information

MDPI

Article

A Web Platform for Integrated Vulnerability Assessment and Cyber Risk Management

Pietro Russo *, Alberto Caponi, Marco Leuti and Giuseppe Bianchi *

Department of Electronic Engineering, University of Rome Tor Vergata, Rome 00133, Italy
* Correspondence: rssptr01@uniroma2.it (P.R.); giuseppe.bianchi@uniroma2.it (G.B.)

Received: 26 June 2019; Accepted: 15 July 2019; Published: 17 July 2019

Abstract: Cyber risk management is a very important problem for every company connected to the internet. Usually, risk management is done considering only Risk Analysis without connecting it with Vulnerability Assessment, using external and expensive tools. In this paper we present CYber Risk Vulnerability Management (CYRVM)—a custom-made software platform devised to simplify and improve automation and continuity in cyber security assessment. CYRVM's main novelties are the combination, in a single and easy-to-use Web-based software platform, of an online Vulnerability Assessment tool within a Risk Analysis framework following the NIST 800-30 Risk Management guidelines and the integration of predictive solutions able to suggest to the user the risk rating and classification.

Keywords: cyber risk management; vulnerability assessment; risk analysis; recommending system; OpenVas; NIST 800-30; web based software platform

1. Introduction

In recent years, with the ever growing reliance on ubiquitous connectivity and Information Technology, cyber security has dramatically increased its importance. Different enterprises and organizations in the public and private sectors manage potentially very diverse Information systems, ranging from office networks to business-specific processes to very specialized systems. With a potentially complex and very specific attack surface and exposure to threats and disruptions [1,2], it is of paramount importance that not only security administrators, but also leaders and managers at all levels, do fully understand cyber security risks. To a greater extent, assessment continuity is nowadays crucial owing to today's ever faster changing threat and vulnerability landscape.

In the light of such a context, in this paper we present a software platform called CYRVM (CYber Risk Vulnerability Management), specifically designed to be used by small and micro Enterprises who do not have the skill for (or cannot afford the cost of) a dedicated IT security team/person. Its development has been done with four main design drivers in mind. Two of them, namely simplicity of usage and partial automation, are quite typical and as such are targeted in most relevant systems, including most commercial systems. What distinguishes CYRVM is (i) the careful integration, within a single framework, of both the online (operational) Vulnerability Assessment phase, as well as the subsequent offline Risk analysis—two phases which are typically handled by different platforms or tools; (ii) the supplementary possibility to integrate a collaborative environment which enables the possibility to predict for every vulnerability how *dangerous* they are for his network, using the information shared by other users. The result is a software:

- with an extremely simple web-based user-interface, implemented as a web application (with no need for local machine installation);
- standard-compliant because it follows the NIST 800-30;

- that is able to provide a ready-to-distribute reports and statistics, along with vulnerability details for all the assets, giving to the user a complete view of the cyber health status of system.

In summary, CYRVM's main novelties are:

- the combination of Vulnerability Assessment and Risk Analysis: the former is based on OpenVas (an open source vulnerability scanning and management solution), while the latter is custom-programmed;
- the capability to predict values of Impact and Likelihood for every vulnerability, using 3 different algorithms (Average, Matrix Factorization and a Custom Algorithm) which take advantage of the aggregate information about what other users have done—note that CYRVM's user interface provides such predictions as "suggestions" which end users may eventually override.

The rest of the paper is organized as follows. We start with the analysis of the state of the art followed by a chapter where the description of how the software platform is able to do Risk Analysis combined with Vulnerability Assessment is presented. This chapter is followed by one in which a practical use case is described. Then we present the implemented prediction algorithms and finally the validation results.

2. State of the Art

2.1. Risk Management

Cyber risk can be defined as a function of 3 parameters: (i) Impact expresses the level of damage that a given risk may cause; (ii) Threat expresses whether or not a given risk can occur; (iii) Vulnerability expresses whether or not existing information security measures are effective [3,4]. New technological advances in every sectors, such as cloud manufacturing [5], industry 4.0 [6] and realtime service composition [7] has opened a new horizon into the cyber world with a new type of risk that was unknown beforehand.

In general, risk management is a crucial factor in the success of an organization. Several different groups are concerned with risk management in an organization in order to meet many business or regulatory requirements [8]. The management of this risk should give a general view of the organization through the cataloguing of resources, the identification of threats and the valuing of vulnerabilities [9]. In a Cyber scenario, Risk Management should be done on very different and complex systems with the aim of identifying the risk, then validate it and, lastly, consider responses to the risk [10].

There are many approaches to Cyber Risk Management. ISO 31000:2018 [11], an international standard issued in 2009 by ISO (International Organization for Standardization), is intended to serve as a guide for the design, implementation and maintenance of risk management. Whereas the National Institute of Standards and Technology (NIST) [12] developed a cyber security framework according to which risk management can be divided into two different phases. The first one is Vulnerability Assessment, the process through which is possible identify, quantify and prioritize the system vulnerabilities. The second one is Risk Analysis, the process of identifying and analyzing potential elements that could negatively impact key business initiatives or critical projects in order to help organizations avoid or mitigate those risks. Usually traditional way of doing Risk Management focuses only on this second phase without considering the Vulnerability Assessment one [13]. In addition to this, it is done in different places by different organizations in different moments or internally using additional software or contractors with the consequential increase of costs.

The 2 phases described above should be followed by security testing. In fact, security risk assessment and security testing both contribute to an overall assessment of the security of a system on different levels [14]. Several approaches can be used (e.g., ISO/IEC/IEEE 29119) combining risk assessment and testing focus on specific aspects of the risk or testing process such as test case generation or risk estimation [15,16].

This is our first challenge: can we help the network manager in this hard work connecting vulnerability assessment and risk analysis without increasing costs? Can we do this work using only one software, without buying different tools in different moments of the *work flow*? Our application is different from previous works because it permits with a unique easy to use software to implement a standard framework, combining vulnerability assessment and risk analysis phases, without using different software or external (and typically expensive) instruments.

2.2. Prediction

It is not sufficient to do this work without helping the network administrator in the quantification of the impact of a particular negative event on the organization. In fact, especially in a context of small/medium enterprise, the network manager does not know exactly the importance of every asset, how dangerous is their lost, which type of cyber attack can affect them. For this reason he is not able to chose correct values of impact and likelihood for every vulnerability of every assets. In the last years there has been an increase of machine learning techniques use not only into the traditional sectors of risk analysis [17,18] but also into the cyber risk management [19,20] with the aim of predicting risk and avoiding cybercrime in future [21]. Our work goes in this direction: trying to predict cyber risk through the use of custom algorithms and Matrix factorization [22–24]. This is our second big challenge: can we *predict* and suggest to a network manager the right risk value for every assets and vulnerabilities?

