SOSK Tools for Software Security

Phong Minh Vu[1], Tung Thanh Nguyen[2]
[1]Auburn University, USA
lenniel@auburn.edu

[2]Fulbright University
Vietnam thanhtung, Vietnam
nguyen@fulbright.edu.vn

ABSTRACT

Software security incidents occur everyday and thousands of software security reports are announced each month. Thus, it is difficult for software security researchers, engineers, and other stakeholders to follow software security topics of their interests in real-time. In this paper, we propose, SOSK, a novel tool for this problem. SOSK allows a user to import a collection of software security reports. It pre-processes and extracts the most important keywords from the textual description of the reports. Based on the similarity of embedding vectors of keywords, SOSK can expand and/or refine a keyword set from a much smaller set of their interests userprovided keywords. Thus, SOSK allows users to define any topic of their nterests and retrieve security reports relevant to that topic effectively.Our preliminary evaluation shows that SOSK can expand keywords and retrieve reports relevant to user requests.

Keywords: Software Security, SOSK, Security Reports

1. Introduction

Software security incidents occur everyday. While most are trivial, some high profile incidents could lead to big financial losses or even human lives. To discover new software security vulnerabilities or enhance their software systems with latest security updates, software security researchers, engineers, and other stakeholders have to closely monitor software security topics of their interests.However, this is a difficult task because thousands of software security reports are announced each month. For example, in 2023, there are in total nearly 30,000 ecords newly added to CVE (Common Vulnerability and Exposures) - the most popular database of software security vulnerabilities[3]. The CVE website has asimple function for searching CVE reports. A user can provide one or several keywords to search for records containing those keywords in their textual description[4]. However, it is still difficult for users, because they need to provide all relevant

---

[3] **https://www.cvedetails.com/browse-by-date.php**
[4] **https://cve.mitre.org/cve/search cve list.html**

keywords to a specific topic of their interests.To address this problem, in this paper, we propose SOSK, a novel approach for extracting keywords and tracking security reports. SOSK allows a user to download a database of software security reports.It then pre-processes and extracts the most important keywords from the textual description of the reports. Based on the similarity of embedding vectors of keywords, SOSK can expandand/or refine a keyword set from a much smaller set of user-provided keywords.

Thus, SOSK allows users to define any topic of their interests and retrieve security reports relevant to that topic effectively.

We conducted a preliminary evaluation on a dataset of 112,197 reports fromthe CVE database. After indexing, we used SOSK to define and expand keywords for three topics of common software vulnerabilities: SQL Injection, Cross-site Scripting, and Buffer Overflow. We compared the trends of these three topics found by SOSK with a published report and found SOSK discovered the same trends. Then, we used SOSK to refine a topic of Mobile Devices and retrieved the related CVE reports. The result shows that SOSK can expand keywordsand retrieve reports relevant to our requests. We will perform more extensive evaluation on the performance and usefulness of our approach in the future.

In Section 2, we introduce our approach in detail, including techniques forindexing security reports and and tracking user-defined topics. Section 3 describes our evaluation settings and results. Section 4 presents related work and conclusions appear last.

2. Approach

In this section, we will discuss our approach in detail. SOSK is designed as a document management system with three main components: 1) a document store (DS); 2) a dictionary of keywords (KD); and 3) an index mapping keywords to documents (IX).

The document store DS is a locally stored database containing the securityreports that SOSK has imported and indexed. Each report has two mainattributes: its created date and its textual content. Keywords are stored in a dictionary KD. Each keyword has a textual value (like sql or overflow), a weighting score (like 0.95 or 0.73), and an embedding vector of 768 real values. To simplify the indexing and tracking steps, the textual values of keywords are converted to low-case. The index IX stores for each keyword a hash map containing the ids of documents (security reports) containing that keyword and its positions in each document. Keywords and such positions are extracted from the documents' content while indexing.

