A Recommender System for User-Specific Vulnerability Scoring

Karlsson, Linus; Nikbakht Bideh, Pegah; Hell, Martin

Published in:
CRiSIS 2019: Risks and Security of Internet and Systems

DOI:
10.1007/978-3-030-41568-6

2020

Document Version:
Peer reviewed version (aka post-print)

A Recommender System for User-Specific Vulnerability Scoring

Linus Karlsson, Pegah Nikbakht Bideh, Martin Hell

Lund University, Department of Electrical and Information Technology, Sweden
{linus.karlsson, pegah.nikbakht_bideh, martin.hell}@eit.lth.se

Abstract. With the inclusion of external software components in their software, vendors also need to identify and evaluate vulnerabilities in the components they use. A growing number of external components makes this process more time-consuming, as vendors need to evaluate the severity and applicability of published vulnerabilities. The CVSS score is used to rank the severity of a vulnerability, but in its simplest form, it fails to take user properties into account. The CVSS also defines an environmental metric, allowing organizations to manually define individual impact requirements. However, it is limited to explicitly defined user information and only a subset of vulnerability properties is used in the metric. In this paper we address these shortcomings by presenting a recommender system specifically targeting software vulnerabilities. The recommender considers both user history, explicit user properties, and domain based knowledge. It provides a utility metric for each vulnerability, targeting the specific organization's requirements and needs. An initial evaluation with industry participants shows that the recommender can generate a metric closer to the users' reference rankings, based on predictive and rank accuracy metrics, compared to using CVSS environmental score.

1 Introduction

The Common Vulnerability Scoring System (CVSS) [4,8] defines a severity ranking for vulnerabilities. The base score does not take into account individual preferences of users. Instead, CVSS has an environmental metric which can be used to modify the base score such that it represents user dependent properties of vulnerabilities. It will rewrite the confidentiality, integrity, and availability metrics both to adjust them according to measures already taken by the organization, but also to capture the actual impact such loss would have on the organization. As this will differ between organizations, such a modified metric will better reflect the actual severity of a vulnerability to that organization.

The environmental metrics must be evaluated on a per vulnerability basis and are handled manually. This is both time consuming, error prone, and can lead to inconsistencies in case there are several vulnerabilities and they are handled by different analysts. Moreover, the environmental metric, though unique for the organization, only constitutes the sub-metrics available in the base score. Additional information that might affect the organization is not covered.

Recommender systems work by analyzing information about user preferences, and combine this with information about items, or with the history of other users. Their goal is to output recommendations targeting the specific user.

In this paper, we explore ways to improve measuring how a vulnerability affects an organization. Using machine learning techniques applied to recommender systems, we combine different properties and metrics in order to capture vulnerability data and map it to requirements of the specific organization. Compared to CVSS environmental metrics, our method provides several advantages.

First, the requirements for the organization is derived by combining explicit requirements with requirements learned from previous analysis of vulnerabilities. This data driven approach will not only use personal preferences, but also take into account how real vulnerabilities have been evaluated previously. Such learned data is able to capture information that might be overseen by analysts, or that are difficult to express. Second, our approach is general and is not restricted to a certain group of properties. It can be amended with new metrics if needed, focusing on metrics relevant for the given organization or device.

Our goal is to design a recommender that provides a personalized severity assessment based on a user profile. The profile is both explicit, based on the users' own choices, and implicit as the recommender learns from the users' previous actions. We also support inclusion of domain knowledge into the system, and discuss how the different parts can be weighted. Suitable similarity functions are used to form a utility function that outputs the personalized severity assessment. The recommender is also evaluated using participants from the industry. Though the evaluation is small scale, the results indicate that our recommender system is able to provide severity information that is closer to the users' actual preferences than the CVSS environmental score.

Compared to previous work such as [3, 5–7, 14], our approach uses more features, consider user preferences, learns from past user behavior, and/or provides scores instead of suggested actions (cf. [5]).

2 Recommenders and Vulnerability Severity Ratings

Generally, the goal of a recommender is to present recommendations of *items* to a set of *users*. An item can be for example a movie, a song, or a website. The idea is that the recommender should present a subset of items to the user, such that the user finds this subset relevant. The subset is found by matching user preferences or activity using a learnt profile and sometimes other similar users' activity. In a shopping scenario, the added value for the user also leads to higher sales. In this paper, the goal of the recommender is to add value to an end-user by tailoring the severity score for vulnerabilities.