In order to solve this problem, considering the choices of other users with similar systems, also through the use of new predictive algorithm, a solution is presented to the user. In the literature there are many examples of collaborative systems [25–29]. Our choice is different because in the software platform we implement algorithms able (i) to predict the user choices of risk values starting from what other users did before, (ii) understanding which users are more similar to the new one. For every vulnerability, the user is able to consider a rate (from very low to very high) in terms of impact and likelihood. In this way, the user can decide to update some values, accept all values because he *completely agrees* or accept all values because he *trusts* the other users' choices. These updated choices will be saved into the database with different weights (1 if the user *agrees*, 0 if the user *trusts*) and used in the other calculation.

2.3. Shared Information

All the above algorithms and procedures work well if there is information sharing between users. In fact, only if everyone shares vulnerabilities of the system can a good recommendation be made. Actually the proposed software platform is based on the concept of a safe third part [30,31] where everyone can save information. In the future we plan to remove this assumption creating a privacy preserving mechanism able to guarantee the confidentiality of the shared information.

3. Cyber Security Management in CYRVM

We create a software platform called Cyber Risk Vulnerabiity Management (CYRVM) that can be used for the cyber risk management using the standard NIST 800-30. We decided to use this framework because we think it analyzes cyber risk starting from a higher level going down in detail without needing any previous IT technical know-how. The most important features of this software are:

- It is easily reached by every browser web and It does not need any other installation on your machine;
- It is standard because it follows the NIST 800-30;
- It is able to integrate Vulnerability Assessment through the import of OpenVas (a free framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution) report;
- It lists all the vulnerabilities for all the assets giving to the user a complete view in terms of integrity and confidentiality;
- It calculates the probability of an event and an estimation of the impact on the system;

3.1. Implemented Methodology

Following the procedures described into the standard NIST 800-30, this software platform can help the network administrator in the *risk analysis*. These steps are (Figure 1):

1. Characterization of the system. In this step the boundaries of the system are identified, along with the resources and the information that constitutes it.
2. Threats Identification. The goal of this step is to identify the potential threat-sources and compile a threat statement listing potential threat-sources that are applicable to the system being evaluated.
3. Vulnerabilities identification. The objective of this step is to derive a list of the system vulnerabilities (observations) that could be exercised by the potential threat-sources.
4. Controls analysis. The goal of this step is to analyze the controls that have been implemented, or are planned for implementation, by the organization to minimize or eliminate the likelihood (or probability) of a threat's exercising a system vulnerability;
5. Events Likelihood Analysis. The objective of this step is to derive a likelihood rating score for each vulnerability Threat-source motivation and capability, nature of the vulnerability, current controls applied to the vulnerability;
6. Events Impact Analysis. The next major step in measuring level of risk is to determine the adverse impact resulting from a successful threat exercise of a vulnerability after obtaining information such as System mission (e.g., the processes performed), System and data dangerousness (e.g., the system's value or importance to an organization), System and data sensitivity.
7. Risk determination. The purpose of this step is to assess the level of risk to the system.
8. Recommended Controls. During this step of the process, controls that could mitigate or eliminate the identified risks, as appropriate to the organization's operations, are provided.
9. Final documentation. Once the risk assessment has been completed (threat-sources and vulnerabilities identified, risks assessed, and recommended controls provided), the results should be documented in an official report or briefing.

This step can be divided into two parts. The first is the vulnerability assessment (from step 1 to step 3) where it is analyzed where the system can be attached. The second (from step 4 to step 9) is the risk analysis phase where every vulnerability of every asset is calculated to establish what the real risk for the system is and how it is possible to mitigate it.

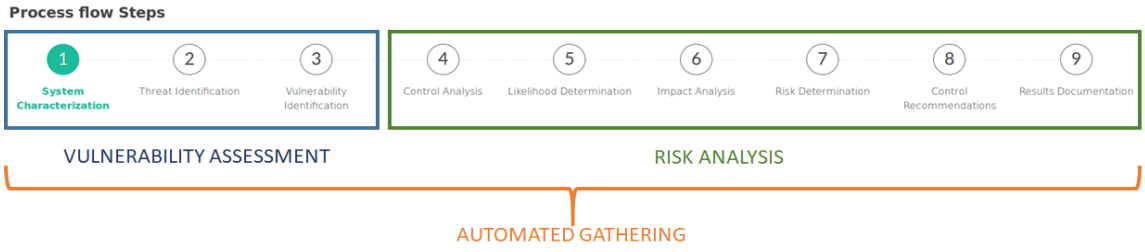

Figure 1. How CYRVM implements NIST 800-30.

3.2. Software Architecture

CYRVM from the *front end* side is written in HTML and Javascript, whereas for the *back end* PHP has been chosen. The web application is built in order to completely automate the steps described above and for this reason we can divide its software structure into 9 parts, each one composed of php files, connected between themselves. Each group needs some input and produced some outputs that can be used by one of the following parts. Figure 2 presents these groups.

