability database from 2010 to 2018;
- ✓ Identify the essential parameters for assessing the level of software vulnerability;

- ✓ Predict the level of software vulnerabilities for building an artificial learning model.

This approach is illustrated by the following diagram

FIG. 2
PROCESS FOR CONDUCTING THE SOFTWARE VULNERABILITY LEVEL ASSESSMENT

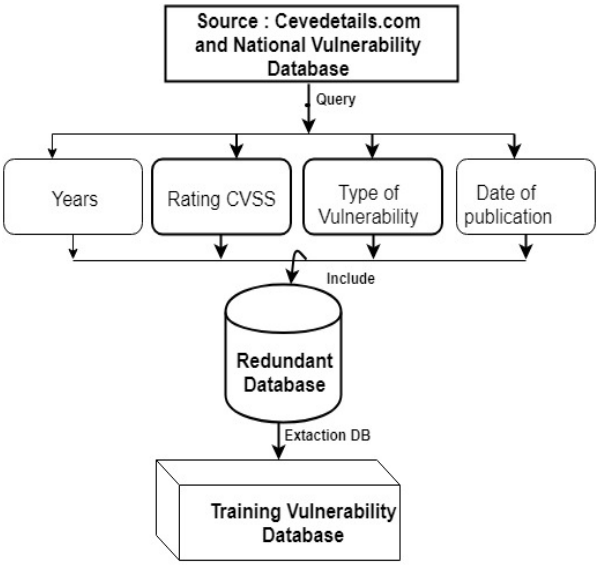


5.1 Building The Vulnerability Database

To build the database, we collected data from the NVD and Cvedetail.com [6], using queries on the CVSS score, years of vulnerability discovery and types of vulnerabilities. After receiving the data, we proceed to extract the redundant information. which allowed us to build our training database.

Illustrated by the following Fig 3 and Table 6:

FIG. 3
VULNERABILITY DATABASE ACQUISITION PROCESS



5.2 Data Preprocessing And Preparation

In this part we proceeded as follows:

*For data pre-processing, we have:

1- After building the database, we created the "Level" variable that will allow us to assess a vulnerability;

2- Remove unnecessary variables and correct errors (on AccessVector);

3- Partition data into learning and test samples.

FIG.4
DATA PREPROCESSING

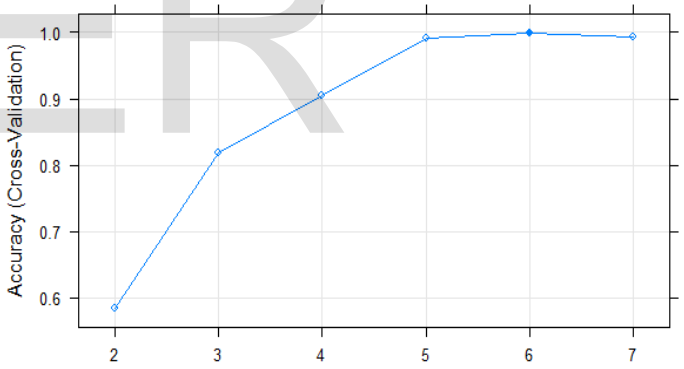
base	6670 obs. of 9 variables
dbApp	4003 obs. of 9 variables
dbTest	2667 obs. of 9 variables

* For the preparation of the data; we proceeded:

1- In Search of the Best Predictors (With the following learning sample): "AccessVector" "AccessComplexity" "AvailImpact" "Authentication" "ConfImpact" "IntegImpact"

As illustrated by the following graph:

FIG. 5
THE BEST PREDICTORS



2-For the following we have kept the best predictors, which allows us to go from 9 variables to 7 variables as shown here

FIG. 6
DATA PREPARATION

dbApp	4003 obs. of 7 variables
dbTest	2667 obs. of 7 variables

TABLE 6
DATASET OVERVIEW

xity	Authentication	Access Vector	Conf Impact	Integ Impact	Avail Impact
	Not required	Local	Complete	Complete	Complete
0864	Information				

5.3 Model Construction (Decision Tree)

Our model is based on decision trees. Decision Tree is a very popular classifier that serves as the basis for many set classifiers and is represented as its name suggests in the form of a tree graph [13][14][15][16][17]. Each of the tree structure represents a condition based on the characteristics, the branches below each represent the output of the condition and finally, the leaves represent the result of the algorithm of the decision tree. The decision tree is also very effective with categorical and tolerant data in the presence of outliers and missing data. The Decision Tree is more accurate and faster, more efficient in terms of memory usage.
Purpose: Evaluate software vulnerabilities based on CVSS settings.

