www.czasopisma.pan.pl PAN www.journals.pan.pl
INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2017, VOL. 63, NO. 3, PP. 329-336
Manuscript received May 27, 2017; revised June, 2017. DOI: 10.1515/eletel-2017-0044

Trust and Risk Assessment Model of Popular Software Based on Known Vulnerabilities

Marek Janiszewski, Anna Felkner, and Jakub Olszak

***Abstract*—This paper presents a new concept of an approach to risk assessment which can be done on the basis of publicly available information about vulnerabilities. The presented approach uses also the notion of trust and implements many concepts used in so called trust and reputation management systems (which are widely used in WSN, MANET or P2P networks, but also in e-commerce platforms). The article shows first outcomes obtained from the presented model. The outcomes demonstrate that the model can be implemented in real system to make software management more quantified and objective process, which can have real and beneficial impact on institutional security. In article, however the emphasis was set not on the model itself (which can be easily changed) but on the possibility of finding useful information about vulnerabilities.**

***Keywords*—software vulnerabilities, risk assessment, software management, trust and reputation management models, 0-day vulnerabilities forecast, risk of information systems, prediction model**

I. Introduction

RISK assessment of software based on objective and quantified measures is still very rare, because of the fact that this task is not trivial. Most approaches use qualified measures to assess risk of software or (more common) of information systems which use various software. Risk assessment is tried to be done on the basis of a methodology, which tries to ensure that the outcome of risk assessment process will be objective. Still such approach imposes that this task is always done by an analyst or an auditor (of course with support of a methodology and a system, which can simplify this process). Because of human factor involved in the process of risk assessment, the results of such process are always subjective to a certain extent. Moreover, process is time-consuming and is repeated at most once every couple of months (the most common risk assessment interval is one or two years). Even more important is the fact that such analysis is not often concentrated on technical vulnerabilities of software, but such vulnerabilities take a crucial role in the evaluation of security of a system. On the other hand, new vulnerabilities are discovered every time, so the risk assessment should be done in a real time.

Without any doubt one can claim that every piece of software has some vulnerabilities, despite of the fact that many of them are still not discovered. On the other hand, claims that every software is equally vulnerable, cannot be justified, and are simply not true. The normal disclosure process of the new discovered vulnerability assumes that the vendor of a software in which a vulnerability has been found is informed at first. Vendor, after an investigation and research, prepares an appropriate software fix (also called as a patch or an update), which should dispatch this vulnerability. After a patch is prepared, vendor informs (for example, through published bulletins on a vendor's website) all potential users and a community about the new patch, and of course about the vulnerability. This process is known as coordinated disclosure process. Of course, vulnerabilities are discovered not only by white hats (analysts of cybersecurity that aims is to increase security of software), but also by black hats (crackers that aims at breaking information systems to gain some information or to prevent from honest usage of these systems). When a cracker find a new vulnerability (0day), he is trying to use it to gain benefits instead of informing a vendor. Because of that, unknown vulnerabilities can have a great potential of breaking security of the systems.

The aim of this paper is to present a new model of trust and risk assessment of software which takes into account a lot of publicly available information of existing discovered vulnerabilities. The model on the basis of such information tries to predict and evaluate vulnerabilities which can exists in a software, but which are not yet discovered (more precisely: which are not yet disclosed).

The paper is organized as follows. Section II discusses related work connected to the area of software vulnerability management and also to the area of technical risk assessment of popular software. Section III describes in more detail the main characteristics of proposed model of trust and risk assessment. Section IV lists and describes the sources of information used in proposed system. In section V the formal model of proposed algorithm is presented. The results of trust and risk assessment obtained for a few types of software and for a few vendors are provided in section VI. Conclusions and propositions of future works are provided in section VII.

II. Related works

There many papers which have addressed the problem of forecasting vulnerabilities. Authors of [3] emphasize that estimating risk connected to zero-day vulnerabilities is important. In paper [4] results of experiments on predicting the number of vulnerabilities on each given day can be found.

Papers [1], [2] address similar problem, which is prediction of the time to the next vulnerability for a given software through various machine learning techniques. The paper use, however, only information provided by NVD (National Vulnerability Database) [10]. The results obtained by the authors

M. Janiszewski is with NASK - Research and Academic Computer Network, Kolska 12, Warsaw, Poland, and with the Institute of Telecommunication, Warsaw University of Technology, Nowowiejska 15/19, Warsaw, Poland (e-mail: marek.janiszewski@nask.pl).

A. Felkner and J. Olszak are with NASK - Research and Academic Computer Network, Kolska 12, Warsaw, Poland (e-mail: {anna.felkner; jakub.olszak}@nask.pl).

show that information about vulnerabilities provided by NVD have poor prediction capability, and have a real value only for a few vendors and types of software. Articles listed above used only NVD as the most known vulnerability database. However, NVD is not the only one database and is not the one which provide the most information. The several limitations of NVD database were also indicated by the author of paper [6].

