ATLANTIS PRESS

Advances in Engineering Research, volume 185

2nd International Conference on Electrical and Electronic Engineering (EEE 2019)

An Automatic Approach for Scoring Vulnerabilities in Risk Assessment

Ying-jun ZHANG[1], Peng LIAO[2], Ke-zhen HUANG[1] and Yu-ling LIU[1]

[1]Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, China

[2]NARI GROUP CORPORATION, China

Keywords: Risk assessment, Vulnerability.

Abstract. Risk assessment is vital to an information system. Current approaches usually rely on human experts' experience to give scores to vulnerabilities in the information system and synthesize the scores to form the whole risk score of the system. The experts give such scores by understanding a vulnerability in terms of the difficulties of exploiting and impacts of being exploited. However, such scores are mostly dependent on the human's experiences, which makes the results are not consistent when different analysts give the scores. In this paper, we design an approach to give such scores without any need of human experiments. Specifically, we acknowledge a vulnerability, especially the impact of the vulnerability, by searching it online. From the results, we are able to know its popularity and impacts using machine learning algorithms. To avoid the redundant searched results, we utilize an n-gram based approach to eliminate them. We also give examples in the evaluation to show how our approach work. Results show that our approach is able to give such scores without any need on human's experiences, in the result of giving unbiased scores.

Introduction

As we know, an information system needs risk assessment performed regularly to identify and analyze potential events and judge the risk of being attacked [1]. Such assessment will also expose vulnerabilities of the whole system, which could further be addressed (e.g., patched) to enhance the security of the system. In this process, human analysts usually score the vulnerabilities and also give an overall risk value to the system, telling the owner the possibility of being attacked and being successfully exploited [2]. The lower the score, the more risks the system will face. The value of the risk is not fixed. This is mainly because that the risk assessment should not be performed for only one time. Instead, it should be carried out regularly. The score of risk also changes with time due to the understanding of the vulnerabilities changes and the patches could be developed and executed on the system.

Currently, the risk assessment is highly dependent on human analysts. Especially, they will give score to a vulnerability found on the system to stand for the risk brought by the vulnerability. As mentioned before, the score should be changed when time passes by or environment changed, since the understanding and the capability of leveraging the vulnerability changes. For example, a vulnerability will have more impacts if it is discussed a lot on a web (i.e., known by more attackers). Also, the exploits of the vulnerability could be open to public. As a result, more attacks could be performed with minimal efforts on studying the vulnerability. Even for an attacker lacking of experience, he could utilize the exploit code published online to finish the attack. Sometimes, a scanner targeting the vulnerability would be open to the public so that the attacker could easily scan the whole network to find the vulnerable systems. As a result, the score of assessing the vulnerability should be changed when considering the extra information released online, which finally impacts the risk scores of the whole system.

However, a security analyst may not notice such changes that happen and continuously evolve online. Thus, he usually utilizes only the static features of a vulnerability to give the score. For example, he could use the type of the vulnerability. If it is a memory overflow vulnerability, arbitrary code could be executed so that the risk score could be higher than an information disclosure vulnerability. However, he may not know what kind of information could be leaked by the

information disclosure vulnerability. He may not understand the difficulties of exploiting the two vulnerabilities. So the real impact of the two vulnerabilities may not be considered. Another problem of using static features is that this method does not consider the combinations of exploiting several vulnerabilities which could be introduced on web. In this way, some vulnerabilities looking un-harmful may be of great importance in an attack.

Based on the discussion above, the analyst should always keep his eyes on web to know the details of vulnerabilities. However, this is quite challenging. A human analyst may not have enough time to achieve this goal. For an inexperienced analyst, he might neither have the ability to understand the details about all the vulnerabilities. Even if the analyst could search online when meeting every new vulnerability, lots of time could be spent on searching and comparing the results of the searched results.