To elaborate our model, we have written the following algorithm in R programming language. This algorithm that we have named "SVTree" will allow us to build rules to evaluate the level of software Vulnerability.

ALGORITHM

$Algorithme : SVTree$
$X: Database$
$dbApp: TrainData$
$R: Rules$
$Y: Software\ Vulnerability\ Score$
$C: Vulnerability\ Level$
$Begin$
(*Installation of packages*)
$Input\ (Package$: caret)
$Input\ (Package$: partykit)

(*Loading the database*)
Input (X)

(*Creating the "Level" variable*)
$if Y \leq= 3.9\ then$
$C < -$"Low"
$else\ if Y < 7\ then$
$C < -$"$Medium$"
else $C < -$High
End if
(*Data preprocessing and pretreatment*)
(* Partitioning of data into training and test samples*)

$numApp < -CreateDataPartion(X\$C, p = 0.6)$
$dbApp < -X[numApp,]$
$dbTest < -X[-numApp,]$

(* Search for the best predicators (with the training) *)
(* names of the best predicators*)
(*Choice of optimizable parameters*)
(*Optimal model construction*)
(* Importance of predicators in the prediction of each class*)
(*Building the optimal model (Training)*)

$Mtree < -train(C, dbApp)$

(*Display the importance of predictors in the prediction of each class*)
(*Display of the best tree*)

End

6 Logical Rules of the Model

TO build the model, we are divided the dataset in two parts: 60% to build the model (train dataset) using Leave One Out cross validation, and 40% (test dataset) to evaluate the quality of the model. The model obtained is

set of thirty-one (31) logical rules. Height (8) rules gives the conditions in which the software vulnerability level is low, fourteen (14) rules gives the conditions in which the software vulnerability level is medium and nine (9) rules gives the conditions in which the software vulnerability level is high.

6.1 Rules of a higher software vulnerability level

The rules characterizing software with high vulnerability are:

Rule H1:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (High OR Medium)] AND
[IntegImpact = Complete] AND
[AccessVector = (Network OR Remote)]
THEN LEVEL = High

Rule H2:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = Low] AND
[Authentication = (Multiple OR Single] AND
THEN LEVEL = High

Rule H3:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (Low)] AND
[Authentication = Not Required] AND
[AccessVector = (Network OR Remote] AND
[IntegImpact = (Complete OR Partial]
THEN LEVEL = High

Rule H4:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (High OR Medium)] AND
[AccessComplexity = Low] AND
[Authentication = Not Required)] AND
[AccessVector = (Network OR Remote)] AND
[IntegImpact = (Complete OR Partial)]
ALORS LEVEL = High
Rule H5:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = Low] AND
[Authentication = Not Required] AND
[AccessVector = (Network OR Remote] AND
[IntegImpact = None]
THEN LEVEL = High

Rule H6:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector = (Network OR Remote] AND
[AccessComplexity = (High OR Medium)]
THEN LEVEL = High

Rule H7:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector =(Network OR Remote] AND
[AccessComplexity = Low] AND
[Authentication = Not required]
THEN LEVEL = High

Rule H8:
IF
[AvailImpact = None] AND
[ConfImpact = Complete]
THEN LEVEL = High

6.2 Rules of a medium software vulnerability level

The rules characterizing software with medium vulnerability are:

Rule M1:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (High OR Medium)] AND
[IntegImpact = Complete] AND
[AccessVector = Local]
THEN LEVEL = Medium
Rule M2:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (High OR Medium)] AND
[IntegImpact = (None OR Partial)] AND
[AccessVector =(Local OR Network)]
THEN LEVEL = Medium