Because SOSK is designed for software security domain, in addition to common English words (nouns, verbs, adjectives), the keyword dictionary KD also contains domain-specific names and identifiers such as windows, win32, sql, linux, ls (a command in Linux), http (a network protocol), or svchost (a process inWindows OS). We pre-populated this keyword dictionary with a dictionary of English words extracted from WordNet [3] and a domain specific vocabulary for software documents collected from previous studies [6, 1, 2]. Users can also import user predefined dictionaries to add more domain-specific vocabulary. SOSK has two main functions: indexing and tracking. When a user imports one or (typically) a collection of security reports into SOSK, it will perform indexing on imported documents to update its document store, keyword dictionary,and index. In tracking, SOSK allows a user to define a topic as a list of keywords, expand this list with more related keywords recommended by SOSK,keywords, expand this list with more related keywords recommended by SOSK,

2.1. Indexing
SOSK performs indexing each document (security reports) by low-casing andtokenizing its textual content first. SOSK tries to correct each token using a common for mapperly misspelled words and names provided by MARK [6]. For example, watsapp is corrected as whatsapp (the name of a popular mobile app for texting). Then, for each string token, if it exists in the keyword dictionary KD as a domain specific term (e.g., windows or sql), no stemming is applied. Otherwise,if it is a common English word, Snowball Stemmer is used for stemming. If the token does not exist in the dictionary, it is added as a new keyword.

During this process, SOSK will update the index IX and the keyword dictionary KD for each keyword. The index IX will be updated to keep track the original positions of each keyword in the document under indexing. This is necessary in case users want to identify or highlight the analyzed text on the actual document later. The occurrence count of each keyword and co-occurrence count of each pair of keywords are also updated for calculating the keywords' scores and embedding vectors later.

Embedding vectors According to recent advances in natural language processing and language modeling, the semantic similarity of words can be measured by the distance of their embedding vectors. Thus, SOSK calculates for each keyword an embedding vector of 768 real numbers using Glove [5]. Then, for two keyword a, b with corresponding embedding vectors va, vb, their similarity is define as

$$\rho(a,b) = \frac{|v_a . v_b|}{||v_a||.||v_b||}$$

Keyword scores SOSK calculates for each keyword a score for later ranking. Currently, this score is its inverse document frequency (idf). If the document store DS has in total N documents and a keyword a appears in N_a documents, then its idf score is

$$d(a) = log(N/N_a)$$

2.2. Tracking
In SOSK, a topic is a set of keywords. A document contains one of some of those keywords will be considered to be relevant to that topic. For example, the set $\{\text{buffer, memory, overflow}\}$ could be used to define the topic buffer overflow. A security report contains such words will likely be written about vulnerabilities from buffer overflow.

SOSK calculates the relevance of a document to a topic based on the Vector Space Model. Assume a keyword a occurs n_a, S times in a document S and S has in total n_s word occurrences. The term frequency (tf) of keyword a in document S is defined as

$$t(a,S) = n_{a,S}/n_S$$

The relevance of keyword a to document S is t(a, S).d(a). Then, the relevance of a topic Q to a document S is the total relevance from all its keywords

$$\Gamma(Q,S) = \sum_{a \in Q} t(a,S).d(a)$$

That means, the more keywords are used to define a topic, the higher relevance is calculated for documents containing keywords of that topic. However, a user might not know all the keywords to define a topic of his interest. Therefore, a key function in SOSK is to recommend more keywords to expand a user-defined topic. For a topic Q, SOSK will calculate the list of recommended keywords:

$$\chi(Q) = \{b \mid \exists a \in Q : \rho(a,b) > \theta\}$$

This list contains every keyword b that is sufficiently similar (e.g. higher than a threshold θ) to at least a keyword a in Q. This list will be ranked descending by the keyword scores d(b) and presented to the user. He can add some of those keywords into Q and repeat the recommending step until satisfied.

