Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.85
DOI: https://doi.org/10.24867/15BE23Cvetanovic

SISTEM ZA PRAĆENJE RANJIVOSTI U SOFTVERU

SOFTWARE VULNERABILITY MONITORING SYSTEM

Vladimir Cvetanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast –ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je opisan sistem za praćenje ranjivosti u softveru. Objašnjeni su osnovni pojmovi i mehanizmi uz pomoć kojih je moguće javno identifikovanje ranjivosti u softveru. Opisan je model sistema i koraci u radu sistema. Na kraju su data zaključna razmatranja i pravci kojima bi dalji razvoj ovog alata mogao da ide.*

Ključne reči: *NVD, CVSS, CVE, ranjivost, zavisnost*

Abstract – *This paper describes a system for monitoring vulnerabilities in a software. The basic concepts and mechanisms by which it is possible to publicly identify vulnerabilities in a software are explained. The system model and steps in the operation of the system are described. Finally, there are concluding considerations and directions that the further development of this tool could be done.*

Keywords: *NVD, CVSS, CVE, vulnerability, dependency*

1. UVOD

Danas postoji velika zavisnost od veb aplikacija, u rasponu od pojedinaca do velikih organizacija. Skoro sve je uskladišteno, dostupno ili trguje na vebu. Veb aplikacije mogu biti lične veb stranice, blogovi, vesti, društvene mreže, banke, agencije, forumi, aplikacije za e-trgovinu itd. Sveprisutnost veb aplikacija u našem načinu života i u našoj ekonomiji je toliko važna da ih čini prirodnom metom za zlonamerne umove koji žele da ih eksploatišu [1].

Softver otvorenog koda revolucionisao je modernu tehnološku industriju i nigde to nije evidentnije od područja razvoja veb aplikacija. Zahvaljujući otvorenim izvornim jezicima, bibliotekama, okvirima i razvojnim alatima, programeri su u stanju da kreiraju bogate i složene veb aplikacije, obično uz delić troškova - i vremena i novca - u poređenju sa prošlim godinama [2].

Iako upotreba komponenti otvorenog koda ubrzava razvoj, to košta, jer ranjivosti otkrivene u tim bibliotekama mogu uticati na zavisne aplikacije. Sve te komponente predstavljaju zavisnosti.

Termin zavisnost je široko rasprostranjen i koristi se kada se neki softver oslanja na drugi [3].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Sladić, red. prof.

Veliki broj tih zavisnosti poseduju javno poznate ranjivosti koje napadači mogu da eksploatišu. Zbog toga, potrebno je vršiti redovnu analizu tih zavisnosti kako bi se utvrdilo koje su ranjivosti prisutne i kako bi se pronašao način da se te ranjivosti uklone. Metoda putem koje se ovo ostvaruje jeste provera zavisnosti (eng. *dependency check*).
Upravo tim problemom bavi se ovaj rad. U njemu je prikazano jedno rešenje za proveru javno poznatih ranjivosti. Ono obuhvata prikupljanje svih poznatih ranjivosti i generesanje izveštaja koji pokazuje koje ranjivosti postoje i kako treba da se pristupi u rešavanju tih ranjivosti

2. MEHANIZMI ZA OKTRIVANJE RANJIVOSTI

2.1 Lista ranjivosti i izloženosti

Lista ranjivosti i izloženosti (eng. *Common Vulnerabilities and Exposures*) (CVE) je industrijski standard uobičajenih naziva za javno poznate sigurnosne ranjivosti, a organizacije su ga široko usvojile kako bi obezbedile bolju pokrivenost, lakšu interoperabilnost i poboljšanje sigurnosti [4].

CVE lista sastoji se od unosa (u zajednici ih nazivaju kao CVE identifikatori, CVE imena, CVE brojevi i CVE) koji predstavljaju jedinstvene, uobičajene identifikatore za javno poznate ranjivosti u oblasti informacione bezbednosti. Svaki CVE unos sadrži: identifikator, opis i sve relevantne reference (tj. izveštaje o ranjivosti i savete) [5].

2.2 Nacionalna baza ranjivosti

National Vulnerability Database (NVD) i njen pratilac, National Checklist Program (NCP) repozitorijum, pružaju vredan i fleksibilan skup usluga korisnicima širom sveta. NVD je osnovana 2005. godine kako bi obezbedila skladište podataka američke vlade o ranjivostima softvera i podešavanja konfiguracije, uz korišćenje otvorenih standarda kako bi se obezbedile pouzdane i interoperabilne informacije o metrici uticaja ranjivosti, metodama tehničke procene, podaci o identifikaciji IT proizvoda i reference za pomoć u sanaciji [6].