Rule M3:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = (High OR Medium)] AND
[IntegImpact = (None OR Partial)] AND
[AccessVector = Remote]
THEN LEVEL = Medium

Rule M4:
IF
[AvailImpact = (Complete OU Partial)] AND
[ConfImpact = (Complete OU Partial)] AND
[AccessComplexity = Low] AND
[Authentication= (Multiple OR Single)]
THEN LEVEL = Medium

Rule M5:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = Low] AND
[Authentication= (Multiple OR Single)]
THEN LEVEL = Medium

Rule M6:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = (Complete OR Partial)] AND
[AccessComplexity = Low] AND
[Authentication= Not Required] AND
[AccessVector = (Network OR Remote)] AND
[IntegImpact= None]
THEN LEVEL = Medium

Rule M7:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector = (Local OR Network)] AND
THEN LEVEL = Medium

Rule M8:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector = Remote] AND
[AccessComplexity = Low] AND
[Authentication= Single]
THEN LEVEL = Medium

Rule M9:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector = Remote] AND
[AccessComplexity = (Low OR Medium)] AND
[Authentication= Not Required]
THEN LEVEL = Medium

Rule M10 :
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector = Remote] AND
[AccessComplexity = (Low OR Medium)] AND
[Authentication= Single]
THEN LEVEL = Medium

Rule M11:
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact = (Complete OR None)] AND
[AccessVector = Remote] AND
[AccessComplexity = (Low OR Medium)]
THEN LEVEL = Medium

Rule M12:
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact = (Complete OR None)] AND
[AccessVector = Remote] AND
[AccessComplexity = (Low OR Medium)] AND
[Authentication = Not Required]
THEN LEVEL = Medium

Rule M13:
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact = Partial] AND
[Authentication = Not Required] AND
[AccessVector = Remote] AND
[AccessComplexity = (Low OR Medium)]
THEN LEVEL = Medium

Rule M14:
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact = Partial] AND
[Authentication = Single] AND
[AccessComplexity = Low]
THEN LEVEL = Medium

6.3 Rules of a lower software vulnerability level

The rules characterizing software with lower vulnerability are:

Rule L1:
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector= (Local OU Network)]
THEN LEVEL = Low

Rule L2 :
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector= Remote] AND
[AccessComplexity= High]
THEN LEVEL = Low

Rule L3 :
IF
[AvailImpact = (Complete OR Partial)] AND
[ConfImpact = None] AND
[AccessVector= Remote] AND
[AccessComplexity= High]
THEN LEVEL = Low

Rule L4 :
IFI
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact= (Complete OR None)] AND
[AccessVector= Remote] AND
[AccessComplexity= High]
THEN LEVEL = Low

Rule L5 :
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact= (Complete OR None)] AND
[AccessVector= Remote] AND
[AccessComplexity= (Low OR Medium)] AND
[Authentication= Single]
THEN LEVEL = Low

Rule L6 :
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact= Partial] AND
[Authentication= Not Required] AND
[AccessVector = (Local OR Remote)]
THEN LEVEL = Low

Rule L7:
IF
[AvailImpact = None] AND
[ConfImpact = (None OR Partial)] AND
[IntegImpact= Partial] AND
[Authentication= Not Required] AND
[AccessVector= Remote] AND
[AccessComplexity= High]
THEN LEVEL = Low

Rule L8:
IF
[ConfImpact = (None OR Partial)] AND
[IntegImpact= Partial] AND
[Authentication= Not Required] AND
[AccessVector= Remote] AND
[AccessComplexity= High]
THEN LEVEL = Low

Rule L9:
IF
[AvailImpact = None] AND
[ConfImpact= (None OR Partial)] AND
[IntegImpact= Partial] AND
[Authentication= Single] AND
[AccessComplexity= Medium]
THEN LEVEL = Low

7. PERFORMANCE OF THE MODEL

The software vulnerability level develop in this paper is a decision tree classification model. To study the model performance, we use the most used performance metrics in machine learning. This metrics are the error prediction rate, the accuracy and the Kappa index for the global model quality; and the precision, the recall and the F1-measure for the quality of the prediction of each vulnerability level.