To address this problem, our idea is to design an automatic approach to give risk scores to vulnerabilities, which dynamically evolves with the discussions on the vulnerabilities online. In particular, when a new vulnerability is met in the process of risk assessment, our approach first searches the vulnerability online. From the results, we could roughly understand how popular the vulnerability is. To further ensure the impact of the vulnerability, we should calculate the redundancy among the web pages and decrease the impact of it. This is due to that the repeated posts of the vulnerability may not be of great value for attacks to understand and exploit the vulnerability. The more diverse of the discussion, the more impacts could be of the vulnerability. Moreover, we further calculate the scores using the features of the web pages discussing the vulnerability. In this process, machine learning algorithms are leverage to generate a model for scoring risks. To train the model, labelled data by human experts are utilized. In the evaluation, we give an example of how to use our approach and compare a popular vulnerability with an unpopular one. Even if they look quite similar from CVE descriptions, their popularities are quite different, which makes the risks of the two vulnerabilities be different.

Roadmap. The rest of the paper is organized as follows. Section 2 provides the background and related work of risk assessment and vulnerabilities. Section 3 gives the details of our design and Section 4 gives concrete examples which include two real vulnerabilities. In the end, we conclude our work in Section 5.

Background and Related Work

There are many vulnerability scoring system, which are put forward by different organizations, for example microsoft, USCERT, NVD, oracle and so on. So there are a variety of security vulnerability rating standards, which brings uncertain factors to risk evaluation, emergency response. CVSS can solve the problems above. CVSS[9] is the common vulnerability scoring system, which is a framework for rating the severity of security vulnerabilities in software. The CVSS uses an algorithm to determine three severity rating scores: base, temporal and environmental. And the scores are numeric, which range from 0.0 to 10.0. And some works improved the CVSS framework. For example, in [2], presents a vulnerability scoring mechanism based on CVSS by analyzing advantages and disadvantages of CVSS. However, we find that the score from CVSS is too centralized and usually the weights in CVSS are based on experience.

Besides, ASVA[3] is an automatic security vulnerability assessment framework, which can automatically apply any quantitative vulnerability assessment standards (QVAS) to special vulnerability databases. ASVA obtain values of metrics of a QVAS with new features of Text mining, then assign these values to a formula of QVAS and finally compute the severity values of the vulnerabilities. In [4], presents a novel method to implement multi-dimension evaluation on vulnerability, which is based on influence scope and patch fixed situation. Besides, researchers focus on the vulnerability analysis and assessment for smart phone, tablet operating systems[5] and IoT[6]. All these methods are mostly based on the experience. So we propose a new approach to score vulnerability which is based on searching results.

Approach

The basic idea our approach is shown in Figure 1. When a vulnerability is found on a system, our approach first searches it online. From the search results, we first do some preparation on them such as calculating redundant information. In the second step, we extract features of the searched results and feed them to the model for prediction. The model is trained using previous knowledge labelled by human experts. In this way, the model could output a risk score for the vulnerability instead of the human, which could catch the continuous evolvement of the vulnerabilities, freeing the human efforts in the process of risk assessment. Below we elaborate the details of each module.

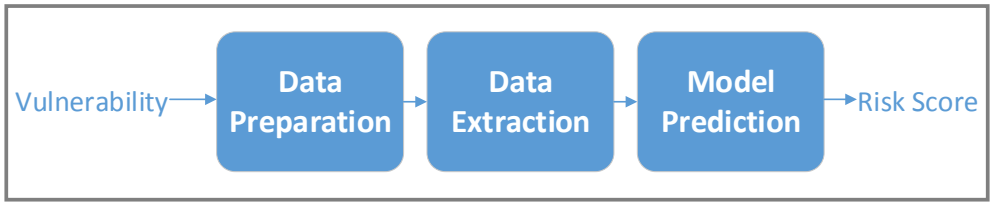
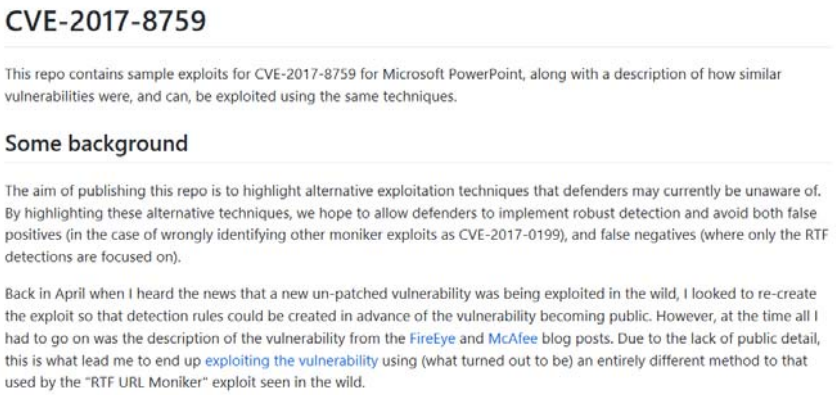