There are three major categories of recommender systems [1]: knowledge-based systems, content-based systems, and collaborative filtering. Other than these, recommendations can be generated from domain-specific knowledge. This generates recommendations for a specific field of knowledge, and is designed specifically to handle data for that domain.

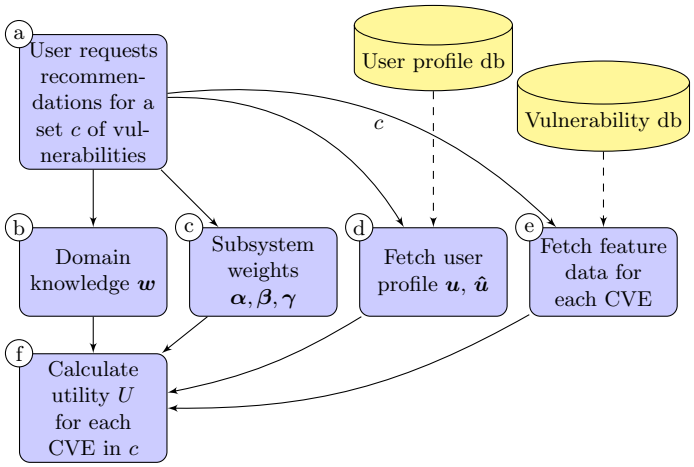

Fig. 1. Flow chart of recommendation generation

Many vulnerabilities are reported and given a CVE identifier. For each vulnerability, NVD provides a severity score. This score, denoted the base score, uses exploitability and impact submetrics in order to define a severity score between 0–10. This score is made to be reproducible and organization independent. Instead, the environmental score can be used to adapt the base score to an organization's requirements and needs.

3 System Model

We have identified the following requirements: 1) The recommender should give reasonable recommendations for new users of the system, avoiding the cold-start problem. 2) It should allow the user to select certain preferences that the system will honor. 3) It should expose a meaningful subset of user preferences to the user. 4) It should learn from user actions, so that future recommendations are as relevant as possible to the user. To avoid privacy concerns, only the user's own actions are considered.

No single class of recommender system can fulfill all requirements. Instead, we propose a hybrid recommender based on three parts. The first is a *domain-based* subsystem which provides domain-specific knowledge unique to a recommender for vulnerabilities. The second part is a *knowledge-based* subsystem which allows the user to select certain user preferences that they are interested in. Lastly, the third part is a *content-based* subsystem which learns from the user's previous actions to provide more meaningful recommendations for each user.

3.1 Overall Recommender System Design

An overview of the recommendation generation process can be seen in Fig. 1. When a user requests recommendations for a set c of vulnerabilities, the feature

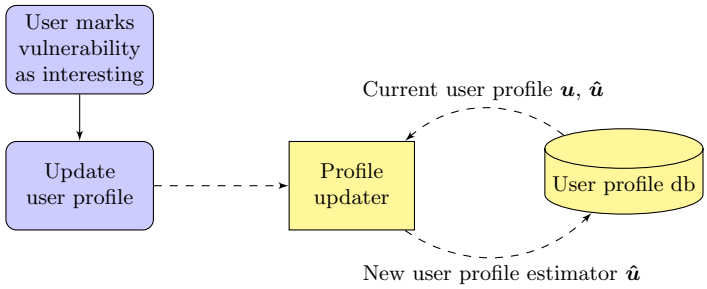

Fig. 2. Flow chart of user rating a vulnerability

data, domain-specific knowledge, user profiles, and weights are fetched from their respective storage. Each of these parts will be described in details in the following sections. These pieces will then be combined in the actual recommender, which then outputs recommendations in the range $[0, 1]$. A higher value means that a vulnerability is more relevant to the user.

Our hybrid recommender system learns user preferences based on the user's interaction with vulnerabilities. An overview of the rating procedure is shown in Fig. 2. First, the user rates a vulnerability based on their own preferences. Next, the current user profile is updated with the new information, so that a new user profile estimated called $\hat{\boldsymbol{u}}$ is stored in the user profile database.