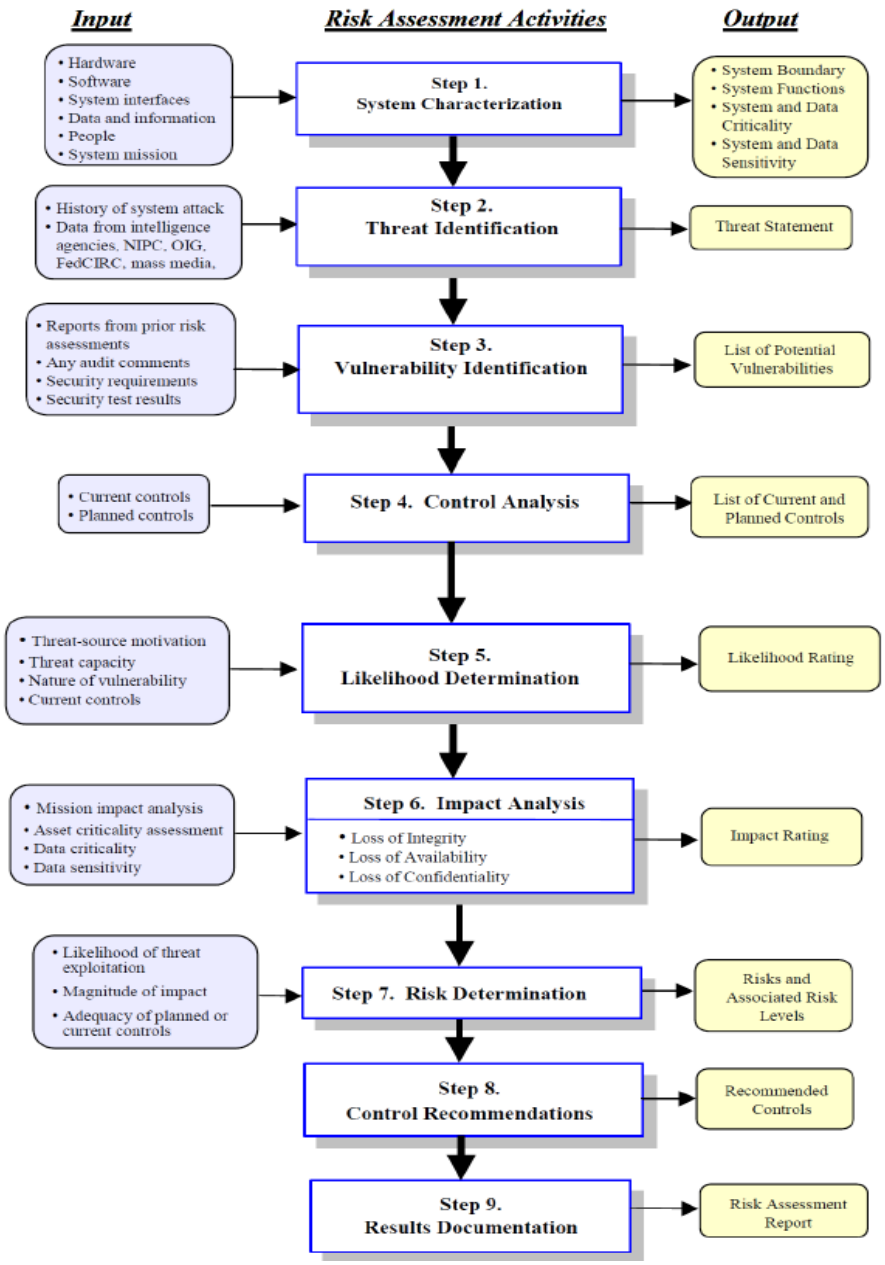

Figure 2. How CYRVM (CYber Risk Vulnerability Management) implements NIST 800-30.

Figure 3 presents a general view of the architecture of the application. Using a simple web browser, the user can connect to the web server where CYRVM is host. This web server uses a network manager module in order to manage all the information provided by user about network (assets, environment, threats, etc.) and it is also directly connected to the DB in order to save vulnerability scan provided by user. NIST 800-30 module provide to the web server all the information needed in order to implement the framework steps (described in Figure 2. Finally the prediction module is used to predict Impact and Likelihood values provided to the user.

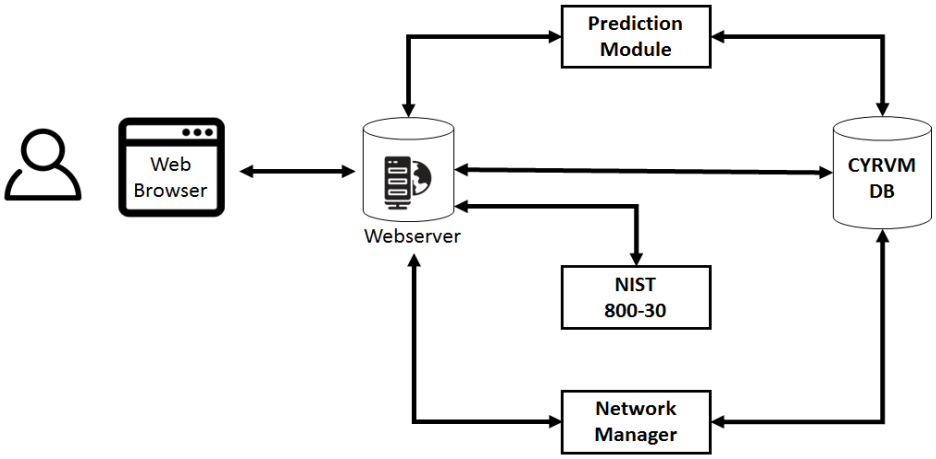

Figure 3. CYRVM architecture.

3.3. Database Structure

We save information into the DB using a very simple principle: for every system (defined by a unique name) every vulnerability has a value greater than 0 if that particular vulnerability is one of that system. The tables can be divided into 3 groups:

- the first group is composed of values given by OpenVas (or similar software) that is able to give for every vulnerability its severity and Quality of Detection (in Figure 4). These values for the vulnerability have a range between 0 and 10 (where 0 is the lowest and 10 is the highest), whereas the Quality of Detection can have a range from 0% to 99.9%.

SYSTEM	SLOT	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
router1	1	5	0	0	0	0	0	0	0	0	0
hotspot router2	2	0	2,6	0	0	0	0	0	0	0	0
router linkem	3	0	0	0	0	0	0	0	4	0	0
local net	4	0	0	0	0	0	0	5	0	0	0
ADSL net1	5	0	2,6	10	9,3	5	0	4,2	0	0	0
ADSL net2	6	0	2,6	10	9,3	5	5	0	0	0	0
ubuntu server	7	0	0	0	0	0	0	0	0	1,2	1,5
Fedora	8	0	2,6	0	0	0	0	0	0	1,5	0

Figure 4. Severities of vulnerabilities of systems saved into the DB.

- the second group is composed of values calculated by the software platform for Impact and Likelihood (in Figure 5). These values can be *(null)* if the particular vulnerability is not present in a particular system, or 1, 2, 3, 4 or 5 that mean Very Low, Low, Medium, High, Very High.

SYSTEM	SLOT	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
router1	1	5	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
hotspot router 2	2	(null)	4	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
router linkem	3	(null)	(null)	(null)	(null)	(null)	(null)	(null)	4	(null)	(null)
local net	4	(null)	(null)	(null)	(null)	(null)	(null)	5	(null)	(null)	(null)
ADSL net1	5	(null)	5	2	5	3	(null)	1	(null)	(null)	(null)
ADSL net2	6	(null)	2	1	5	4	4	(null)	(null)	(null)	(null)
ubuntu server	7	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	4	1
Fedora	8	(null)	4	(null)	(null)	(null)	(null)	(null)	(null)	5	(null)

Figure 5. Impact of vulnerabilities of systems calculated by CYRVM.

- the third group is composed of *weights*. In fact, considering choices of the user, every Impact and Likelihood value is saved with a different *importance* (in Figure 6). Values can be 0 or 1 if the weight is minimum or maximum. In the following chapters we will better analyze this point.