Our work implements much different approach than in papers [1] and [2] for several reasons. First of all, we use much more information from several databases and, what is even more important, we use data not only about vulnerabilities itself but mainly about patches and software fixes. We also provide an in-depth critical analysis of potential sources of information about vulnerabilities and patches. Secondly, authors of the aforementioned papers use only some part of information presented in the databases (namely: publication date of a vulnerability and CPE identifier of vulnerable software), they do not analyze severity of a vulnerability (for example by means of CVSS), we try use as much information as it can be beneficial. Thirdly, our main aim is not to predict when a new vulnerability can occur but to estimate the risk related to a piece of software on the base of evaluation of trust to that software. Of course, our model can be used to predict how many vulnerabilities would affect a software in a certain period of time, but such forecasts can only be done for a quite long periods and is used only to validate our model and select appropriate values of parameters used in model. The exact date of occurrence of new vulnerability cannot be easily predicted also because of the fact that the date of publishing information about new vulnerability is not related in a deterministic way to the date of finding new vulnerability. One can assume that the number of total vulnerabilities in a software is a function of reliability of programmers of the software and the vendor itself. Also, the number of discovered vulnerabilities can be perceived as a function of total vulnerabilities. We can assume that date of discovery of new vulnerability is correlated with the total number of vulnerabilities, but still in the majority we cannot analyze the date of finding a bug, but the publication date of this bug. This means that the speed of providing a patch to a software can influence the distribution of dates of publishing vulnerabilities, which cannot be perceived as a deterministic process.

III. The need for vulnerability management

It is important that vulnerabilities exist in a software since the release of this software (rarely a vulnerability can be introduced by another patch), so the important thing is to evaluate how many vulnerabilities are not discovered.

The number of published vulnerabilities is generally increasing (in 2017 till May the number of vulnerabilities is almost equal to the number of vulnerabilities found in previous years during whole year), which can be observed in Fig. 1. It can be noticed also even more important fact [15] - the overall severity of vulnerabilities is also increasing. Nowadays no one can claim that any software is free from vulnerabilities, but of course some types of software are more vulnerable than the other, and also some vendors produce software more

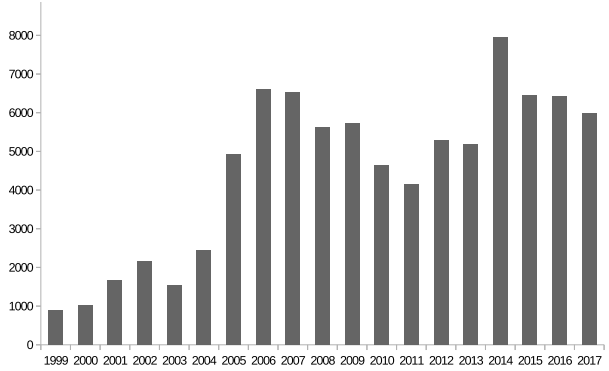

Fig. 1. Number of published vulnerabilities (in NVD database) by year

vulnerable than the one produced by the other vendors or communities. Because of these facts, the need for effective vulnerability management is high.

Vulnerability management system should be present in two areas. First of all, it should support an administrator in update management process. An administrator of a system should be able to indicate all software components composing the system. The vulnerabilities and patches published since the last update should be detected by using an automatic system which aggregates information about patches and vulnerabilities gained from various sources. Secondly, vulnerability management system should estimate technical risk related to software used. The risk is the effect of existing but not discovered and published vulnerabilities. The risk could be estimated on the base of previous experience in using the software and other software provided by the same vendor, in the area of security.

The aim of this work is to address these two very important aspects of vulnerability management: first of all, by review and analysis the sources of information and by presenting a framework to aggregate such information, and secondly by proposing a new model of risk estimation for different software and vendors. The result of such model should indicate whether such software could be used in important or critical systems.

IV. Information sources

A. Vulnerability management

Vulnerable software can be indicated by its name and version, but many other ways of identification exist. CPE (Common Platform Enumeration) can be perceived as such way of identification of vulnerable software and hardware. CPE dictionary [24] is managed by Mitre. Software identification by CPE has some disadvantages. First of all, CPE is tightly related to vulnerabilities (the interesting fact is that CPE identifier is assigned to a software only when a vulnerability in that software is found). Secondly, CPE is not commonly used by software vendors.

CVSS (Common Vulnerability Scoring System) is a framework and a methodology for characterization of the impact of vulnerabilities. CVSS score can be perceived as an indicator of severity of a vulnerability. CVSS score can be between 0.0 and 10.0, where values between 0.0 and 3.9 indicate ”low” severity, and values between 7.0 and 10.0 indicate ”high”

or "critical" severity. CVSS score is commonly used as and indicator of severity of vulnerabilities, but it cannot be found in all information sources.