Figure 1. Overview of our approach

Data Preparationf

In this step, we first search a vulnerability online and collect the searched results. Sometimes, the results contain lots of web pages. We need to group similar ones together. For example, when we search CVE-2019-9601 online, the returned results contain 120 web pages (as shown in Section 4). We find that some of them are only repost. Thus, we calculate the redundancy and pick the unique web pages.

To achieve this goal, for any two web pages, we could compare their similarity. Here, we leverage n-gram similarity approach [7]. Particularly, for each web page, we split the contents in it using n words. Then we slide the window of the n words and get another group of n words. For each group of the n words, we calculate the hash value of them. Then for each web page, we could get a set of hash values which represents the n-word groups in the web page. In this way, we could compare the two pages by calculating the number of same hash values between them. Suppose the number of the same hashes is n_s, and the number of the unique hashes in the two web pages are n_{all}. The similarity score is calculated as $s=n_s/n_{all}$. If the score is higher than a threshold, we say the two web pages are similar. We will put them into the same group so that redundant information will not be calculated repeatedly.



CVE-2017-8759

This repo contains sample exploits for CVE-2017-8759 for Microsoft PowerPoint, along with a description of how similar vulnerabilities were, and can, be exploited using the same techniques.

Some background

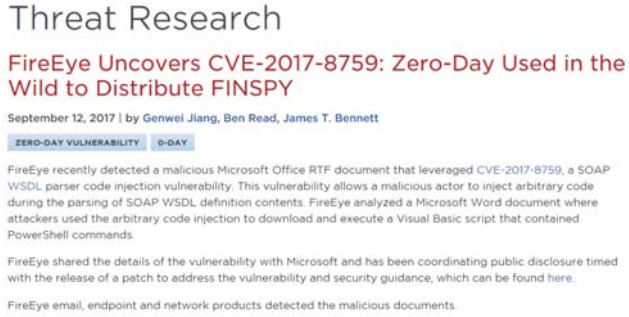
The aim of publishing this repo is to highlight alternative exploitation techniques that defenders may currently be unaware of. By highlighting these alternative techniques, we hope to allow defenders to implement robust detection and avoid both false positives (in the case of wrongly identifying other moniker exploits as CVE-2017-0199), and false negatives (where only the RTF detections are focused on).

Back in April when I heard the news that a new un-patched vulnerability was being exploited in the wild, I looked to re-create the exploit so that detection rules could be created in advance of the vulnerability becoming public. However, at the time all I had to go on was the description of the vulnerability from the FireEye and McAfee blog posts. Due to the lack of public detail, this is what lead me to end up exploiting the vulnerability using (what turned out to be) an entirely different method to that used by the "RTF URL Moniker" exploit seen in the wild.

Figure 2. An example of analysis on a CVE [https://github.com/nccgroup/CVE-2017-8759/]

For example, in the webpage when searching "CVE-2017-8759" as shown in Figure 2, we use every n words to split the contents. Suppose we choose n=5 here. The first group of 5 words are "This repo contains sample exploits". Then we calculate the hash value of the five words. Here we use MD5 to calculate the value, which equals "bd105d538aa2931bd711f3fd5ac81929". Later, we slide the window of 5 words and get another group of words (i.e., "repo contains sample exploits for"). Similarity, we calculate the hash. In this way, the web page could be expressed as a set of the hashes.