SYSTEM	SLOT	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
router1	1	1	0	0	0	0	0	0	0	0	0
hotspot router2	2	0	0	0	0	0	0	0	0	0	0
router linkem	3	0	0	0	0	0	0	0	1	0	0
local net	4	0	0	0	0	0	0	1	0	0	0
ADSL net1	5	0	1	1	1	1	0	1	0	0	0
ADSL net2	6	0	0	0	0	0	0	0	0	0	0
ubuntu server	7	0	0	0	0	0	0	0	0	1	0
Fedora	8	0	1	0	0	0	0	0	0	1	0

Figure 6. Weights of vulnerabilities of systems saved into the DB.

In CYRVM, there is a strong management of vulnerability because of the table structure. For this reason, if a system with a new vulnerability is saved into the DB every table is updated with a new column and, vice versa, if there are not any systems with a particular vulnerability, that column is deleted. In this way It is possible to guarantee the consistency and the information saved in the DB.

3.4. Impact and Likelihood Collaborative Prediction

The risk analysis process we have presented is able to give good results only if It is based on good values of Impact and Likelihood. In fact, using software like Openvas, it is possible to find what are the vulnerabilities of your network but It is not easy to understand how *dangerous* are these vulnerabilities for that. The main problem is giving a right value to Impact and Likelihood in step 5 and 6 of Figure 2, usually assigned by users without a real knowledge of the network. The main novelty of CYRVM is the capability of predicting values of Impact and Likelihood to the user using information of other users saved into the DB.

CYRVM is able to propose a prediction to users using 3 different algorithms, dynamically chosen by the user: arithmetic average, Matrix Factorization, custom algorithm. The first 2 algorithm are known algorithms typically used in the literature in order to solve these kind of problems. The third one is a new type of algorithm specifically designed for this type of context.

3.4.1. Average Algorithm

The easiest way to calculate a value starting from existing ones is through the arithmetic mean that can be defined as the sum of the n numerical values of each observation divided by the total number of observations.

$$A = \frac{1}{n} \sum_{i=1}^{n} a_i$$

Using this simple principle the Algorithm 1 is implemented in CYRVM:
The following functions are used in the Algorithm 1:

- GetRecord is used in order to save all the vulnerabilities with their severity and Quality of Detection starting from the Openvas report.
- GiveSys and GiveVuln are used in order to calculate the number of systems and of vulnerabilities of the DB.
- InsertRecord is used in order to insert the record taken by the Openvas report into the DB. *Insert* a new record means not only create a new record in each table but adding new columns in case of new vulnerabilities.

- averageImp and averageLike are the core functions of the algorithm because are able to calculate the means of Impact and Likelihood values loaded into the DB. In case of first occurrence of a particular vulnerability (so in case of mean equal to 0) the value presented to the user is 3 (Medium).

Algorithm 1 Average algorithm.

```
ImpactNew: array of int;

LikelihoodNew: array of int;

RecordNew = GetRecord("../report.xml");

NumberSys=GiveSys();

everyVuln=GiveVuln();

InsertRecord(RecordNew);

i=0;

for everyVuln do

    ImpactNew[i] = averageImp(NumberSys);

    LikelihoodNew[i] = averageLike(NumberSys);

    i+1;

end for

return = ImpactNew, LikelihoodNew;
```

3.4.2. Matrix Factorization Algorithm

Recommender systems are based on one of two strategies [32]. The first one is called *content filtering approach* and It is based on the construction of a profile for each user or product to characterize its nature (of course It requires gathering external information that might not be available or easy to collect). An alternative methods relies only on past user behaviour (for example, previous transactions or product ratings) without requiring the creation of explicit profiles. This approach is known as *collaborative filtering*. The two primary areas of collaborative filtering are the neighbourhood methods and latent factor models. Matrix Factorization algorithm is the most successful realizations of latent factor model and has been applied on the Netflix problem with better results of other collaborative filtering methods. The strength of matrix factorization is that it allows incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion by observing user behaviour including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modelled as inner products in that space. Accordingly, each item i is associated with a vector $q_i \in R^f$ and each user u is associated with a vector $p_u \in R^f$. For a given item i, the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $q_i^T p_u$ captures the interaction between user u and item i. This approximates the rating of user u for the item i, which is denoted by r_{ui}, leading to the estimate:

$$\hat{r_{ui}} = q_i^T p_u$$

The best way in order to calculate factor vectors (p_u and q_i), the system minimizes the regularized squared error on the set of known ratings:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + (||p_u||^2)$$

Matrix Factorization is very used into Recommending Systems algorithms [33,34] in order to predict a certain rank that a user could give to a particular item. Starting from the idea expressed above, and considering good results given by this type of approach [35] we decide to implement this method like in the Algorithm 2.

Algorithm 2 Matrix Factorization algorithm.

```
1:  ImpactNew: array of int;
2:
3:  LikelihoodNew: array of int;
4:
5:  RecordNew = GetRecord("../report.xml");
6:
7:  NumberSys=GiveSys();
8:
9:  InsertRecordMF(RecordNew);
10:
11: ImpactNew=MatrixFactImpact();
12:
13: LikelihoodNew=MatrixFactLikelihood();
14:
15: return = ImpactNew, LikelihoodNew;
```

The first part of this algorithm is the same of Algorithm 1 because also in this case is important the management of new systems with new vulnerabilities. The different core functions used in this case are:

- InsertRecordMF that is very similar to InsertRecord of the Algorithm 1 because It is able to add information about new vulnerabilities. In addition to this It is able to add a record of 0 in the two tables Impact and Likelihood;
- MatrixFactImpact and MatrixFactLikelihood are the functions that implement the Matrix Factorization algorithm. Through this algorithm the 0 values, inserted in a new record into the Impact and Likelihood tables by InsertRecordMF, are changed considering all the other user choices. These values are normalized and presented to the user.

3.4.3. Custom Algorithm

We call the third implemented algorithm *custom algorithm* because It can not be referred to any other *fixed* algorithm but It uses the same *philosophy* of the above algorithms. Before expressing this algorithm in a formal way It can be useful understand its general steps:

1. find in the DB the *nearest* system (in term of number of common vulnerabilities and severity and Quality of Detection) to the new one;
2. calculate, for every vulnerability, differences between these two systems (we call them *slot*);
3. take the values of impact and likelihood of the common vulnerabilities and apply the calculating slot;
4. if new vulnerabilities are present calculate with matrix factorization the values of Impact and likelihood.

Starting from the above idea, we can define the Algorithm 3.