B. Review and analysis of sources of information

During our research we have analyzed widely available databases of vulnerabilities and patches. Some of the databases have been rejected because of various reasons (such as ambiguity of dispatches, inactivity in adding new vulnerabilities, lack of basic information, problems with accessibility). After elimination of such databases, the following sources have been analyzed:

- Symantec [7]
- CVE (Common Vulnerabilities and Exposures) [8]
- Dragonsoft [9]
- NVD (National Vulnerability Database) [10]
- SecurityFocus [11]
- Security Tracker [12]
- US-CERT Vulnerability Notes Database [13]
- CIRCL [14]
- CVEdetails [15]
- Fulldisclosure [16]
- Exploit-db [17]
- Intelligent Exploit [18]
- Metasploit [19]
- Sans [20]
- Vulnerability-lab [21]
- Vulners [22]
- vfeed [23]

During that research some of initially selected sources (namely: Dragonsoft [9] and Intelligent Exploit [18] have stopped functioning, which prove that all issues related to vulnerabilities are highly dynamic (we have also noticed during research many changes in most of analyzed sources of information).

The main aim of the analysis is an assessment of usability of information in various sources and a description of utilizations of gained information in further works.

1) Symantec:
Symantec provides one of the most popular portal about security of information systems. Portal shares information about recently discovered threats such as viruses, vulnerabilities and spam campaigns. The structure of information enables automatic processing of available information.

2) CVE - Common Vulnerabilities and Exposures:
CVE is a list of identificators of vulnerabilities, which is managed by Mitre. Most of publicly disclosed vulnerabilities have its own CVE-id, which is globally unique and commonly used. The database provided by Mitre do not present many information about vulnerabilities.

3) NVD - National Vulnerability Database:
NVD repository is managed by National Institute of Standards and Technology (NIST). All vulnerabilities in this repository are connected to CVE identifier and precisely characterized. Information in the database can be automatically processed.

4) SecurityFocus:
SecurityFocus provides BugTraq (which is a mailing list aimed at exchange of information about the newest vulnerabilities), SecurityFocus Vulnerability Database (which provides the most current information related to all platforms and services) and SecurityFocus Mailing List (consisted of 31 mailing lists dedicated to information security professionals).

5) US-CERT:
US-CERT coordinates protection and reacts in case of incidents in the Internet in the USA. US-CERT Vulnerability Notes Database provides information about vulnerabilities in software.

6) CIRCL - The Computer Incident Response Center Luxembourg:
CIRCL team provides CVE-search interface, which enables search of information about publicly disclosed vulnerabilities in hardware and software. It aggregates information gained from: NIST National Vulnerability Database and statistics about incidents and threats ranking provided by CIRCL.

7) CVEdetails:
CVEdetails provides an easy to use interface which presents various data about vulnerabilities. It could be also used to present various statistics about vulnerabilities. Information in the portal is retrieved in the majority from NVD database, but also from different sources (such as Exploit-db and Metasploit).

8) Fulldisclosure:
Fulldisclosure enables exchange of specific information about vulnerabilities as well as tools, documents and events related to cybersecurity. Sometimes description about new vulnerabilities can be found in Fulldisclosure earlier than in official sources, but the structure of the portal do not enable automatic processing of information.

9) Exploit-db:
Exploit-db provides extensive repository of exploits, which can be used to hack information systems. Information about vulnerabilities cannot be gained from this database in a direct and an easy way.

10) Vulners:
Vulners.com is an Internet portal, created by group of experts and researches in the area of IT security. Vulners provides database of large amount of vulnerabilities (and also security patches of some vendors). The structure of information provided enables automatic processing. This source is very comprehensive. Vulners aggregates data from many various sources (for example from most of sources which was described in the analysis) and still the number of supported sources is increasing.

11) Vendor's Biulletins:
All vendors provide information about vulnerabilities and patches related to software created by them. Of course the quality of information in various bulletins is very diversified. Most of security bulletins of vendors have cross-reference do CVE-ID, but the characteristic of platform affected by a vulnerability is done in various ways (for example by providing only name an version of the affected application, or in very rare case by providing CPE of affected systems).

12) Summary:
Some of the selected sources aggregate information form different sources, for example Fulldisclosure uses information from Vulnerability Lab, Portcullis Advisories, Asterisk Security Team, Securify B.V. Exploit-db uses information from Secunia, Vulnerability-lab and Intelligent Exploit. Vulners aggregates information from many sources.

C. Limitations and inconsistencies

During research in the area of vulnerability management we have found many limitations and inconsistencies related to publicly accessible vulnerability and patch databases. Some of them are indicated in this section.

In a perfect world, the date of vulnerability discovery combined with the date of issuing a patch to prevent this vulnerability would make a fundamental set of information to assess the reliability of a vendor. However, vendors in most do not provide information about the date of discovery of a vulnerability. Despite the fact that for some of vulnerabilities date of discovery could be found in the Internet, such information is not provided in any of the common well-known sources and is relatively rare. Finding such information would be connected to an in-depth analysis of every vulnerability and it cannot be done through automatic information processing.

It is worth to note that in Mitre CVE database, another date can be found which indicate when a particular CVE-ID was allocated or reserved, but Mitre says [8]:

> ”The entry creation date may reflect when the CVE-ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.”

Because of that we cannot be sure that this date reflects the date of discovery of vulnerability. In practice this date can be before or after the date of discovery a vulnerability. Big vendors which have become CVE Number Authorities - CNA (currently 58) are authorized to assign CVE-IDs to vulnerabilities affecting products within their distinct. Such organizations can reserve in advance some of CVE-ID and because of that the entry creation date can be before the date of discovery a vulnerability. On the other hand, a vendor (which may or may not be a CNA) can assign a number of a vulnerability just before public disclosure.

