Toward Semantic Assessment of Vulnerability Severity: A Text Mining Approach

Yongjae Lee  
KAIST  
Daejeon, Republic of Korea  
ylee.cs@kaist.ac.kr

Seungwon Shin  
KAIST  
Daejeon, Republic of Korea  
claude@kaist.ac.kr

1 INTRODUCTION

A vulnerability is a flaw which exists in either hardware or software systems and can be used to threaten the systems [3]. A vulnerability itself is not a problem unless adversarial one exploits it for the purpose of making the systems fail in terms of security. In other words, the vulnerability can be used by malicious one to violate the systems’ important security properties such as Confidentiality, Integrity, and Availability (CIA) [3]. Therefore, swiftly finding and patching the vulnerabilities are one of the most significant concerns to hardware or software manufacturers, security software vendors, and researchers.

Unfortunately, it is so labor intensive and time consuming to fix the ever increasing vulnerabilities, and thus people want to prioritize vulnerabilities to know much more critical ones. For example, we can decide to fix remotely exploitable vulnerabilities prior to locally exploitable ones because the former can be easily exploited by most attackers. To this end, the vulnerabilities are managed in a systematic manner where they are given a unique id and stored in a central database, NVD (National Vulnerability Database) [14], and their severity is assessed by a severity assessment system.

Common Vulnerabilities and Exposures (CVE) [9, 12] is the most popular vulnerability management scheme, which is operated by NVD. If someone finds and asks to register a newly discovered vulnerability, NVD issues a unique id (CVE identifier) to the vulnerability. Once the vulnerability is registered to NVD, one can check its related information by the issued CVE identifier, which includes a short description, references, a list of affected products, and a severity score. In particular, the severity score is evaluated by Common Vulnerability Scoring System (CVSS) [7], the de facto standard to quantify a vulnerability’s severity. With CVSS score, one can sort out vulnerabilities from highest to lowest, which helps prioritize the vulnerabilities to fix.

However, many researchers have argued that the CVSS does not account for what security experts perceive in the wild [1, 2]. For example, a vulnerability with a low CVSS score is ranked in a higher position in bug bounty programs [13], and the CVSS scores of randomly selected vulnerabilities are not correlated well with severity scores manually evaluated by experts in the security field [8]. More specifically, let’s take Heartbleed as an example. Heartbleed (CVE-2014-0160) is one of the most well-known vulnerabilities that received worldwide attention lately. It is an implementation flaw of OpenSSL, the most used open source encryption library and TLS implementation [15]. This vulnerability can make servers leak confidential data including the encryption key of the servers, which makes the problem much worse, but its CVSS score is just 5.0 out of 10.0 with Medium severity level.
In this paper, we present a semantic approach to assess the severity of vulnerabilities (specifically CVEs) by analyzing descriptions for a CVE. There are many text descriptions for a CVE, such as NVD entries, security blog posts, and manufacturers’ web bulletins, and such text descriptions explain how to exploit the CVE and what kind of damage can be caused if the CVE is exploited by attackers. Since those descriptions commonly present various characteristics about the CVE in human readable natural language, we can glean insightful information from them with the help of Natural Language Processing (NLP) techniques.

To this end, we first collect text descriptions illustrating the characteristics of CVEs from various sources: NVD, blogs, and web bulletins. Next, we extract information from the text descriptions, which includes the type of product where a CVE is found, which version of the product that has the CVE, whether there exists an easy-to-use exploit for the CVE, and so forth. Once such information is extracted, then we apply a ranking method to understand the severity of CVEs clearly. Based on the extracted information, our ranking method first tracks how strongly CVEs are related or similar to one another. This relation can reveal whether characteristics of a CVE are also shared by other CVEs or not. Finally, our ranking method sorts CVEs in order, i.e., a CVE with more common characteristics will be ranked higher. The intuition behind our ranking method is that if characteristics of a CVE are more general, which means that they could be commonly/widely adopted by attackers, then the CVE is more serious. To get the rank of each CVE, we employ the TextRank algorithm [11], and it is an unsupervised ranking algorithm that can summarize and extract important sentences or words within a text.

To initially evaluate our proposed ranking approach, we have collected real CVE data and apply our method to the data. In addition, we compare our ranking results with CVSS score to understand whether our approach can clearly reflect real world opinions. Our initial results show that our approach provides much more realistic (and reasonable) ranking results than CVSS.