3.2 Feature Representation

A key task in designing a recommender is constructing a good feature extraction stage. In our case, this means that we wish to extract features from each vulnerability, to be used as input to the recommender, see block (e) in Fig. 1. First, a selection of features must be made, and later on their respective feature weight parameters must be decided. We will discuss actual features to use in Section 4.1. We consider the features of a vulnerability as a vector $\boldsymbol{v}$, where each individual feature v_i denotes a specific feature value. Such a value could be of any type, such as a Boolean value, a real number, an integer in a specific range, categorical data, or hierarchical data.

3.3 User Profile Representation

There are two distinct parts of the user profile. First, there is the explicit user profile $\boldsymbol{u}$, where the users explicitly select their own preferences. This is similar to the requirements that can be defined in the CVSS environmental metric. Second, there is the *estimated* user profile $\hat{\boldsymbol{u}}$, which is determined from the user's interactions with the system. The system learns this profile about the user automatically. This allows the system to capture user preferences that are hard to explicitly express for users, either because the feature is complex, or

because the user is unaware of them. The explicit user profile is the knowledge-based part of our hybrid recommender, while the estimated user profile is the content-based part.

Each of the two parts of the user profile is represented as a vector, where each element of the vector describes the interest the user has for each feature. The elements of the vectors are matched with the feature value from above, to find vulnerabilities to recommend to the user.

3.4 Domain-specific Knowledge

The recommendations are not only based on the user profile, but also on a set of domain-specific knowledge, unique to the field of vulnerability assessment, and the same for all users. Such knowledge is required both to provide recommendations suitable for such a highly specific area of interest, but also serves as a component to solve the cold-start problem. The domain-specific knowledge $\boldsymbol{w}$ is represented in the same way as the user profile above.

3.5 Subsystem Weights

As described earlier, the recommender system is a hybrid system with three major parts. The three parts all contribute to the final result of the recommender, but they should be able to do so to different extents depending on the features, see Section 4.1. The subsystem weights are fetched at point (c) in Fig. 1.

The subsystems are given a weight between 0 and 1. Let the vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ describe the weights for the domain-based, knowledge-based, and content-based subsystems, respectively. For any given feature i, the sum $\alpha_i + \beta_i + \gamma_i = 1$. Note that relative weight of each subsystem can vary between different features.

3.6 Similarity Functions

A similarity function compares a value from the user profile, called the target value t_i, with the feature value extracted from the vulnerability v_i. We denote this function $\mathsf{sim}_i(t_i, v_i)$, where $0 \leq \mathsf{sim}_i(t_i, v_i) \leq 1$. Higher value means that the feature value is more similar to the target value. Here, we use the similarity functions given below. For examples of other variants, see e.g. [13].

First, we use $\mathsf{sim}_{\text{dist}}$ which returns the absolute distance between t_i and v_i, scaled to be in the range $[0, 1]$ by knowing the minimum and maximum value. Second we use a scoring function $\mathsf{sim}_{\text{mult}}$ which sees t_i as a multiplier to multiply the feature value v_i with. Third we use $\mathsf{sim}_{\text{daydist}}$ which describes the distance in days between two values, implemented similarly to $\mathsf{sim}_{\text{dist}}$. Fourth, we use $\mathsf{sim}_{\text{cosine}}$ to calculate the cosine similarity between v_i and t_i, note that v_i and t_i are vectors in this case. Fifth, we use $\mathsf{sim}_{\text{boost}}$ for Boolean values, where t_i is simply a constant which is returned if v_i is true.

3.7 Generating Recommendations

Combining the building blocks from the sections above, a complete recommender can now be described. The goal here is to describe a *utility function* U, which takes a given vulnerability $\boldsymbol{v}$ as input, and outputs the utility value, i.e. the user-specific severity assessment. As can be seen at point (f) in Fig. 1, the utility function U is the final step in a series of actions.