It should be noted that CPE, which is a very good way to characterize an affected software of a vulnerability, is not used in all types of sources. For example vendors bulletins which provides information about patches related to certain vulnerability, in the majority do not use CPE to identify software to which the patch is released. Because of that the relation between vulnerabilities and patches connected to a software cannot be easily identified.

In many cases different sources present various information about a vulnerability, which sometimes are contradictory. The most common inconsistencies are related to characteristics of affected software and version affected by a vulnerability or a patch.

D. Selection of useful information and information sources

Most of vulnerabilities are full disclosed after a patch by a vendor is provided, but sometimes for some reason vendor cannot provide patch before the vulnerability are widely known. In general, such case can be perceived as a vulnerability of vendor itself and because of that the penalty of vendor’s reputation should be higher in such case than when a vulnerability is disclosed after providing a patch.

As we have pointed out we cannot acquire information about discovery date of a new vulnerability, on the other hand information about quickness of releasing security update by a vendor, would be very useful characteristics to estimate trust to vendor. Because of that we decide to make use of slightly different information. When a vulnerability can affect various vendors (which can occur when a vulnerability is for example related to a package used in various systems), various vendors can provide a patch at different time. Of course, the sooner patch will be provided the greater trust can be put to the vendor (and of course to that specific software). On the other hand, a vulnerability can affect various versions of a software provided by a vendor. For different versions, the vendor can release a patch at different time (for example sooner for current software), or even do not provide patch at all (because of reaching end of support time for that version). Because of that, in our model, we try to find a first date anywise connected to a vulnerability (this could be date of releasing a patch by a different vendor or date of publishing a vulnerability) and measure a delay of releasing a patch by specific vendor or to that specific version of software.

When a piece of software has few vulnerabilities, it could be a result of two main reason:

- in optimistic view: due to the fact that such software is highly secure and has no or not many vulnerabilities,
- in pessimistic view: due to the fact that software has not many users or is not very important, vulnerabilities are not explored or reported.

Because of that we should address in our model all above cases.

Information taken into account during assessment of trust or risk to a software or vendor:

- number and severity of vulnerabilities in software (in general the higher CVSS of a vulnerability is, the higher risk and the lower trust is)
- the existence of patch to prevent the vulnerability (unexpected and unpatched vulnerability has more serious impact on trust and risk)
- time of reporting a vulnerability (the newest vulnerabilities have greater impact on trust and risk)
- time (number of days) between first information about a patch and publishing a patch (or patches)

Information which could be taken into account in the future (currently not used):

- vulnerabilities in other products of the same vendor
- popularity of a software affected (this parameter is taken from external sources - is provided by operator of the system)

Cross-correlation of information about vulnerabilities which do not have CVE identifier provided, is not possible. On the other hand, great majority of vulnerabilities of well-known software have CVE-ID provided. The above statement is true especially in steady state (it means that after a patch has been released). Because of these facts, we decide to use only information about vulnerabilities with CVE identifier provided.

On the basis of our analysis we have decided to use the following set of information:

- information about software:
  - vendor
  - software name / package name
  - software version
  - platform specification (if present)
- information about vulnerability:
  - CVE-ID
  - CVSS
  - CPEs of affected software
  - disclosure date
- information about patch:
  - vendor
  - affected software name / package name
  - software version
  - platform specification (if present)
  - release date
  - CVE-IDs related

Such types of information on the one hand can be relatively easy retrieved from public sources and on the other hand could be sufficient to characterize vendor's approach to the area of vulnerability management (and to evaluate risk related to such vendor or software).

We do not use information of patches which are not related to any CVE-ID.

As we have pointed out it is not possible to match software name or package name to CPE directly. To match information about vulnerability to specific software defined by its name or name of the package we use two different approaches. First of all, we use CVE-ID. Secondly, we use our search algorithm to find the best matching between software name and CPE and then to decide whether the scope of the CPE and the name of the software are the same or whether the one set are a subset of the other or whether the sets are disjoint.

All needed information are retrieved in the majority from Vulners but also from NVD database and vendors' security bulletins.

V. Model of Trust and Risk Assessment

A. Variables

Let us denominate:

1) Parameters of model:

- α - forgetting factor: $\alpha \in (0, 1)$
- β - weight of risk of unpatched vulnerabilities: $\beta \in < 0, 1 >$
- T - period length in days (The time from 1999 - when the first CVE was aligned, is split in periods, that has length T), default period length = 1 quarter
- T_n - the n-th period

2) Parameters gained by analysis of information from sources:

- $V_{V:a}^{T_i:N}$ - number of vulnerabilities of Vendor a published during period T_i
- $V_{V:a}^{T_i:S}$ - sum of CVSS score for all vulnerabilities of Vendor a published during period T_i
- $D_{V:a}^{T_i:S}$ - sum of delay in days of publishing patches to all vulnerabilities related to Vendor a published during period T_i