Then for another web page (as shown in Figure 3), we just compare the similarity using the two sets. From the results, we can see that the two webpages are not the same.


Threat Research

FireEye Uncovers CVE-2017-8759: Zero-Day Used in the Wild to Distribute FINSPY

September 12, 2017 | by Genwei Jiang, Ben Read, James T. Bennett

ZERO-DAY VULNERABILITY 0-DAY

FireEye recently detected a malicious Microsoft Office RTF document that leveraged CVE-2017-8759, a SOAP WSDL parser code injection vulnerability. This vulnerability allows a malicious actor to inject arbitrary code during the parsing of SOAP WSDL definition contents. FireEye analyzed a Microsoft Word document where attackers used the arbitrary code injection to download and execute a Visual Basic script that contained PowerShell commands.

FireEye shared the details of the vulnerability with Microsoft and has been coordinating public disclosure timed with the release of a patch to address the vulnerability and security guidance, which can be found here.

FireEye email, endpoint and network products detected the malicious documents.

Figure 3. The other webpage when searching CVE-2017-8759
[https://www.fireeye.com/blog/threat-research/2017/09/zero-day-used-to-distribute-finspy.html]

In this way, for each vulnerability, we could get the web pages and the similarity between each two webpages. Then we group similar webpages and get the number of groups. In this way, we have prepared the dataset for a vulnerability.

Feature Extraction

In this step, we extract features for a vulnerability, which could be utilized to predict the risk score. For each vulnerability, we collect the following features as shown in Table 1. In the table n_p stands for the number of web pages returned from searching. For example, if we use google as a search engine, the number of returned web pages is 22,200, which shows that the vulnerability is really popular and attackers could gain lots of information from the results. n_{up} is the number of groups of unique web pages, which means that we group similar web pages using the n-gram similarity score mentioned above. This feature could also illustrate the popularity of studies on the vulnerability. n_{exp} shows the number of groups containing exploits. If an exploit is given, the vulnerability is highly possible to be utilized for attacking. We get this value by checking whether a web page contains links to an attachment (in .c, .zip, or .rar type). This may have some false positives, but it could give a rough idea on whether the vulnerability could be exploited. Sometimes, the exploits are not directly given. The author of the web page may give some instructions in the page to show the instructions to generate exploits. So we use n_{ins} to characterize the number of groups containing instructions to generate exploits. We count a web page as the one containing instructions if some keywords (e.g., “exploits”, “step”, “generation”, “eip”, “execution flow”, etc.) and a good number of figures appear in the web pages. We also measure the average length of the webpages in groups (referred to as ave_{len}), since a detailed webpage contains more information for attackers to generation exploits or to know how to utilize the vulnerability. Using these features, we are able to characterize a vulnerability and score the risk.

Table 1. Features of web pages for scoring the risk of a vulnerability

Features	Descriptions
n_p	Number of web pages returned from searching
n_{up}	Number of groups of unique web pages
n_{exp}	Number of groups containing exploits
n_{ins}	Number of groups containing instructions to generate exploits
ave_{len}	Average length of the webpages in groups

Model Training and Prediction

Our idea to score a vulnerability is to utilize a trained model for prediction. Such model is trained using labelled data (e.g., by human experts), and could be later for prediction. In the training process, we give a number of vulnerabilities (e.g., CVEs) and let human experts score their risks. For each vulnerability, we search it online and generate the features (n_p, n_{up}, n_{exp}, n_{ins}, ave_{len}). Then we use a

machine learning model (e.g., SVM [8]) to train the model. The training approach is standard. In this way, we could get the parameters and finally form the model. At last, for a newly given vulnerability, we could use the model to predict the risk of the vulnerability. The more the data, the more accurate of the model. If the model is not accurate enough, we could use more data for training until human experts are satisfied with the model prediction results.

Evaluation

In this section, we use a case study to illustrate our approach. For a given vulnerability CVE-2019-9601 which is a newly appeared vulnerability for android, we searched it online. The number of returned pages is 120. Then from the searched results, we could combine some similar web pages. For example, Figure 4 and Figure 5 show the similar results of the vulnerability, and we should combine them into the same group. Also, we can find the web page contains a link to https://www.exploit-db.com/exploits/46380, which actually includes an exploit.