Algorithm 3 Custom algorithm.

```
Slot: array of int;

ImpactNew: array of int;

LikelihoodNew: array of int;

RecordNew = GetRecord("../report.xml");

NumberSys=GiveSys();

everyVuln=GiveVuln();

InsertRecord(RecordNew);

RecordWinner = FindWinner(RecordNew);

RecordWinnerImpact = GetImpact(RecordNew);

RecordWinnerLikelihood = GetLikelihood(RecordNew);

TableRecordVuln = CreateTableVuln(RecordWinner, RecordNew);

TableRecordQoD = CreateTableQoD(RecordWinner, RecordNew);

TableRecordMatrixVuln = MatrixFactorizationC(TableRecordVuln);

TableRecordMatrixQoD = MatrixFactorizationC(TableRecordQoD);

i=0;

for everyVuln do

    Slot[i] = slotCalculation(TableRecordMatrixVuln, TableRecordMatrixQoD);

    SlotApplication(ImpactNew[i], LikelihoodNew[i], RecordWinnerLikelihood,
RecordWinnerImpact, Slot[i]);

    i+1;

end for

if differentVuln() then

      MatrixFactImp(ImpactNew, RecordWinnerImpact);

      MatrixFactLike(LikelihoodNew, RecordWinnerLikelihood);

end if

return = ImpactNew, LikelihoodNew;
```

The main functions used in the Algorithm 3 (different from the previous ones) are:

- RecordWinnner. This function is used in order to find in the DB the nearest system (in terms of severity and Quality of Detection of vulnerabilities) to the new one. The idea in this case is: can I use the information of another system that is *very similar* to the new one? For this reason, for each vulnerabilities, this function does an algebraic sum of values of severity and QoD finding the system that have the minimum difference.
- GetImpact and GetLikelihood find the corresponding values of Impact and Likelihood for the winner system.
- CreateTableVuln and CreateTableQod. These two functions create two particular tables composed by two rows (the first one for the Winner system, the second one for the new system) and by N columns, where N is the number of Vulnerabilities into the DB. In the first table are inserted the values of the severities of the systems whereas in the second one the Qod values. In addition to this a 0 is placed when there is an unknown value.
- MatrixFactorizationC is a function that applied the Matrix factorization to the tables produced by the previous functions.
- slotCalculation. It is the core function of the algorithm because It is able to calculate a value (here called *slot*) that represents how different is that vulnerability between the two systems. In fact the idea is to take the common vulnerabilities between the new system and the *WINNER* system and, if It is needed, increment or decrease their impact/likelihood values basing on the few differences of the systems.
- SlotApplication applies the calculated slots to the impact and likelihood values of the *WINNER* system.

- MatrixFactImp and MatrixFactLike are used for the calculation of impact and likelihood values for those vulnerabilities that the new system has and are not present in the WINNER one.

4. Validation and Results

The following chapter is organized as follows. Firstly we compare our three algorithms. Then we present how we conduct the validation process and what the obtained results were.

4.1. Comparison of the Algorithms

When a user decide to upload the vulnerability scan of its network can decide to obtain the values of impact and likelihood through the use of one of algorithms analyzed above. These 3 algorithms are very different because they have a very different *way of calculation* of values. In fact:

- *Average*. It is the simplest way to solve the problem because, considering a particular vulnerability, it gives an *idea* of the value of Impact and Likelihood for a new system starting from the saved data. But It is not able to consider the difference between two or more systems, or the case of a system with atypical values (this is the limit of the arithmetic mean).
- *Matrix Factorization*. This is the best mathematical method considering a *collaborative filtering approach* for the resolution of this type of problem. It considers all values of every systems saved into the DB, considering also *strange* values. The only problem is that It gives to the system (or systems) nearest to the new one the same importance of the other.
- *Custom Algorithm*. It can consider a mix of the two others because it finds the system similar to the new one and calculates the Impact and Likelihood values starting from the nearest system one. The only problem is the calculation of Impact and Likelihood values in case of new vulnerability. In fact in that case Matrix Factorization is used (considering only two systems, the new one and the *WINNER* one). But from a mathematical point of view Matrix Factorization works well if a very large number of data are used and not only two.

4.2. Validation Execution

The proposed algorithms have been assessed using a leave-one-out method. According to this method, one by one, every instance of the database is used like test set whereas the rest of the database like training set. Particularly, the test set is composed of 4 type of information: the severity of every vulnerability of the system, their quality of detection, the chosen impact and likelihood. Using the first two information, like happens for a new system, the chosen algorithm calculates a solution (value of impact and likelihood) using the rest of the DB (training set). This solution is compared with impact and likelihood values containing into the test set in order to calculate the error percentage. This percentage has been calculated using the following formula:

$$P = \frac{\sum_{j=1}^{N_s} \sum_{i=1}^{N_v} \left| T_{ji} - C_{ji} \right|}{N_v N_s 4} \tag{1}$$

where:

- N_v is the number of vulnerabilities;
- N_s is the number of systems;
- T_{ji} is the value of Impact (or Likelihood) for vulnerability i of the system j of the Test Set;
- C_{ji} is calculated value of Impact or Likelihood for vulnerability i of the system j;
- 4 is the maximum deviation from the calculated value and the Test Set one;

The DB is composed of 50 systems and 13 different types of vulnerabilities. The 50 records can be divided in these 3 groups:

1. **Real information.** 8 records are composed by 8 real systems with real Vulnerabilities. These networks are:

    - a network composed by a CISCO router;
    - a network composed by 2 mobile phones with Android 5 connected to another hot-spot mobile phone;
    - a network composed by 4 PCs with Windows 10 64bit connected to a LINKEM router;
    - a Local Area Network composed by 2 PCs with Ubuntu 14.10;
    - an ADSL network composed by a PC with Windows 10 64bit, 2PCs with Windows 7 32bit, 2 mobile phone with Android 5 and 6, a router;
    - an ADSL network composed by 4 PCs with Windows 10 64bit, 4 PCs with Windows 7 32bit, a CISCO router;
    - a network composed by a router and a PC with Ubuntu Server and 2 PCs with Windows 10 64bit;
    - a network composed by a router and a PC with Fedora and 1 PC with Windows 10 64bit;

2. **Random information.** 20 records with vulnerabilities completely randomly generated;
3. **Structured information.** 22 records *similar* to the first 8 ones. In fact, these systems have some vulnerabilities depending by factors such as Operative Systems non updated or old version of firmware. For this reason, we can reasonably suppose that some other networks can have the some problems or a part of those.