Utility U for a vulnerability can be described as:

$$U = \frac{1}{d} \sum_{i=1}^{d} \alpha_i \cdot \mathsf{sim}_i(w_i, v_i) + \beta_i \cdot \mathsf{sim}_i(u_i, v_i) + \gamma_i \cdot \mathsf{sim}_i(\hat{u}_i, v_i)\,, \tag{1}$$

where $\alpha_i, \beta_i, \gamma_i$ are the subsystem coefficients, sim_i is the similarity function for the i^{th} feature, $w_i, u_i, \hat{u}_i$ are the target values for feature i for the different subsystems (i.e. elements of $\boldsymbol{w}, \boldsymbol{u}, \hat{\boldsymbol{u}}$ respectively), and v_i is the feature value for feature i.

Because the similarity functions are limited to the range $[0, 1]$, and $\alpha_i + \beta_i + \gamma_i = 1$, the output of U will be a value between 0 and 1.

3.8 Updating User Profile

For estimating the user profile $\hat{\boldsymbol{u}}$, we wish to combine the previous estimation with the new data about the user's preferences. We consider only input of vulnerabilities that the user *is interested in*, that is, positive training examples. Then, the update function update can be expressed as a function of the form $\hat{\boldsymbol{u}}' = \mathsf{update}(\hat{\boldsymbol{u}}, \boldsymbol{v})$, i.e., a function taking a new vulnerability $\boldsymbol{v}$, the current $\hat{\boldsymbol{u}}$, and returning a new estimation of the user preferences $\hat{\boldsymbol{u}}'$.

We propose an approach inspired by [9], with some adaptions to make the update function applicable for any type of feature, not only text. The proposed update function is given by

$$\mathsf{update}(\hat{\boldsymbol{u}}, \boldsymbol{v}) = (\mathsf{mer}_1(\hat{u}_1, v_1), \ldots, \mathsf{mer}_i(\hat{u}_i, v_i), \ldots, \mathsf{mer}_d(\hat{u}_d, v_d))\,, \tag{2}$$

where d is the number of features, and therefore elements in $\hat{\boldsymbol{u}}$ and $\boldsymbol{v}$.

For each pair $(\hat{u}_i, v_i)$, a *merge function* mer_i is applied. The merge function is similar to the similarity functions sim_i, but instead of comparing two elements, it merges them. The merging needs to be handled different for each feature type, and this construction is thus a generalization of [2, 9].

In this paper we use two different merge functions, $\mathsf{mer}_{\text{mma}}$ which is a merge function based on the Modified Moving Average, and $\mathsf{mer}_{\text{add}}$ which simply performs an element-wise addition over two vectors $\hat{u}_i$ and v_i.

4 Implementation

Given the theoretical model described in the previous section, the actual recommender can now be constructed. This section describes such decisions for our

implemented recommender. We stress that this is an example implementation of the model described in the previous section. Another implementation may choose different features, weights, or functions.

4.1 CVE Features

The implementation has used several sources for vulnerability information. A majority of the data is collected from NVD [11], but also other sites such as CVEdetails [10], and Google have been used. A list of features extracted is available in Table 1, and below we discuss the features in more detail.

Table 1. Feature selection in the implementation, feature types, weights of domain-based (α), knowledge-based (β), and content-based (γ) subsystems, and finally similarity and merge functions

		Subsystem weights			Functions	
Features	Data type	α	β	γ	sim	mer
Impact metrics	Categorical	0.0	0.5	0.5	$\mathsf{sim}_{\text{mult}}$	$\mathsf{mer}_{\text{mma}}$
Exploitability subscore	Numerical	0.0	0.8	0.2	$\mathsf{sim}_{\text{mult}}$	$\mathsf{mer}_{\text{mma}}$
Authentication	Categorical	0.3	0.35	0.35	$\mathsf{sim}_{\text{mult}}$	$\mathsf{mer}_{\text{mma}}$
Access vector	Categorical	0.3	0.35	0.35	$\mathsf{sim}_{\text{dist}}$	$\mathsf{mer}_{\text{mma}}$
CWE	Hierarchical	0.0	0.0	1.0	$\mathsf{sim}_{\text{cosine}}$	$\mathsf{mer}_{\text{add}}$
Published date	Date	1.0	0.0	0.0	$\mathsf{sim}_{\text{daydist}}$	N/A
Metaspolit exploits	Boolean	0.3	0.7	0.0	$\mathsf{sim}_{\text{boost}}$	N/A
Linked external resources	Numerical	1.0	0.0	0.0	$\mathsf{sim}_{\text{mult}}$	N/A
Google hits	Numerical	1.0	0.0	0.0	$\mathsf{sim}_{\text{mult}}$	N/A

Impact metrics includes the impact metrics in the CVSS score, namely confidentiality, integrity, and availability impact. These are categorical values where the impact can be `NONE`, `PARTIAL`, or `COMPLETE`. In our implementation, we map these values to numerical scores of 0.0, 0.5, and 1.0 respectively.