Analogously:

- $V_{S:a}^{T_i:N}$ - number of vulnerabilities of Software a published during period T_i
- $V_{S:a}^{T_i:S}$ - sum of CVSS score for all vulnerabilities of Software a published during period T_i
- $D_{S:a}^{T_i:S}$ - sum of delay in days of publishing patches to all vulnerabilities related to Software a published during period T_i

3) Derived values:

- $EV_{V:a}^{T_{n+1}:N}$ - expected number of vulnerabilities of Vendor "a" published during period T_{n+1}, estimated on the basis of data from previous periods
- $VV_{V:a}^{T_{n+1}:N}$ - expected error of estimations of number of vulnerabilities published during period T_{n+1}
- $EV_{V:a}^{T_{n+1}:S}$ - sum of CVSS score for all vulnerabilities of Vendor "a" published during period T_{+1}, estimated on the basis of data from previous periods
- $VV_{V:a}^{T_{n+1}:S}$ - expected error of estimations of CVSS sum for all vulnerabilities published during period T_{n+1}

In an analogous way can be defined values for specific software (instead of vendor)

4) Trust and risk values:

- $P_{V:a}^{T_{n+1}}$- total penalty of vendor a during period $T_(n+1)$
- $P_V^{T_{n+1}} = max(P_{V:a}^{T_{n+1}}, P_{V:b}^{T_{n+1}}, \ldots)$ - maximum penalty of all vendors
- $R_{V:a}^{T_{n+1}}$- total risk of vendor a during period T_{n+1}
- $T_{V:a}^{T_{n+1}}$- total trust to vendor a during period T_{n+1} Trust and risk values are between 0 and 1. Analogously:
- $P_{S:a}^{T_{n+1}}$- total penalty of software "a" during period T_{n+1}
- $P_S^{T_{n+1}} = \max(P_{S:a}^{T_{n+1}}, P_{S:b}^{T_{n+1}}, \ldots)$ - maximum penalty of all software of the same type (for example: web browsers, operating systems, etc.)
- $R_{S:a}^{T_{n+1}}$- total risk of software a during period T_{n+1}
- $T_{S:a}^{T_{n+1}}$- total trust to software a during period T_{n+1}

B. Main algorithm

1) get all information from sources
2) match all vulnerabilities with software and patches by using algorithm described in section C
3) for each vendor:
   - calculate the number and the sum of vulnerabilities in each period - $V_{V:a}^{T_i:N}, V_{V:a}^{T_i:S}$
   - calculate the sum delay between vulnerability disclosure and patch publishing - $D_{V:a}^{T_i:S}$
   - remember vulnerabilities without patches

- calculate expected number and severity of vulnerabilities in next period - $EV_{V:a}^{T_{n+1}:N}, EV_{V:a}^{T_{n+1}:S}$ and expected error of estimations - $VV_{V:a}^{T_{n+1}:N}, VV_{V:a}^{T_{n+1}:S}$
- calculate the total penalty - $P_{V:a}^{T_{n+1}}$
- calculate the total risk - $R_{V:a}^{T_{n+1}}$

4) calculate the maximum penalty $P_{V}^{T_{n+1}}$
5) for each vendor:
   - calculate the total trust - $T_{V:a}^{T_{n+1}}$
6) for each software:
   - calculate the number and the sum of vulnerabilities in each period - $V_{S:a}^{T_i:N}, V_{S:a}^{T_i:S}$
   - calculate the sum delay between vulnerability disclosure and patch publishing - $D_{S:a}^{T_i:S}$
   - remember vulnerabilities without patches
   - calculate expected number and severity of vulnerabilities in next period - $EV_{S:a}^{T_{n+1}:N}, EV_{S:a}^{T_{n+1}:S}$ and expected error of estimations - $VV_{S:a}^{T_{n+1}:N}, VV_{S:a}^{T_{n+1}:S}$
   - calculate the total penalty - $P_{S:a}^{T_{n+1}}$
   - calculate the total risk - $R_{S:a}^{T_{n+1}}$
7) calculate the maximum penalty $P_{S}^{T_{n+1}}$
8) for each vendor:
   - calculate the total trust - $T_{S:a}^{T_{n+1}}$

C. Algorithm to match vulnerabilities with software and patches

for every vulnerability (identified by CVE-ID):

- find all vulnerable software (identified by CPE or by name and version) related to this vulnerability
- find all patches related to this vulnerability (by matching CVE-ID of this vulnerability)
- compare publication dates of all patches and the publication date of the vulnerability and choose and remember the earliest date
- for every patch related to this vulnerability
  - if vendor of the current patch is not the same as current vendor for which trust and risk is calculated, continue with the next patch
  - calculate the difference in days between publication date of the patch and the date remembered in the previous step
  - remember every software (identified by CPE or by name and version) which is related to the patch
- compare whether for every software affected by the vulnerability, a patch was published

D. Calculations

To calculate the expected number of vulnerabilities of a software or of a vendor in the next period we take into account number of vulnerabilities in the previous periods weighted by forgetting factor, which can be seen on equation 1.