After the user defines a topic Q, SOSK uses the index IX to find every document S containing at least a keyword in Q. Then, it calculates the relevance score "(Q, S). The user can rank those documents by their relevance scores or by their created dates (e.g., to focus on more recent security reports first).

3. Evaluation

In this section, we report a preliminary evaluation on a dataset of 112,197 security reports from the Common Vulnerability and Exposures (CVE) database,dated from 1999 to 2016.

We first downloaded this dataset and imported it into SOSK for indexing.Then, we pre-loaded the embedding vectors for all applicable keywords (common English words and technical names) in SOSK with the pre-trained embedding vectors from the English Wikipedia dump. After that, the co-occurrence counts from SOSK indexing step were used to fine-tuning those embedding vectors.

That task was needed to produce high quality embedding vectors for SOSK.Because the CVE dataset contains only very short and descriptive documents,it is not sufficient to learn high quality embedding vectors. Using pre-trained vectors and fine-tuned them using a domain-specific dataset reinforces the embeddingvectors of common words that appear in both corpora (Wikipedia and CVE), while the technical or domain-specific keywords in the CVE dataset would have more significant embedding vectors.

| Topic	Seeded Keywords	Expanded Keywords
SQL Injection	sql, inject, vulnerability, php	sql, inject, vulnerability, php vulnerable,code, arbitrary, command, injection, attacker, attack, remote
Cross-site Scripting	vulnerability, script, cross, site, xss	vulnerability, script, cross, site, xss, website, server, remote, attacker, web, attack, directory, credential
Buffer Overflow	overflow, buffer, stack, function	overflow, buffer, stack, function, component,method, memory, implement, implementation, heap,

Table 1. Examples of Keywords Expansion

### 3.1. Studying previously reported security topics

Neuhaus and Zimmermann published a study of trend analysis for security problems in the CVE dataset in 2009 [4]. However, as per technological advance, the security topics may have changed after that. Therefore we picked three major topics SQL Injection, Cross-site Scripting and Buffer Overflow from their study and used our tool on more recent CVE reports to answer two questions: 1) How did their trends changed after 2009; and 2) What are the actual descriptions of the security problems for those topics?

The results are shown in Table 1, Table 2, and Figure 1. In Table 1, the column Seeded Keywords lists the keywords used by Neuhaus and Zimmermann in their study. We fed them into SOSK and SOSK recommended new keywords in the right column (with a similarity threshold of 0.9).

How did security trends changed after 2009? As shown in Figure 1, even though we used a different approach, the shape of the trends from 2000 to 2009 reported by SOSK is similar to the findings reported in [4]. Then, after 2009 topic SQL Injection had a sharp decline in the total number of reports, while the reports for Cross-site Scripting fluctuated for the next few years. While the percentage of this type of attack was reduced dramatically during the those years,its actual number remained steady. Finally, after the gradual drop observed by Neuhaus and Zimmermann, Buffer Overflow attacks kept declining slowly only to increased again during 2015 and 2016. These numbers could indicate that SQL Injection is not a popular breach around 2009 - 2016, but Cross-site Scripting and Buffer Overflow are still prevalent.