Exploitability subscore is the numerical exploitability subscore from the CVSS ranking, which estimates the ease of exploiting the vulnerability.

Authentication (CVSS metric) describes how many times an attacker needs to authenticate before performing an attack. It is a categorical feature with values `NONE`, `SINGLE`, or `MULTIPLE`. In our implementation, we map these values to numerical scores of 1.0, 0.5, and 0.0 respectively.

Access vector (CVSS metric) describes the attack vector for the vulnerability. It is categorical with the value `NETWORK`, `ADJACENT`, `LOCAL`. In our implementation, we map these to numerical values of 1.0, 0.5, and 0.0, respectively.

CWE ID categorizes vulnerabilities according to the type of the vulnerability.

Metasploit exploits is a Boolean value which describes if there is a Metasploit module [12] available for this vulnerability.

Linked external resources is a numerical value which counts the number of linked resources for a specific CVE on NVD.

Google hits is the number of Google search hits a specific CVE-ID has.

4.2 User Requirements Selection

When users start using the system, they should select what makes certain vulnerabilities more relevant to them. This is used to create the explicit user profile $\boldsymbol{u}$ for the recommender. The user profile is constructed by rating the importance of certain information about a vulnerability. The rating should be in the interval of [0,1], and will be used to construct the vector $\boldsymbol{u}$. User requirements can be selected in many ways, in our implementation the user can rate the following properties: confidentiality, integrity, and availability impact; exploit accessibility; access vector; and authentication.

4.3 Similarity and Merge Functions

The choice of similarity and merge functions are described in Table 1. In general, $\mathsf{sim}_{\text{mult}}$ is the most common similarity function, since it maps a higher feature value to a more important vulnerability, by multiplying with some factor. In some cases, the $\mathsf{sim}_{\text{dist}}$ distance similarity function is used instead, since this instead measures how close the feature value is to the user's preference. The Metasploit and publication date features have straightforward similarity functions based on their data type, while the CWE feature requires the use of the $\mathsf{sim}_{\text{cosine}}$ similarity to correctly handle the comparison between CWE vectors.

If we instead look at merge functions, a modified moving average $\mathsf{mer}_{\text{mma}}$ is used for most features, since it provides a simple way to converge towards to user's preference. For CWE, the special $\mathsf{mer}_{\text{add}}$ function needs to be used such that the vector of previously seen CWEs are merged with the newly rated CWE. Finally, features with $\gamma_i = 0$ do not need merge functions, and are marked as N/A in Table 1.

5 Evaluation

In this section we present an initial evaluation of our recommender. For the evaluation, 8 users have been asked to participate. The users are working in the industry, for five different companies, and are people with high security awareness. These people are potential users of such a recommender.

Each user started by selecting their own user profile, with preferences described in Section 4.2. Then, 30 sample CVEs were selected, the CVEs were from different products, years, described different vulnerabilities, and were presented in a random order. The users were asked to rank these CVEs on a scale from 0 to 10, where a higher value indicated higher interest to the user. The users were asked to only consider properties of the CVE itself, rather than the product it affected. To avoid bias from the CVSS base score, this score, as well as the impact and exploitability subscores, were hidden from the user during the evaluation. The users could however see other information in the CVE to make an informed decision.

Then, CVEs were divided into training and test sets using k-fold cross-validation, using $k = 5$. We performed an evaluation where both the user profile

and the training set were used to train the recommender, before generating recommendations. As a comparison, we also compared the results to using the CVSS2 environmental score, with explicit user profiles mapped to impact subscore modifiers. For both cases, the reference ranking was the manual ranking performed by the users.