7.1 Global quality of the model

All the metric to measure the global quality of the model are based on the confusion matrix. The confusion matrix on the test dataset in show in the table below:

TABLE 7
CONFUSION MATRIX ON TEST DATASET

		Observed Vulnerability Levels		
		High	**Low**	**Medium**
Predicted Vulnerability Levels	**Higth**	1032	0	8
	Low	0	343	9
	Medium	11	9	1255

We can see that all correct predictions are located in the diagonal of the table, so prediction errors can be easily found in the table, as they will be represented by values outside the diagonal. This confusion matrix shows that the main test data have correctly been classify. The values of the global predictive quality indicators are:

- Accuracy : 0.9861267
- Kappa : 0.9769574
- Error prediction rate : 0.01387327

These indicator shows that the model is have a very good quality for predict the software vulnerability levels.

7.1 Quality of each vulnerability level prediction

For each software vulnerability level (predicted classes), the precision, the recall and the F1-measure compute using the test dataset are:

TABLE 8
DETECTION QUALITY OF EACH CLASS INDIVIDUALLY

		Precision	Recall	F1
Software Vulnerability Level	High	0.9923077	0.9894535	0.9908785
	Low	0.9744318	0.9744318	0.9744318
	Medium	0.9843137	0.9866352	0.9854731

These results show that the model predict very well each software vulnerability level. The advantage of our model is that we do not need to know the CVS score to compute the software vulnerability level.

8. CONCLUSION

In this paper, we proposed a machine learning based approach to predict software vulnerability level instead of compute CVSS score to be able to get the same vulnerability level. Our method is very simple to use compare to the complexity of CVSS Score computing because it is compose only by a set of logical rules.
The model obtained have a very good prediction performance and can be use to evaluate quickly software vulnerability in industry and in research.

REFERENCES

[1] W. K. Syed Shariyar Murtaza, «Mining Trends and Patterns of Software Vulnerabilities, » The Journal of Systems & Software, 28 February 2016

[2] FIRST.org, "Forum of Incident Response and Security Teams,"Available at: https://www.first.org.

[3] V. N. F. Siv Hilde Houmb a, «Quantifying security risk level from CVSS estimates of frequency and impact, » The Journal of Systems and Software, August 2009.

[4] J. T. John Chambers, «Available from: http://www.first.org/cvss/v1/guide.html., » oct 2014.

[5] K. S. S. R. Peter Mell, «A complete guide to the system version 2.0, http://www.first.org/cvss/cvss-guide.pdf» June 2007.

[6] CVE Details, "The Ultimate Security Vulnerability Datasource", Available at : https://www.cvedetails.com

[7] National Institute of Standards and Technology, "National Vulnerability Database," Available at: http://nvd.nist.gov/statistics.cfm

[8] SANS Institute, "SANS Information Security Training," Available at: http://www.sans.org .

[9] Secunia Research Community, Available at: https://secuniaresearch.flexerasoftware.com/advisories

[10] X-Force, X-Force frequently asked questions. Available at: http://935.ibm.com/services/us/iss/xforce/faqs.html

[11] Vupen Security, Available at: http://www.vupen.com

[12] P. Mell, K. Scarfone et S. Romanosky, «Common Vulnerability Scoring System, » IEEE, pp. 85-89, Nov.-Dec. 2006.

[13] I. Witten, E. Frank. (1999). Data Mining: Pratical Machine Learning Tools and Techniques. Morgan Kaufmann.

[14] Jones O, Maillardet R, Robinson A. "Introduction to Scientific Programming and Simulation Using R". Chapman & Hall/CRC, Boca Raton, 2009.

[15] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem,Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2018). caret: Classification and Regression Training.R package version 6.0-80. https://CRAN.R-project.org/package=caret

[16] R. Ihaka and R. Gentleman. "R: A language for data analysis and graphics". Journal of Computational and Graphical Statistics, 5:299–314, 1996.

[17] Torsten Hothorn, Achim Zeileis (2015). partykit: A Modular Toolkit for Recursive Partytioning in R.Journal of Machine Learning Research, 16, 3905-3909. URL http://jmlr.org/papers/v16/hothorn15a.html