(a) Total reports in CVE

(b) Buffer Overflow

(c) SQL Injection

(d) Cross-site Scripting
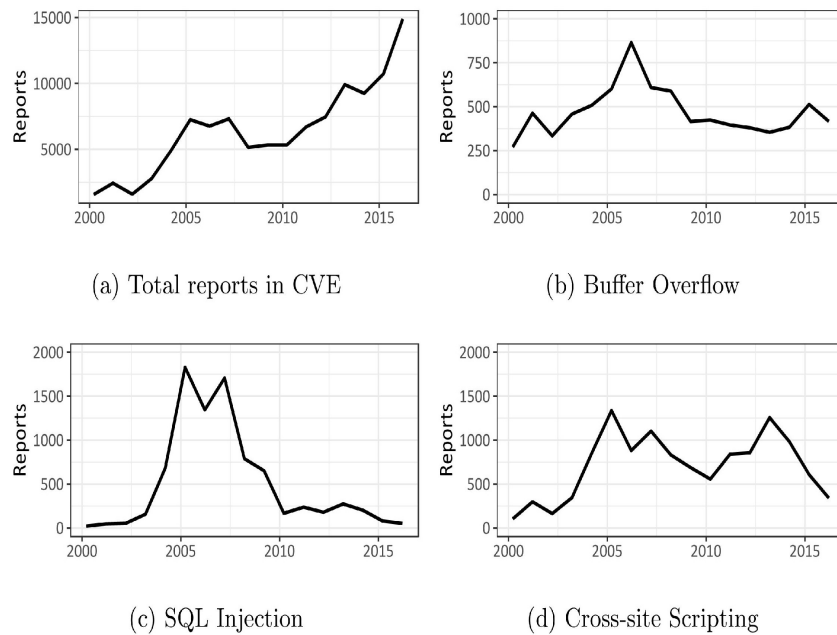
Figure 1. Trends analysis of three topics in CVE Dataset
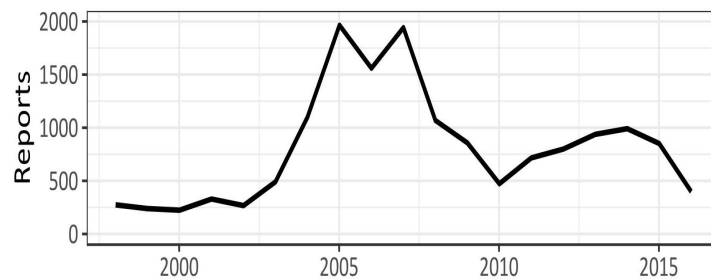
Figure 2. Trends analysis of topic Mobile Devices in CVE

What are the actual descriptions of the security reports? To answer the second research question, we printed out the descriptions matched keywords of the studying topics. Table 2 shows some examples. More can be viewed completely along with the artifacts of this paper[5.] For SQL Injection, we can see the the results listed the platforms or method used by attackers to inject arbitrary SQL code. For Cross-site Scripting and Buffer Overflow, we can see the vulnerabilities in different modules, software products, and components.

---

[5] https://goo.gl/gNnk5i

These results are informative enough to give users a general idea of what to be expected from each particular security problem, while saving the time needed to investigate directly in the CVE website.

3.2. Exploring new topic
One topic not reported in the original study of Neuhaus and Zimmermann was security risk in mobile devices. With CVE data at the time (from 1999 to 2009),the LDA method could not identify the rising of security reports related to mobile devices despite of a booming first year of iPhone being introduced to the world in 2007-2008. It is also not a pre-defined security topic at cvedetails.com.

Topic	Examples of Topic Description
SQL Injection	timecard cms allow remote attacker to execute arbitrary sql command execute arbitrary sql command via the execute query array myphpnuke allow remote attacker to execute arbitrary php code php in mybulletinboard allow remote attacker to execute arbitrary sql statement via the fidme sql injection vulnerability in aspwebalbum allow remote attacker to execute arbitrary sql statement via the username field
Cross-site Scripting	site script vulnerability in the search function in the web management interface buymyscript lyric script allow remote attacker to inject arbitrary web script site script vulnerability in the calendar application yahoo answer clone allow remote attacker to inject arbitrary web script site script attack by upload
Buffer Overflow	buffer overflow in the urarlib get function multiple buffer overflow in the rtconfigload function heap buffer overflow in function pnmtoimage base buffer overflow in the imap server component stack buffer overflow in vshttpd

Table 1. Examples of Keywords Expansion

However, in our keyword dictionary, the keyword mobile is among the highest ranked. To verify if this missing topic was truly important, we have applied our tool and the trend analysis method of MARK [6] (detecting abnormal spikes of Moving Average and Ratio of Difference to Standard Deviation) to see if it had a noticeable peak at some point in time, or somehow relevant at all. The results (Figure 2) suggest that reports containing mobile and related keywords (android,ios, iphone, samsung) had a peak in 2005 and 2007, with 1965 and 1942 reports, respectively. After that, the yearly number of reports still remained high. This result has proven that this topic was overlooked by the LDA method and should have been a new important topic since 2005.