2 METHOD

Before we give the detailed explanation on our vulnerability ranking method, we illustrate our system overview in Figure 1. Our method operates in three phases: (1) corpus building, (2) graph building, and (3) vulnerability ranking. Our method is a text-oriented vulnerability ranking, and thus we need a lot of text descriptions about CVEs. Fortunately, NVD compiles related information in a database and allows to access the information freely, we glean CVE descriptions from NVD and built CVE description corpus. After building the corpus, our method generates a vulnerability ranking graph where a vertex represents a CVE. In the graph, vertices are linked to one another when there is a certain relation between two CVEs. We will discuss the relation that the CVEs can have in the following sections. Once we completed the graph building, we run TextRank algorithm on the graph to obtain importance scores of the CVEs, by which the CVEs are sorted.

2.1 Vulnerability ranking

Ranking model Our vulnerability ranking method is based on TextRank [11], which is a graph-based and unsupervised ranking model. TextRank summarizes a text by ranking sentences in the text according to their importance and singling out a set of higher ranked sentences. In the algorithm, a sentence is represented as a vertex, and two sentences are linked to each other if they share similar contents, or words. Although TextRank is an application of Google’s PageRank [5] to text summarization, the two ranking methods are different in that TextRank operates on an undirected graph. This is because, unlike web pages, sentences do not have explicit reference relations. Therefore, TextRank cannot use the graph structure information which denotes that a node votes another one. Instead, TextRank assigns a similarity weight on each link between two nodes, and they exchange the weight when calculating the importance score.

In the ranking graph \( G=(V, E) \), where \( V \) is the set of vertices and \( E \) is the set of edges, let’s assume that there is a vertex \( V_i \). For \( V_i \), let \( \text{In}(V_i) \) and \( \text{Out}(V_i) \) be the set of predecessors of \( V_i \) and the set of successors of \( V_i \), respectively. In addition, if there is a vertex \( V_j \) that belongs to \( \text{In}(V_i) \), the similarity weight between \( V_i \) and \( V_j \) is defined as \( w_{ji} \). Then, the importance score of the vertex \( V_i \) can be computed as below:

\[
\text{IS}(V_i) = (1 - d) + d \times \sum_{V_j \in \text{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in \text{Out}(V_j)} w_{jk}} \text{IS}(V_j)
\]

where \( d \) is a damping factor, which denotes the probability \((1 - d)\) for a random surfer on the graph to jump from a vertex to another one randomly [5]. In this model, the importance score of a vertex is distributed to its successors proportionally to the similarity weight. Therefore, a vertex that is similar to majority of other vertices within the graph, tends to have a higher importance score.
In our vulnerability ranking problem, we believe that a vulnerability that has similar properties with all kinds of vulnerabilities is important and thus needs to be fixed first. This is because, if such a vulnerability is found in a hardware or software product, it means that the vulnerability makes the product have broad attack surfaces. In other words, the product can be attacked in various ways. Here, for two vulnerabilities to have similar properties means that they can be used by similar types of attacks or violate one of the CIA triads alike.

**How to represent a vulnerability in a graph?** In our ranking graph, a vertex is represented as the short description of a CVE, which is less than 10 sentences and recorded for every CVE in NVD. For example, a vertex labeled with CVE-2014-0160 represents the text description presented in Figure 2 which is excerpted from NVD. From content words in the description, we can grasp how the vulnerability can be exploited by attackers and what kind of damages can be caused after the vulnerability is successfully exploited. In other words, in the CVE description, the characteristics of the CVE are expressed in natural language, and the existence of common characteristics between two vertices determines whether they are linked to together or not. Notice that the text description cannot be used directly to be drawn as a vertex and needs to pass the predefined preprocessing steps to be converted to a bag-of-words such as part-of-speech tagging, lemmatization, and so forth.

**How to define similarity between two vulnerabilities?** For two CVEs to share similar characteristics can be defined as having similar words in both of their bags-of-words simultaneously. If the two bags-of-words describing the two CVEs have similar words, we can compute the similarity between them by employing text similarity measures such as Jaccard index or TF-IDF cosine similarity [4]. In this work, we employ Jaccard index [16] as presented in Equation 2, where \( X \) and \( Y \) are the sets of unique words that constitute each bag-of-words of the two vulnerability descriptions.

\[
\text{JaccIndex}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}
\]  

(2)

Using this metric, we can measure how similar two CVEs are. For instance, we present three CVEs and their descriptions in Figure 3 and summarize their similarities in Table 1. Since both CVE-2015-0311 and CVE-2015-7645 are vulnerabilities of Adobe Flash Player and affect the same operating systems (i.e., MS Windows and Apple OS X), they are the most similar vulnerability pair among the three vulnerability pairs. Next, CVE-2015-0311 and CVE-2015-2567 are similar to each other because they have the same property that remote attackers can exploit the vulnerabilities via unknown vectors. In conclusion, CVE-2015-0311 has various factors to be exploited by attackers such as Adobe Flash Player, OS, and unknown attack vectors, and we can conclude it should be handled earlier than others.