In order to test in the best way algorithms working, our intention was to create a database with very heterogeneous networks composed by different types of assets (mobile devices, servers, personal computers, routers) each of them with different Operative Systems. The consequence of this choice was the detection of different types of vulnerabilities (showed in Figure 7) in different fields (internet, drivers, operative system, etc.). In this way the created scenario is very similar to a real one.

v2	TCP timestamps
v3	OS End Of Life Detection
v4	Microsoft Windows SMB Server Multiple Vulnerabilities
v5	DCE/RPC and MSRPC Services Enumeration Reporting
v6	Acme thttpd and mini_httpd Terminal Escape Sequence in Logs Command Injection Vulnerability
v7	SSL/TLS: Report Vulnerable Cipher Suites for HTTPS
v8	SSL/TLS: Certificate Signed Using A Weak Signature Algorithm
v9	ICMP Timestamp Detection
v10	OS Detection Consolidation and Reporting
v11	SSL/TLS: Report Weak Cipher Suites
v12	SSL/TLS: Deprecated SSLv2 and SSLv3 Protocol Detection
V13	SSL/TLS: SSLv3 Protocol CBC Cipher Suites Information Disclosure Vulnerability (POODLE)
V14	SSL/TLS: Diffie-Hellman Key Exchange Insufficient DH Group Strength Vulnerabil

Figure 7. Different types of vulnerabilities used to test algorithms.

4.3. Results

Figure 8 reports error percentages for implemented algorithms. How It is possible to see, the algorithms work very well with error percentages in the best case around 8% and in the worst case around 18%. This means that, every 10 cases, only 1-2 predictions of impact and likelihood values are wrong.

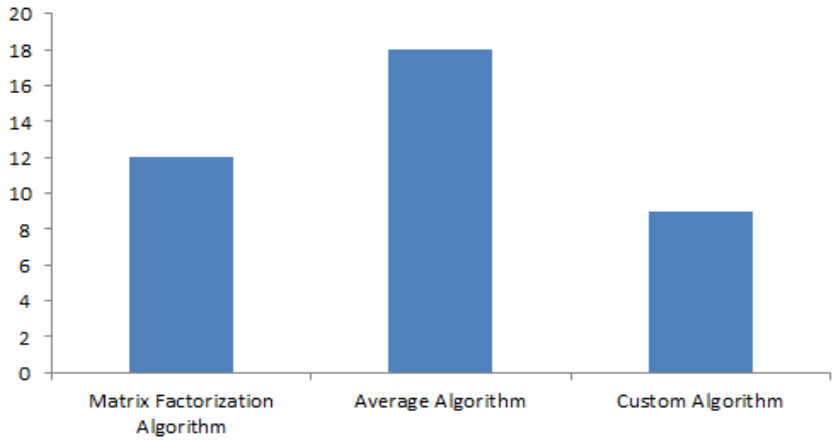

Figure 8. Error Percentages for the implemented Algorithm.

The trend showed in Figure 8 reflects what we expect. The worst algorithm is the Average because it does not consider similarities between systems considering all the new ones in the same way. The best ones are the Matrix Factorization (that is the *state of the art* in the world of the recommending algorithms) and the Custom Algorithm. This is because our problem can not be consider like a *pure* recommending problem because there is not a fixed probabilistic correlation between every data in the database.

All these considered, the algorithms seems to be very strong with errors under 20%. But they should be tested with a larger number of systems. In fact, although the 50 systems inserted into the DB create a very realistic scenario, they are not enough. A real test should be with a database with hundreds of different real networks with a very large number of vulnerabilities. In future we plan to insert other real networks into the DB in order to test in a better way the implemented algorithms.

5. Interaction with the User

When Impact and Likelihood values are calculated and presented to the user the next important step is to *capture* the user preferences. In fact, when a solution is presented, the software platform gives to the user the possibility to change them and It is able to save this changes in order to use in the other computations. Behind this capability there is a general idea: an user can decide to accept passively the prediction given by the software (without any changes) or he can change it. These two choices hide two different behaviour. In the first case the user does not know enough about that vulnerability or simply does not want to change anything. In the second case we have an *expert* user who knows exactly what he is doing. For this reason, the system should consider in a different way these two choices for the future calculation.

All these considered, when a prediction is presented to an user he can follow one of the following steps:

- *Accept the prediction.* In this case the user is completely agree with every values predicted by the software and he does not want to change anything.
- *Follow the others.* In this case the user does not care about the choices because he *trusts* in the software (because he does not have the know-out for example). This case seems to be identical to the previous one (in the Impact and Likelihood table the same values will be saved) but, how It can be easily understood It is very different because in this case values are accepted by the user who accepts passively the calculated values.
- *Modify your values.* In this case the user decides to change just some values, typically where he is more confident, following the software prediction for the other values.