$$EV_{V:a}^{T_{n+1}:N} = \frac{(1-\alpha)\sum_{i=1}^{n} \alpha^{n-i} * V_{V:a}^{T_i:N}}{1-\alpha^n} \tag{1}$$

To calculate the expected error of estimation of number of vulnerabilities in the next period we can use equation 2. To do this we take into account the difference between numbers of vulnerabilities in the previous periods and the expected number of vulnerabilities calculated for previous periods weighted by forgetting factor.

$$VV_{V:a}^{T_{n+1}:N} = \sqrt{\frac{(1-\alpha)\sum_{i=2}^{n}\left[(V_{V:a}^{T_i:N} - EV_{V:a}^{T_i:N})^2 \alpha^{n-i}\right]}{\alpha^2(1-\alpha^n)}} \tag{2}$$

We can calculate in the same way the expected sum of CVSS of vulnerability in the next period, and the expected error of estimation, by using equations 3 and 4.

$$EV_{V:a}^{T_{n+1}:S} = \frac{(1-\alpha)\sum_{i=1}^{n} \alpha^{n-i} * V_{V:a}^{T_i:S}}{1-\alpha^n} \tag{3}$$

$$VV_{V:a}^{T_{n+1}:S} = \sqrt{\frac{(1-\alpha)\sum_{i=2}^{n}\left[(V_{V:a}^{T_i:S} - EV_{V:a}^{T_i:S})^2 \alpha^{n-i}\right]}{\alpha^2(1-\alpha^n)}} \tag{4}$$

To calculate total penalty of a vendor (or a software) we can use the equation 5. The penalty is based on two components: the first is penalty for a vulnerability itself - any vulnerability decreases trust (even if it is patched), because vendor should do everything it can to prevent from vulnerabilities. The second component is the penalty for delays (which indicate how fast a vulnerability will be patched if it appears). It can be noticed, that we take into account the sum of CVSS of all vulnerabilities related to the vendor (or software) - which is the first summand and the average severity of existing vulnerability (which is $\frac{V_{V:a}^{T_i:S}}{V_{V:a}^{T_i:N}}$) multiplied by the sum of delay of releasing a patch - which is the second summand.

$$P_{V:a}^{T_{n+1}} = \sum_{i=1}^{n}\left(V_{V:a}^{T_i:S} + \frac{D_{V:a}^{T_i:S} * V_{V:a}^{T_i:S}}{V_{V:a}^{T_i:N}}\right) \tag{5}$$

After calculating penalty for each vendor, a trust for each vendor can be calculated by using equation 6.

$$T_{V:a}^{T_{n+1}} = 1 - \frac{P_{V:a}^{T_{n+1}}}{P_{V}^{T_{n+1}}} \tag{6}$$

Trust can range between 0.0 (the lack of trust) to 1.0 (the highest trust). It can be noticed that the trust is relative and the takes into account only historic data (do not include any estimations about the future).

The risk of each vendor can be calculated by using equation 7. The first component estimates the risk related to vulnerabilities which are predicted to appear in the next period and also take into account risk related to the fact that patches can be delayed. The second component is related to the risk connected to vulnerabilities which are still unpatched.

$$R_{V:a}^{T_{n+1}} = (1-\beta) * \min\left(10, \frac{V_{EV:a}^{T_{n+1}:S} + \frac{D_{V:a}^{T_n:S}}{T}}{V_{V:a}^{T_{n+1}:N}}\right) + \\ + \beta * maxCVSS_{V:a} \tag{7}$$

$maxCVSS$ in the equation 7 is the highest value of CVSS of a vulnerability which is not patched by the vendor a. Risk can range from 0.0 to 10.0 (similar to CVSS).

Calculations for software can be done analogously. β should be relatively small for estimating risk of vendors but relatively high for estimating risk of specific software.

VI. Results

Our results are presented for a few vendors and for one specific types of software (namely: operating systems). However in practice, presented system/model can be applied to all types of software and all vendors. The area of research was narrowed down because of the fact that some works have to be done to achieve information about other kinds of software due to the necessity of gain information directly from vendor's bulletins. The process of gaining information is related to some technical problems but not to research problems.

The results were obtained for vendors:

- Microsoft
- Debian
- Canonical (Ubuntu)
- Redhat
- Mozilla

The results were obtained for OS:

- RedHat Enterprise Linux 6
- RadHat Enterprise Linux 7
- Debian 7 (Wheezy)
- Debian 8 (Jessie)
- Ubuntu 15
- Ubuntu 16

Vulnerabilities and patches published to 2015.12.31 was used to train our model. Vulnerabilities and patches published since 2016.01.01 to 2016.12.31 was used to test our model. The period length was set as one quarter and the following parameters was evaluated:

- number of vulnerabilities,
- sum of CVSS of vulnerabilities.

and these parameters was compared to the real values obtained in the same periods. The results are presented in Table I and in Table II. The forgetting factor was adapted individually for each software and vendor on the base of the training set. The results in Table I and in Table II can be different from the results obtained from public sources (e.g. CVEdetails [15]) due to the fact that we take into account a vulnerability even if it was anywise related to a software or vendor (not only directly).