### 3 Evaluation

To evaluate our ranking method, we randomly selected 100 CVEs from NVD, which was registered in 2014, and then constructed a small corpus consisting of the 100 CVE descriptions. The descriptions are converted to a bag-of-words, which requires preprocessing that normally consists of sentence boundary detection, stop word removal, and lemmatization. After that, we run TextRank algorithm on the CVE description corpus and obtain the rank of the 100 CVEs. In Table 2, we present the CVEs ranked in the top 10 out of 100. Due to the page limitation, we could not specify the whole description about each of the CVEs but present some keywords in the table.

Taking a look at Table 2, we know that most of the CVEs are related to an X.509 public key certificate problem and can cause some remote attacks such as Man-In-The-Middle (MITM) attacks and Denial of Service (DoS) attacks. In addition, the CVEs are reported to be found in widely used software products including Android, Google Chrome, and Apple OS X. In summary, our ranking method ranks CVEs higher, which (1) are related to a security hole (e.g., certificate verification bypass), (2) are found in popularly used products (e.g., Android), and (3) can cause well-known types of attacks (e.g., MITM and DoS).

![Image](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160)

**Table 2: Descriptive statistics of the corpus data**

<table>
<thead>
<tr>
<th>CVE pair</th>
<th>Jaccard Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2015-0311 &amp; CVE-2015-7645</td>
<td>0.6182</td>
</tr>
<tr>
<td>CVE-2015-0311 &amp; CVE-2015-2567</td>
<td>0.2223</td>
</tr>
<tr>
<td>CVE-2015-7645 &amp; CVE-2015-2567</td>
<td>0.1132</td>
</tr>
</tbody>
</table>

### References

Table 2: Top 10 CVEs generated by our ranking method and their CVSS scores and keywords. CVSS score ranges from 0.0 (not severe) to 10.0 (the most severe) and it is increased by 0.1.

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>CVSS</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2014-5694</td>
<td>5.4</td>
<td>Android, X.509 certificate, SSL, MITM</td>
</tr>
<tr>
<td>CVE-2014-3169</td>
<td>7.5</td>
<td>Use-after-free, DOM, Google Chrome, DoS, remote attack</td>
</tr>
<tr>
<td>CVE-2014-6707</td>
<td>5.4</td>
<td>7Sage LSAT Prep - Proctor, Android, X.509 certificates, SSL, MITM, spoof</td>
</tr>
<tr>
<td>CVE-2014-1741</td>
<td>7.5</td>
<td>Integer overflows, Blink, Google Chrome, remote attackers, DoS</td>
</tr>
<tr>
<td>CVE-2014-1316</td>
<td>5.0</td>
<td>Heimdal, Apple OS X, remote attackers, DoS, Kerberos 5</td>
</tr>
<tr>
<td>CVE-2014-2536</td>
<td>4.3</td>
<td>Multiple directory traversal, McAfee, remote authenticated users</td>
</tr>
<tr>
<td>CVE-2014-2279</td>
<td>6.4</td>
<td>Multiple directory traversal, SeedDM5, remote authenticated users, read arbitrary files, (.dot dot) in the logname parameter</td>
</tr>
<tr>
<td>CVE-2014-5836</td>
<td>5.4</td>
<td>GittiGidyor, Android, X.509 certificates, SSL, MITM, spoof</td>
</tr>
<tr>
<td>CVE-2014-0885</td>
<td>6.8</td>
<td>CSRF, Admin Web UI, IBM Lotus Protector for Mail Security, remote authenticated users, unknown vectors</td>
</tr>
<tr>
<td>CVE-2014-5780</td>
<td>5.4</td>
<td>Bouncy Bill, Android, X.509 certificates, SSL, MITM, spoof</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENT
This research is (in part) based on the work supported by Samsung Research, Samsung Electronics.

REFERENCES

5 CONCLUSION
In this work, we present a semantic way to assess vulnerabilities by examining their textual descriptions from which we can grasp characteristics of the vulnerabilities. We then build the vulnerability ranking graph by representing each vulnerability’s characteristics, which are expressed in natural language, as a node, and run the TextRank algorithm on the graph to obtain the rank of the vulnerabilities. As our future work, we will address the issues discussed in Section 4 to improve the performance of our ranking method to the degree to which security experts and practitioners can agree with our ranking result. To this end, we are going to carry out expert-based performance evaluation for our ranking method, inspired by the existing research work [8].