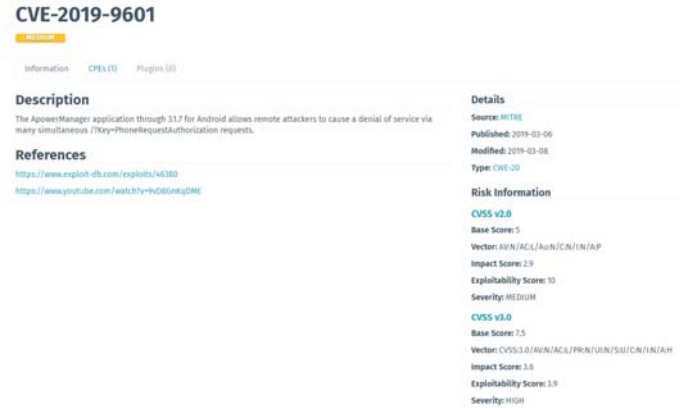


Figure 4. A webpage showing CVE-2019-9601 (https://www.tenable.com/cve/CVE-2019-9601)

Figure 5. A similar webpage showing CVE-2019-9601 (https://nvd.nist.gov/vuln/detail/CVE-2019-9601)

For some vulnerabilities, the results are very few, especially the newly appeared vulnerabilities. For example, the vulnerability CVE-2019-9632 is newly appeared in March 2019. The number of pages returned by Google is 27, most of them only contain the brief information of the vulnerability. Also there is no exploit is given by the web pages, which means that the vulnerability is hard to be utilized by attackers while compared with previous mentioned vulnerabilities. Therefore, the risk score should not be high. By this approach, we are able to score the risk of a vulnerability automatically.

Conclusion

In this paper, to solve the problem of relying on human experts' experiences to score risks of vulnerabilities, we design an automatic approach, which only uses the searched results of the

vulnerabilities by popular search engines. By extracting features from the searched results and training the model using labelled data by human experts, we are able to utilize the model for future risk prediction. In this way, human experiences would not be very necessary in this step of risk assessment. Also, our approach is much more quick in prediction and is able to catch the continuous evolvement of the vulnerabilities. If a new exploit is generated for a vulnerability and opened to public, our approach could quickly catch it and change the risk score. In this way, our approach could be more accurate to reflect the real risk of the vulnerabilities.

Acknowledgement

This research was financially supported by the State Grid Science and Technology Project: Research on Key Technologies of Security Vulnerabilities and Risk Experiment Capability Improvement in Dispatching Automation Software.

References

[1] Rausand, M. (2013). "Chapter 1: Introduction". Risk Assessment: Theory, Methods, and Applications. John Wiley & Sons. pp. 1–28. ISBN 9780470637647.

[2] R.Wang, L Gao, D Sun, An improved CVSS-based vulnerability scoring mechanism, Third International Conference on Multimedia Information Networking and Security, 2011.

[3] T Wen, Y Zhang, Y Dong, G Yang, A Noval Automatic Severity Vulnerability Assessment Framework, Journal of Communications, Vol.10, No.5, May 2015.

[4] WC Jia, RG Hu, YY Wang, XM Li, A Method for Multi-dimension Evaluation on Vulnerability Based on Program Dependence Graph, Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC), 2015.

[5] A Coskun, U Bostanci, Vulnerability Analysis of Smart Phone and Tablet Operating Systems, Tehnicki vjesnik, 2018.

[6] AU Rehman, I Gondal, J Kamruzzuman, A Jolfaei, Vulnerability modelling for hybrid IT system, IEEE International Conference on Industrial Applications, 2019.

[7] G Kondrak, N-Gram Similarity and Distance, International Symposium on String Processing and Information Retrieval, pp.115-126, 2005.

[8] C Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297.

[9] Margaret Rouse, CVSS (Common Vulnerability Scoring System), https://searchsecurity.techtarget.com/definition/CVSS-Common-Vulnerability-Scoring-System