Poster: OS-Aware Vulnerability Prioritization via Differential Severity Analysis

PUBLISHED PAPER

Title: OS-Aware Vulnerability Prioritization via Differential Severity Analysis
Authors: Qiushi Wu, Yue Xiao, Kangjie Lu
Email: {xiaoyue}@indiana.edu, {wu000273,kjlu}@umn.edu
Date: August 10-12, 2022
Venue: Proceedings of the 2022 USENIX Security Symposium(USENIX'22)

ABSTRACT

The Linux kernel is quickly evolving and extensively customized. This results in thousands of versions and derivatives. Unfortunately, the Linux kernel is quite vulnerable. Each year, thousands of bugs are reported, and hundreds of them are severe vulnerabilities. Given the limited resources, the kernel maintainers have to prioritize patching the more severe vulnerabilities. In practice, the Common Vulnerability Scoring System (CVSS) [1] has become the standard for characterizing vulnerability severity. However, a fundamental problem exists when CVSS meets Linux—it is used in a "one for all" manner. The severity of a Linux vulnerability is assessed for only the mainstream Linux, and all affected versions and derivatives will simply honor and reuse the CVSS score. Such an undistinguished CVSS usage results in underestimation or overestimation of severity, which further results in delayed and ignored patching or wastes of the precious resources. In this paper, we propose OS-aware vulnerability prioritization (namelyDIFFCVSS), which employs differential severity analysis for vulnerabilities. Specifically, given a severity-assessed vulnerability, as well as the mainstream version and a target version of Linux, DIFFCVSS employs multiple new techniques based on static program analysis and natural lan-guage processing to differentially identify whether the vulnerability manifests a higher or lower severity in the target version. A unique strength of this approach is that it transforms the challenging and laborious CVSS calculation into automatable differential analysis. We implement DIFFCVSS and apply it to the mainstream Linux and downstream Android systems. The evaluation and user-study results show that DIFFCVSS is able to precisely perform the differential severity analysis, and offers a precise and effective way to identify vulnerabilities that deserve a severity reevaluation.

[1] INDIANA UNIVERSITY
[2] UNIVERSITY OF MINNESOTA

Poster: OS-Aware Vulnerability Prioritization via Differential Severity Analysis

Qiushi Wu*[2], Yue Xiao*[1], Kangjie Lu[2]
(* co-first authors)

Abstract

The severity of a Linux vulnerability is assessed for only the mainstream Linux, and most affected versions and derivatives will simply honor and reuse the CVSS score. Such an undistinguished CVSS usage results in underestimation or overestimation of severity, which further results in delayed and ignored patching or wastes of the precious resources. In this paper, we propose OS-aware vulnerability prioritization (namely DIFFCVSS), which employs differential severity analysis for vulnerabilities.

Introduction

A fundamental problem arises when CVSS meets Linux — it is used in an *"one for all"* manner. When a bug reporter requests a CVE for a vulnerability, the CVE maintainers assign a (single) CVSS score for it, typically based on the mainstream Linux. All affected versions and some derivatives will then simply honor the assigned CVSS score for prioritizing their patches. This is understandable because assigning the CVSS score is quite laborious and requires expertise. Maintainers of small derivatives may not afford the reevaluation for all of their system. However, this results in both severity overestimation which wastes maintenance resources and severity underestimation which delays the patching of critical vulnerabilities and incurs critical threats.

To address it, we present DIFFCVSS, a system that can automatically and precisely determine if a vulnerability will have a higher or lower severity in a different OS. DIFFCVSS incorporates multiple new techniques, such as automatically identifying the call-chain for a vulnerability and mapping kernel functions to CVSS metrics, to ensure precision and effectiveness.