The RMSE and NDPM [15] values were then calculated between the reference ranking and the recommender output, and between the reference ranking and the CVSS2 environmental score. The metrics can be seen in Table 2. We see that the RMSE values of the recommender system are lower compared to the CVSS environmental score. This indicates that the recommender has higher predictive rating accuracy for all users in comparison to just using the environmental score. The results also indicate higher rank accuracy in comparison to the environmental score based on the NDPM metric, for the majority of test users.

Table 2. RMSE and NDPM of recommender system and CVSS environmental score, relative the reference ranking, for different users. A lower value means higher accuracy.

	RMSE		NDPM	
	Recommender	Environmental	Recommender	Environmental
User 1	0.179	0.222	0.303	0.287
User 2	0.247	0.340	0.195	0.271
User 3	0.200	0.256	0.207	0.333
User 4	0.153	0.296	0.179	0.276
User 5	0.168	0.286	0.294	0.283
User 6	0.138	0.234	0.175	0.228
User 7	0.115	0.224	0.147	0.251
User 8	0.198	0.267	0.349	0.340

6 Conclusions and Future Work

We have defined, implemented and evaluated a recommender system providing severity assessments of vulnerabilities. The recommender system is specialized for vulnerabilities, and is designed to be useful specifically for the context of vulnerability assessment. Recommendations are generated by considering both users' explicit preferences, and by considering their previous interactions with the recommender. The system can be used with a variety of different inputs, and can easily be extended with new features if desired.

The evaluation shows that the system gives better recommendations compared to just using the CVSS environmental score. To be able to tune the parameters for optimized performance, data from more users is needed. However, the results from our evaluation with real users suggests that it is possible to improve the assessment using a recommender system approach. Other possible

future work includes consider negative feedback in the learning phase, which may further improve the results when learning is enabled.

Acknowledgements This work was partially supported by the Swedish Foundation for Strategic Research, grant RIT17-0035, and partially supported by the Wallenberg Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg foundation.

References

1. Aggarwal, C.C.: Recommender Systems. Springer (2016)
2. Chen, L., Sycara, K.: Webmate: A personal agent for browsing and searching. In: Proceedings of the Second International Conference on Autonomous Agents. pp. 132–139. AGENTS '98, ACM (1998)
3. Farris, K.A., Shah, A., Cybenko, G., Ganesan, R., Jajodia, S.: Vulcon: A system for vulnerability prioritization, mitigation, and management. ACM Transactions on Privacy and Security (TOPS) **21**(4) (2018)
4. First: Common vulnerability scoring system v3.0: Specification document, `https://www.first.org/cvss/specification-document`
5. Gadepally, V.N., et al.: Recommender systems for the department of defense and the intelligence community. MIT Lincoln Laboratory (2016)
6. Lee, Y., Shin, S.: Toward semantic assessment of vulnerability severity: A text mining approach. In: 1st International Workshop on EntitY REtrieval (EYRE '18) (2018)
7. Liu, Q., Zhang, Y.: VRSS: A new system for rating and scoring vulnerabilities. Computer Communications **34**, 264–273 (2011)
8. Mell, P.M., et al.: A complete guide to the common vulnerability scoring system version 2.0 (2007), `https://www.nist.gov/publications/complete-guide-common-vulnerability-scoring-system-version-20`
9. Meteren, R.v., Someren, M.v.: Using content-based filtering for recommendation. In: Proceedings of ECML 2000 Workshop: Machine Learning in Information Age. pp. 47–56 (2000)
10. MITRE Corporation: Cve details. `https://www.cvedetails.com/`
11. NIST: National vulnerability database. `https://nvd.nist.gov/`
12. Rapid7: Vulnerability and exploit database. `https://www.rapid7.com/db`
13. Smyth, B.: Case-Based Recommendation, pp. 342–376. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
14. Spanos, G., Sioziou, A., Angelis, L.: WIVSS: A new methodology for scoring information systems vulnerabilities. In: Proceedings of the 17th Panhellenic Conference on Informatics. pp. 83–90. PCI '13, ACM, New York, NY, USA (2013)
15. Yao, Y.Y.: Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science **46**(2) (1995)