It is worth to note that the trust or risk assessment cannot be compared directly to the real data because these parameters indicate only expectations about a future. The results are presented in table III and in table IV.

Results show that the prediction of the number and sum of CVSS of vulnerabilities is sometimes not possible due to high changeability of different factors, which was not considered in the model. On the other hand, the main aim of the model is to evaluate risk connected to usage of specific software or provided by specific vendors as a way to differentiate such software in the context of security. Results suggest that the model can be used in that purpose. Trust of vendors which release closed software is higher than trust of vendors related to open-source software. On the other hand risk related to such software is also generally higher (due to lack of information provided by such vendors). The highest risk is related to Red Hat due to the highest number of delays in releasing patches.

VII. Conclusions and future works

Results of our work, far from being comprehensive, indicate that the proposed approach is promising, but it needs further research. The paper has proved that information needed for effective vulnerability management are dispersed over many databases and usage of all information is not a trivial task.

In the future we plan to develop our model through taking into account more information about vulnerabilities and patches (for example: how a vulnerability was discovered) and also through including different factors (for example connections between different software). The main aim is to create a comprehensive model which can assess in a real time risk related to specific software.

References

[1] S. Zhang, X. Ou, and D. Caragea, "Predicting Cyber Risks through National Vulnerability Database," *Information Security Journal: A Global Perspective*, vol.24, 2015, pp. 194-206, DOI: 10.1080/19393555.2015.1111961
[2] S. Zhang, D. Caragea, and X. Ou, "An Emperical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities," *LNCS 6860*, 2011, pp. 217-231, DOI: 10.1007/978-3-642-23088-2_15
[3] K. Ingols, M. Chu, R. Lippmann, S. Webster, S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," *Annual Computer Security Conference*, ACSAC, 2009, DOI: 10.1109/ACSAC.2009.21
[4] M. McQueen, T. McQueen, W. Boyer, M. Chaffin, "Empirical estimates and observations of 0day vulnerabilities," *42nd Hawaii International Conference on System Sciences*, 2009, pp. 1-12
[5] A. Ozment, Vulnerability Discovery & Software Security, *PhD thesis*, University of Cambridge, 2007
[6] A. Felkner, "Review and analysis of sources of information about vulnerabilities," *Przegld telekomunikacyjny i wiadomoci telekomunikacyjne*, vol. 8-9/2016, 2016, pp. 929-933, DOI: 10.15199/59.2016.8-9.37
[7] Symantec http://www.symantec.com/security_response/landing/ vulnerabilities.jsp - access date: 02.05.2017
[8] Common Vulnerabilities and Exposures (CVE) http://www.cve.mitre.org/ access date: 02.05.2017
[9] Dragonsoft vulnerability database http://vdb.dragonsoft.com/ - access date: 02.05.2016, currently not accessible
[10] National Vulnerability Database http://nvd.nist.gov/ access date: 02.05.2017
[11] SecurityFocus http://www.securityfocus.com/vulnerabilities/ - access date: 02.05.2017
[12] Security Tracker http://www.securitytracker.com/ - access date: 02.05.2017
[13] US-CERT vulnerability notes database http://www.kb.cert.org/vuls/ - access date: 02.05.2017
[14] The Computer Incident Response Center Luxembourg http://cve.circl.lu/ - access date: 02.05.2017
[15] CVEdetails http://www.cvedetails.com/ - access date: 02.05.2017
[16] Fulldisclosure http://seclists.org/fulldisclosure/ - access date: 02.05.2017
[17] Exploit-db http://www.exploit-db.com/ - access date: 02.05.2017
[18] Intelligent Exploit http://www.intelligentexploit.com/ - access date: 02.05.2016, currently not accessible
[19] Metasploit (Rapid7) https://www.rapid7.com/db/ - access date: 02.05.2017
[20] Sans http://isc.sans.edu/diary/ - access date:02.05.2017
[21] Vulnerability-lab http://www.vulnerability-lab.com - access date: 02.05.2017
[22] Vulners.com https://vulners.com/ - access date:02.05.2017
[23] Vfeed https://github.com/toolswatch/vFeed - access date:02.05.2017
[24] CPE dcitionary: https://cpe.mitre.org/ - access date:02.05.2017