Methodology

1. Mapping Metrics to Functions
2. Recognize Vulnerability Artifact
3. Identify Vulnerability Call-Chain
4. Differential Severity Analysis

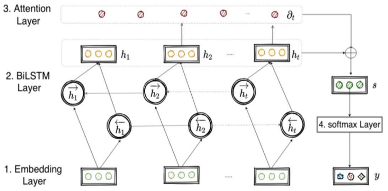


Figure 3: BiLSTM+Attention model

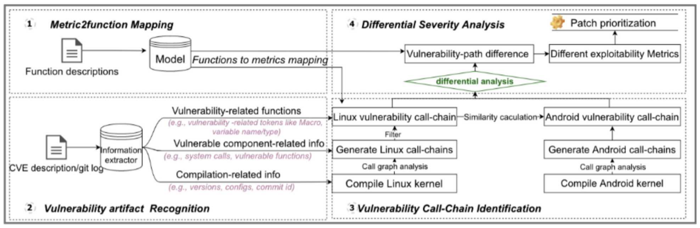
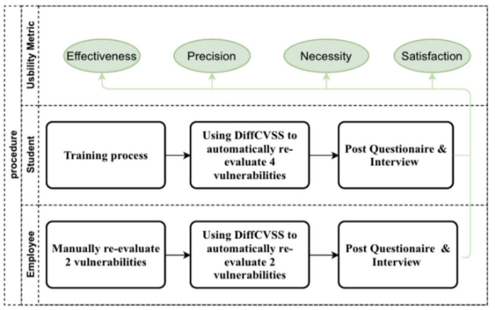

1 Metric2function Mapping
Function descriptions
Model
Functions to metrics mapping
4 Differential Severity Analysis
Patch prioritization
Vulnerability-path difference
Different exploitability Metrics
differential analysis
Vulnerability-related functions
(e.g., vulnerability -related tokens like Macro, variable name/type)
Vulnerable component-related info
(e.g., system calls, vulnerable functions)
Compilation-related info
(e.g., versions, configs, commit id)
CVE description/git log
Information extractor
Linux vulnerability call-chain
Similarity caculation
Android vulnerability call-chain
Filter
Generate Linux call-chains
Generate Android call-chains
Call graph analysis
Call graph analysis
Compile Linux kernel
Compile Android kernel
2 Vulnerability artifact Recognition
3 Vulnerability Call-Chain Identification

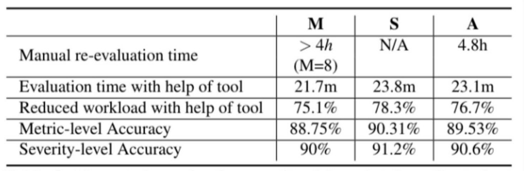
Figure 1: An overview of DIFFCVSS.

Evaluation

Q1: How efficient is DIFFCVSS in reducing maintainer workload?
Q2: How accurate is DIFFCVSS in re-evaluating vulnerability?
Q3: How usable is DIFFCVSS in practice?


Usbility Metric
Effectiveness
Precision
Necessity
Satisfaction
procedure
Student
Training process
Using DiffCVSS to automatically re-evaluate 4 vulnerabilities
Post Questionaire & Interview
Employee
Manually re-evaluate 2 vulnerabilities
Using DiffCVSS to automatically re-evaluate 2 vulnerabilities
Post Questionaire & Interview

- DIFFCVSS can save 91.98% of time, reduce 76.7% of workload
- DIFFCVSS achieves an accuracy of 89.53% in Metric-level and 90.6% in Severity-level on average.

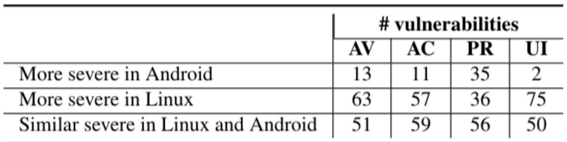


	M	S	A
Manual re-evaluation time	> 4h (M=8)	N/A	4.8h
Evaluation time with help of tool	21.7m	23.8m	23.1m
Reduced workload with help of tool	75.1%	78.3%	76.7%
Metric-level Accuracy	88.75%	90.31%	89.53%
Severity-level Accuracy	90%	91.2%	90.6%

Table 9: User study evaluation results. M=maintainer, S=student, A=average

Findings

- 110 vulnerabilities that have different severity across Android and Linux.
- 18 of them have a higher severity and should be reevaluated per OS to avoid delayed patching.



	# vulnerabilities			
	AV	AC	PR	UI
More severe in Android	13	11	35	2
More severe in Linux	63	57	36	75
Similar severe in Linux and Android	51	59	56	50

Table 7: Cross-OS vulnerability exploitability metric difference between Linux and Android.

Conclusions

- The "one for all" strategy results in both severity overestimation and severity underestimation.
- We purpose a system that can automatically and precisely determine if a vulnerability will have a higher or lower severity in a different OS.