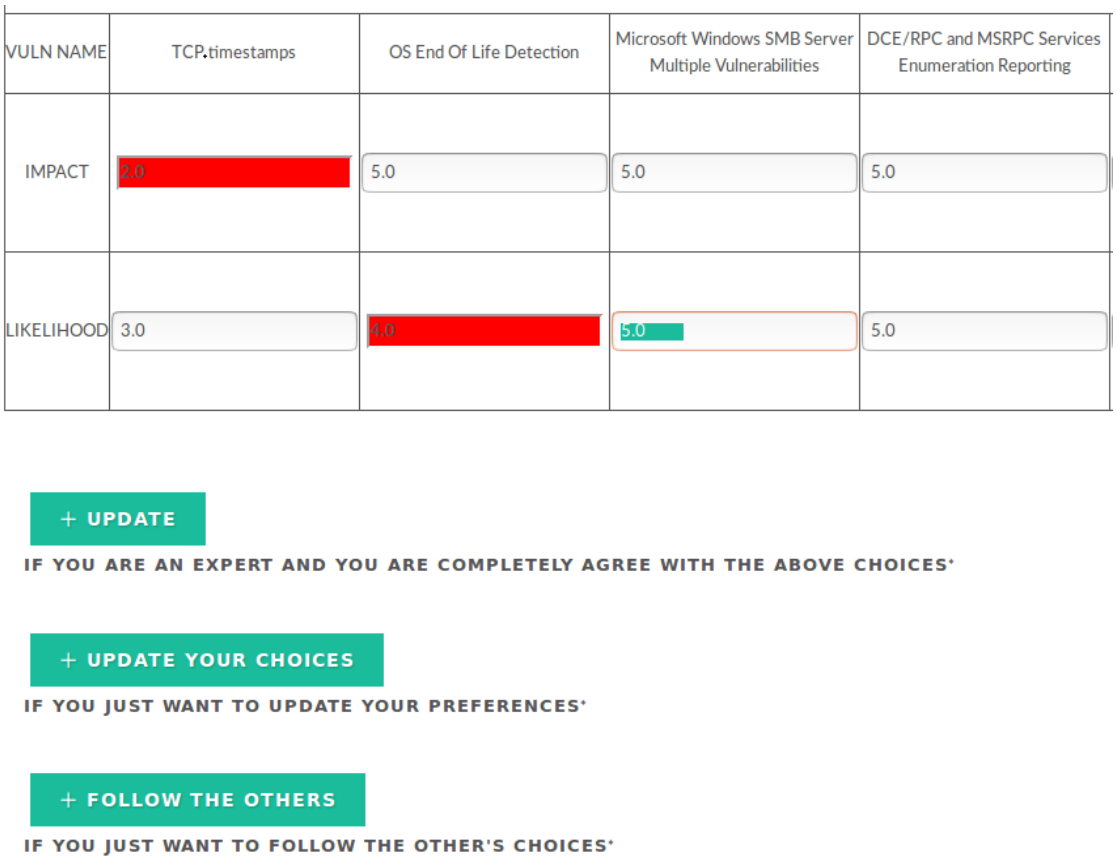

Figure 9. User choices.

From a graphical point of view, to the user is presented a table like in Figure 9 and he can change values into the cells of the table (when a value is changed the cell becomes red) and at the end he can decide which type of update he would like to do: an acceptance of every values, an acceptance of just the updated values, the passively acceptance of every values.

But, what are the effects of the user choices? Like we said above there is a different importance between the user choices. This means that every choice has a different weight and this weight has a different effect on the algorithms implemented by the software platform. Particularly we decide to give a weight of 0 when the user *does not have an opinion*, 1 when the user *agrees* with the software prediction. This type of data are saved into a table for the Impact and another one for the likelihood (like in Figure 6).

6. Conclusions and Future Works

In this paper we presented CYRVM, a software platform designed to do cyber risk management for small and micro Enterprises who do not have the skill for (or cannot afford the cost of) a dedicated IT security team/person. This software is able to help the network administrator during all risk management phases, following the procedures described into the standard NIST 800-30, combining Vulnerability Assessment with Risk Analysis. In addition to this, CYRVM gives to the user a prediction of Vulnerability Impact and Likelihood values basing on the history of the other systems saved into the DB. The choice of these values is the main weakness in the Risk Management process because network manager could not have all the needed know-out to understand how dangerous is a certain vulnerability of the network.

Actually the main limitation of CYRVM is the availability of real information derived from real network, used by the algorithms to predict Impact and Likelihood values. In the future we plan to extend the present DB of CYRVM with other vulnerability scans from different real networks

(Universities, Public Administration Networks, etc.) in other to increment the know-out used for the prediction.

Our second main goal for the future is the implementation of a mechanism able to ensure the confidentiality of the shared information given by the user. In fact, one of the most important element of CYRVM is the DB with the information given by the users who, obviously, could decide to preserve them simply because they do not want to reveal the vulnerabilities of their networks. In literature there are some mechanism used by collaborative systems in order to preserve privacy [36–39] using different methods such as non negative Matrix Factorization [40]. Our challenge is implementing inside the software platform secret sharing mechanism able to protect the information given by different users.

Author Contributions: Investigation, P.R. and A.C.; Project administration, G.B.; Resources, A.C. and M.L.; Software, P.R. and M.L.; Supervision, G.B.; Validation, P.R. and A.C.; Writing—original draft, P.R.; Writing—review & editing, P.R.

Funding: This project has received funding within the BPR4GDPR project from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787149.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baldoni, R.; Querzoni, L.; Montanari, L. Italian Cybersecurity Report—Controlli Essenziali di Cybersecurity. In Proceedings of the CIS Sapienza e Laboratorio Nazionale CINI, Piazzale Aldo Moro, Italy, 2 March 2017.
2. Jenkins, B.D. *Security Risk Analysis and Management—Risk Analysis Helps Establish a Good Security Posture; Risk Management Keeps it that Way*; Countermeasures Inc.: Hollywood, CA, USA, 1998.
3. Biener, C.; Eling, M.; Wirfs, J.H. Insurability of Cyber Risk: An Empirical Analysis. *Geneva Pap. Risk Insur.* **2015**, *40*, 131–158. [CrossRef]
4. Grange, J.S.; Schields, T.; Vandenberg, T.; Zeichner, L. *BITS Technology Risk Transfer Gap Analysis Tool*; BITS Financial Services Roundtable: Washington, DC, USA, 2002.
5. Krishnaiyer, K.; Chen, F.F.; Bouzary, H. Cloud Kanban Framework for Service Operations Management. *Proced. Manuf.* **2018**, *17*, 531–538. [CrossRef]
6. Hofmann, E.; Rüsch, M. Industry 4.0 and the current status as well as future prospects on logistics. *Comput. Ind.* **2017**, *89*, 23–34. [CrossRef]
7. Bouzary, H.; Chen, F.F. Service optimal selection and composition in cloud manufacturing: A comprehensive survey. *Int. J. Adv. Manuf. Technol.* **2018**, *97*, 795–808. [CrossRef]
8. Luděk, N.; Petr, D.; Lea, N. Efficient Cyber Risk Management, Auditors Experience. In Proceedings of the International Conference on Organizational Science Development, Organizationa and Uncertainity in the Digital Era, Portorož, Slovenia, 21–23 March 2018.
9. Crovini, C.; Ossola, G.; Marchini, P.L. Cyber Risk. The New Enemy for Risk Management in the Age of Globalisation. *Manag. Control* **2018**, *2*, 135–155. [CrossRef]
10. Zarreh, A.; Wan, H.D.; Lee, Y.; Saygin, Can.; Al Janahi, R. Risk Assessment for Cyber Security of Manufacturing Systems: A Game Theory Approach. In Proceedings of the 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2019), Limerick, Ireland, 24–28 June 2019.
11. International Organization for Standardization (ISO). *Risk Management Guidelines*; ISO 31000:2018(E); ISO: Geneva, Switzerland, 2018.
12. Feringa, A.; Goguen, A.; Stoneburner, G. *Risk Management Guide for Information Technology Systems. NIST Special Publication 800 30*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2002.
13. FERMA. *Standard di Risk Management*; Federation of European Risk Management Associations: Brussels, Belgium, 2003.
14. Groffmann, J.; Seehusen, F. Combining Security Risk Assessment and Security Testing Based on Standards. In Proceedings of the 3rd International Workshop on Risk Assessment and Risk-Driven Testing, RISK 2015, Berlin, Germany, 15 June 2015; Lecture Notes in Computer Science; Springer: Cham, Switzerland; Volume 9488, pp. 18–33.