TABLE I
RESULTS - NUMBER OF VULNERABILITIES AND SUM OF CVSS FOR VENDORS

Vendor	1st quarter of 2016		2nd quarter of 2016		3rd quarter of 2016		4th quarter of 2016	
	#vuln	CVSS	#vuln	CVSS	#vuln	CVSS	#vuln	CVSS
Microsoft	102	763.1	178	1398.2	124	803.5	172	1242.8
	117.0±39.0	*769.0±317.0*	*114.0±37.0*	*769.0±302.0*	*124.0±42.0*	*826.0±345.0*	*124.0±39.0*	*824.0±329.0*
Debian	342	1986.8	292	1800.0	274	1649.7	197	1025.6
	243.0±19.0	*1426.0±124.0*	*291.0±71.0*	*1701.0±402.0*	*292.0±50.0*	*1701.0±287.0*	*283.0±38.0*	*1676.0±208.0*
Ubuntu (Canonical)	246	1532.5	317	1816.4	280	1703.5	355	1876.5
	243.0±27.0	*1494.0±135.0*	*245.0±19.0*	*1519.0±85.0*	*284.0±55.0*	*1712.0±245.0*	*282.0±37.0*	*1707.0±145.0*
Redhat	265	1816.6	359	2399.5	317	2155.8	224	1308.7
	266.0±86.0	*1828.0±724.0*	*266.0±79.0*	*1826.0±659.0*	*280.0±81.0*	*1923.0±645.0*	*285.0±76.0*	*1963.0±596.0*
Mozilla	65	424.9	29	183.2	45	273.1	0	0.0
	31.0±16.0	*214.0±126.0*	*32.0±17.0*	*219.0±128.0*	*32.0±17.0*	*218.0±127.0*	*32.0±16.0*	*219.0±126.0*

Expected values from the model is set *in italics*, the real values is set in romans
Expected values consist of the estimated value and expected error of estimation
#vuln - number of vulnerabilities; CVSS - sum of CVSS of vulnerabilities

TABLE II
RESULTS - NUMBER OF VULNERABILITIES AND SUM OF CVSS FOR OPERATING SYSTEMS

OS	1st quarter of 2016		2nd quarter of 2016		3rd quarter of 2016		4th quarter of 2016	
	#vuln	CVSS	#vuln	CVSS	#vuln	CVSS	#vuln	CVSS
RedHat EL 6	230	1596.9	322	2217.5	267	1856.9	182	1044.1
	232.0±77.0	*1603.0±651.0*	*232.0±71.0*	*1602.0±596.0*	*245.0±74.0*	*1702.0±599.0*	*248.0±69.0*	*1727.0±552.0*
RedHat EL 7	163	967.5	207	1251.6	172	1013.9	132	550.0
	162.0±58.0	*968.0±284.0*	*162.0±49.0*	*968.0±230.0*	*175.0±48.0*	*1065.0±250.0*	*174.0±40.0*	*1047.0±205.0*
Debian 7 (Wheezy)	227	1271.2	27	176.8	4	26.6	2	17.2
	161.0±14.0	*845.0±96.0*	*226.0±66.0*	*1267.0±424.0*	*29.0±198.0*	*188.0±1086.0*	*4.0±32.0*	*28.0±194.0*
Debian 8 (Jessie)	325	1904.4	298	1744.2	272	1628.2	194	910.6
	224.0±59.0	*1303.0±322.0*	*298.0±91.0*	*1748.0±543.0*	*298.0±48.0*	*1745.0±277.0*	*279.0±33.0*	*1659.0±173.0*
Ubuntu 15	58	345.4	129	708.8	3	9.2	0	0.0
	40.0±11.0	*249.0±68.0*	*44.0±12.0*	*269.0±70.0*	*58.0±31.0*	*344.0±165.0*	*50.0±34.0*	*295.0±188.0*
Ubuntu 16	-	-	125	656.1	15	60.6	8	50.0
	-	-	-	-	*125.0±125.0*	*656.0±656.0*	*16.0±108.0*	*66.0±587.0*

Expected values from the model is set *in italics*, the real values is set in romans
Expected values consist of the estimated value and expected error of estimation
#vuln - number of vulnerabilities; CVSS - sum of CVSS of vulnerabilities

TABLE III
RESULTS - TRUST AND RISK OF VENDORS

Vendor	1st quarter of 2016		2nd quarter of 2016		3rd quarter of 2016		4th quarter of 2016	
	trust	risk	trust	risk	trust	risk	trust	risk
Microsoft	1.0	7.9	1.0	7.9	1.0	8.0	1.0	7.5
Debian	0.6	7.2	0.6	6.9	0.6	6.6	0.6	6.5
Ubuntu (Canonical)	0.6	7.6	0.6	7.9	0.5	7.6	0.5	6.7
Redhat	0.0	9.4	0.0	9.3	0.0	8.7	0.0	7.8
Mozilla	1.0	7.0	1.0	6.9	1.0	6.8	1.0	6.9

TABLE IV
RESULTS - TRUST AND RISK OF OPERATING SYSTEMS

Vendor	1st quarter of 2016		2nd quarter of 2016		3rd quarter of 2016		4th quarter of 2016	
	trust	risk	trust	risk	trust	risk	trust	risk
RedHat Enterprise Linux 6	0.1	8.2	0.0	7.6	0.0	7.5	0.0	7.1
RadHat Enterprise Linux 7	0.7	7.7	0.7	7.3	0.7	7.4	0.6	6.7
Debian 7 (Wheezy)	0.9	6.2	0.9	6.2	0.9	6.2	0.9	6.2
Debian 8 (Jessie)	0.9	6.9	0.9	6.7	0.9	6.5	0.9	6.5
Ubuntu 15	1.0	6.5	1.0	6.4	1.0	6.1	1.0	6.0
Ubuntu 16	-	-	-	-	1.0	5.7	1.0	5.5