Further investigating reports returned by SOSK, we found that some of the most common vulnerabilities of mobile devices were "commonly available simple gps location", or "wi-fi spot configuration software", or "bypass intended permission restriction". More specific reports to the devices included: "samsung galaxy s4 through s7 device", "softback panasonic 3g handset", or "ip phone 1140e". Such information provides a coherent picture of which devices are vulnerable to which attacks, something specific to mobile devices for having various platforms.

4. Related work

Neuhaus and Zimmermann analyzed 39,393 vulnerability reports in the Common Vulnerability and Exposures (CVE) database from 1999 to 2009 [4]. They applied topic modeling (LDA) on descriptions of those CVE reports to find prevalent vulnerability types and new trends. In this work, we demonstrated that using a keyword-based approach can also find the same topics and trends.

MARK [6] is a keyword-based approach to discovering topics and trends on user reviews of mobile apps. In this paper, we reused its spell checker and domainspecific dictionary. However,

due to domain differences (reviews of mobile apps written by end-users vs security reports written by professionals), we had to enrich our keyword dictionary with more security-related terms and names.In addition, we also fine-tuned the pre-trained embedding vectors with textual contents of CVE reports to make it more relevant to the domain of software security.

Yitagesu et al [7] developed unsupervised methods to label and extract important vulnerability concepts from the textual descriptions of software vulnerability reports. Their approach uses deep learning models (neural architecture that learns the Part-of-Speech (POS) of words and phrases and auto-encoder to encode syntactic similarities of paths in parse trees). The results are six fixed
types of vulnerability concepts. In contrast, our tool focuses on user-defined opics which could be more flexible and suitable to users' interests. In addition, SOSK uses Glove [5], a lightweight approach for learning embedding vectors and thus, can index a large collection of data and run locally.

5. Conclusions

In this paper, we introduce SOSK, a tool for tracking software security reports by keywords. A user can collect a database of software security reports (typically text-based) and import it into SOSK. While importing, SOSK pre-processes and extracts the most important keywords from the textual description of the reports. Using pre-trained and fine-tuned embedding vectors of keywords, SOSK
can compute their similarity and is able to expand and/or refine a keyword set from a much smaller set of user-provided keywords. Our preliminary evaluation shows that SOSK can enable users to define several topics of their interests and retrieve security reports relevant to that topic effectively.

References

[1] Chen, N., Lin, J., Hoi, S. C., Xiao, X., and Zhang, B. (2014). AR-Miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the International Conference on Software Engineering (ICSE)*.

[2] Di Sorbo, A., Panichella, S., Visaggio, C. A., Di Penta, M., Canfora, G., and Gall, H. (2016). DECA: Development emails content analyzer. In *Proceedings of the International Conference on Software Engineering (ICSE)*.

[3] Miller, G. (1998). *WordNet: An electronic lexical database.* MIT Press.

[4] Neuhaus, S., and Zimmermann, T. (2010). Security trend analysis with CVE topic models. In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*.

[5] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[6] Vu, P. M., Nguyen, T. T., Pham, H. V., and Nguyen, T. T. (2015). Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Proceedings of the International Conference on Automated Software Engineering (ASE)*.

[7] Yitagesu, S., Xing, Z., Zhang, X., Feng, Z., Li, X., and Han, L. (2023). Extraction of phrase-based concepts in vulnerability descriptions through unsupervised labeling. *ACM Transactions on Software Engineering and Methodology (TOSEM), 32*(5).