15. Erdogan, G.; Li, Y.; Runde, R.; Seehusen, F.; Stolen, K. Approaches for the combined use of risk analysis and testing: A systematic literature review. *Int. J. Softw. Tools Technol. Transf.* **2014**, *16*, 627–642. [CrossRef]
16. Felderer, M.; Schieferdecker, I. A taxonomy of risk-based testing. *Int. J. Softw. Tools Technol. Transf.* **2014**, *16*, 559–568. [CrossRef]
17. Biffis, E.; Chavez, E. Satellite data and machine learning for weather risk management and food security. *Risk Anal.* **2017**, *37*, 1508–1521. [CrossRef] [PubMed]
18. Makov, U.; Weiss, J. Predictive modelling for usage-based auto insurance. In *Predictive Modeling Applications in Actuarial Science*; International Series on Actuarial Science; Cambridge University Press: Cambridge, UK, 2016; pp. 290–308.
19. Gareth, W.P. Statistical Machine Learning and Data Analytic Methods for Risk and Insurance. Version 8, 2017. Available online: https://ssrn.com/abstract=3050592 (accessed on 15 July 2019).
20. Peters, G.W.; Cohen, R.; Maurice, D.; Shevchenko, P. Understanding Cyber Risk and Cyber Insurance; Macquarie University Faculty of Business and Economics Research Paper. Available online: https://ssrn.com/abstract=3200166 (accessed on 15 July 2019).
21. Geluvara, B.; Satwik, P.M.; Ashok Kumar, T.A. The Future of Cybersecurity: Major Role of Artificial Intelligence, Machine Learning, and Deep Learning in Cyberspace. In Proceedings of the International Conference on Computer Networks and Communication Technologies, ICCNCT 2018, Coimbatore, India, 26–27 April 2018; Lecture Notes on Data Engineering and Communications Technologies; Springer: Singapore, 2018; Volume 15, pp. 739–747.
22. Xi, J.; Li, A.; Wang, M. An efficient non negative matrix factorization model for finding cancer associated genes by integrating data from genome, transcriptome and interactome. In Proceedings of the 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March 2018; pp. 1–6.
23. Takacs, G.; Pilaszy, I.; Tikk, D.; Nemeth, B. Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem. In Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, 23–25 October 2008; pp. 267–274 .
24. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
25. Polemi, D.; Ntouskas, T.; Georgakakis, E.; Douligeris, C.; Theoharidou, M.; Gritzalis, D. S Port: Collaborative Security Management of Port Information Systems. In Proceedings of the 4th International Conference on Information, Intelligence, Systems and Applications, Piraeus, Greece, 10–12 July 2013; pp. 1–6.
26. Schauer, S.; Stamer, M.; Bosse, C.; Pavlidis, M. An adaptive supply chain cyber risk management methodology. In Proceedings of the Hamburg International Conference of Logistics (HICL), Hamburg, Germany, 12–14 October 2017; pp. 405–425.
27. Tosh, D.K.; Shetty, S.; Sengupta, S.; Kesan, J.P.; Kamhoua, C.A. Risk Management Using Cyber-Threat Information Sharing and Cyber-Insurance. In Proceedings of the 7th International EAI Conference on Game Theory for Networks, GameNets 2017, Knoxville, TN, USA, 9 May 2017; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Cham, Switzerland; Volume 212, pp. 154–164.
28. Settannia, G.; Skopika, F.; Shovgenyaa, Y.; Fiedlera, R.; Carolanb, M. A collaborative cyber incident management system for European interconnected critical infrastructures. *J. Inf. Secur. Appl.* **2017**, *34*, 166–182. [CrossRef]
29. Skopik, F.; Settanni, G.; Fiedler, R. The Importance of Information Sharing and Its Numerous Dimensions to Circumvent Incidents and Mitigate Cyber Threats. In *Collaborative Cyber Threat Intelligence*; Taylor and Francis: London, UK, 2017; Chapter 4, pp. 129–186.
30. Jensen, C.D. The Importance of Trust in Computer Security. In Proceedings of the Trust Management VIII, 8th IFIP WG 11.11 International Conference, IFIPTM 2014, Singapore, 7–10 July 2014; IFIP Advances in Information and Communication Technology (IFIPAICT); Springer: Berlin/Heidelberg, Germany; Volume 430, pp. 1–12.
31. Cristin, G.; Nicholas, P. *A Framework for Cybersecurity Information Sharing and Risk Reduction*; Microsoft Research; Microsoft Corporation: Redmond, WA, USA, 2015.
32. kumar Bokde, D.; Girase, S.; Mukhopadhyay, D. Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey. *Int. J. Adv. Found. Res. Comput.* **2014**, *1*, 111–118.

33. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the 4th ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
34. Yu, H.-F.; Hsieh, C.-J.; Si, S.; Dhillon, I. Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems. In Proceedings of the 2012 IEEE 12th International Conference on Data Mining, Brussels, Belgium, 10–13 December 2012; pp. 765–774.
35. Oggretir, M.; Cemgil, A.T. Comparison of collaborative deep learning and nonnegative matrix factorization for recommender systems. In Proceedings of the 25th Signal Processing and Communications Applications Conference, Antalya, Turkey, 15–18 May 2017; pp. 1–4.
36. Zhan, J.; Hsieh, C.-L.; Wang, C.; Hsu, T.S.; Liau, C.J.; Wang, D.W. Privacy-Preserving Collaborative Recommender Systems. *IEEE Trans. Syst. Man Cybern.* **2010**, *40*, 472–476. [CrossRef]
37. Kaur, H.; Kumar, N.; Batra, S. An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system. *Future Gener. Comput. Syst.* **2018**, *86*, 297–307. [CrossRef]
38. Polatidisa, N.; Georgiadisa, C.K.; Pimenidis, E.; Mouratidis, H. Privacy-preserving collaborative recommendations based on random perturbations. *Expert Syst. Appl.* **2017**, *71*, 18–25. [CrossRef]
39. Liu, S.; Liu, A.; Li, Z.; Liu, G.; Xu, J.; Zhao, L.; Zheng, K. Privacy-Preserving Collaborative Web Services QoS Prediction via Differential Privacy. In Proceedings of the Web and Big Data: First International Joint Conference, APWeb-WAIM 2017, Beijing, China, 7–9 July 2017; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10366, pp. 200–214.
40. Nikolaenko, V.; Weinsberg, U.; Joye, M.; Taft, N.; Boneh, D. Privacy-Preserving Matrix Factorization. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany, 4–8 November 2013; pp. 801–812.