Posetioci koji pretražuju NVD bazu podataka mogu da pronađu opis mnogih detalja o bezbednosnim ranjivostima u softveru kako bi im pomogli da shvate kakve su to ranjivosti i kako im treba pristupiti. To uključuje opis CVE, koji je uglavnom dat od strane korporacije MITRE. Tada se daje slika koliko određena ranjivost može da bude opasna. Koliko je neka ranjivost opasna utvrđuje se na osnovu CVSS v2 i CVSS v3 metrike. Ova metrika definiše kako je ranjivost ocenjena

(kritična, visoka, srednja, niska), kao i detalje o tome kako bi eksploatacija ranjivosti mogla da se izvrši [7].

2.3. Sistem za ocenjivanje ranjivosti

Sistem za ocenjivanje ranjivosti (eng. *The Common Vulnerability Scoring System*) (CVSS) je specifikacija za dokumentovanje glavnih karakteristika ranjivosti i merenje potencijalnog uticaja eksploatacije ranjivosti. Motivacija za razvoj CVSS-a bila je pružanje standardizovane informacije za organizacije radi postavljanja prioriteta za ublažavanje ranjivosti [8]. CVSS se deli na tri metričke grupe: osnovna (eng. *Base*), vremenska (eng. *Temporal*) i okružene (eng. *Environmental*).

Osnovna metrička grupa predstavlja svojstva ranjivosti koja se menjaju tokom vremena, poput složenosti pristupa, vektora pristupa, stepena ugroženosti, integriteta i dostupnost sistem i zahtev za autentifikaciju sistema [9].

Vremenska metrička grupa meri svojstva ranjivosti koja se menjaju tokom vremena, kao što je postojanje zvanične zakrpe ili funkcionalni eksploatacioni kod [9].

Metrika okruženja meri svojstva ranjivosti koja su reprezentativna za okruženja korisnika, kao što su prevalencija pogođenih sistema i potencijal za gubitak. Pored toga, omogućava analitičarima da prilagode CVSS rezultat u zavisnosti od važnosti pogođene komponente za organizaciju korisnika [9].

3. PREGLED POSTOJEĆIH REŠENJA

U radu [10] opisan je sistem koji služi za pronalaženje ranjivosti u veb aplikacijama. Opisan je pristup automatizaciji skeniranja ranjivosti i testiranja bezbednosti sa rešenjem za cloud koje pruža samostalno okruženje koji ujedinjuje skenere i konfiguracije u jednom kliku [10]. Alat koji su razvili omogućuje dinamičko bezbednosno skeniranje i proveru zavisnosti bez prethodnog znanja o bezbednosti, što manuelnim testerima skraćuje količinu vremena koju inače troše na konfiguraciju i postavljanje okruženja. Rešenje objedinjuje 3 skenera bezbednosti *(Zed Attack Proxy Tool, OWASP Dependency Check, FindSecBugs Plugin*) koji su konfigurisani u distribuiranom sistemu, gde svaki skener generiše XML izveštaj. Generisani izveštaji se dalje provlače kroz proces mašinskog učenja u kom se identifikuju lažno pozitivni rezultati. Nakon toga se uklanjaju lažno pozitivni rezultati i generiše se krajnji izveštaj. Zaključak do kojeg su došli jeste da njihov sistem mnogo efikasnije identifikuje ranjivosti u odnosnu na manualne testere. U ovom sistemu cilj je da se postigne automatizacija skeniranja ranjivosti i testiranja bezbednosti, ali postoji nedostatak ukoliko želi maksimalno da se iskoristi *Zed Attack Proxy Tool* prilikom skeniranja pošto zahteva poznavanje alata i dodavanje određene dodatne konfiguracije. Sistem za praćenje ranjivosti jedino zahteva Github kredencijale kako bi pristupio repozitorijumu projekta za koji se vrši skeniranje.

Rad [11] opisuje statički pristup analizi ranjivosti. Skeniran je Maven centralni repozitorijum statičkom analizom radi otkrivanja potencijalnih softverskih grešaka. Za analizu korišćen je *FindBugs* alat koji ispituje Java bajt kod da bi otkrio brojne tipove grešaka. Skup podataka sadrži metričke rezultate koje pronalazi *FindBugs* alat za svaku verziju projekta koja je uključena u Maven repozitorijum [11]. Rezultat ovog procesa pruža skup podataka kojim svi mogu pristupiti radi daljeg istraživanja i analiziranja vezano za sigurnost u svojim sistemima. Nedostatak ovog rešenja jeste što je napravljen tako da skenira ceo Maven repozitorijum i onda se kao rezultat dobije skup podataka na osnovu kojeg je potrebno zaključiti da li se neke ranjivosti iz pronađenog skupa nalaze u projektu za koji se vrši provera, dok Sistem za praćenje ranjivosti vrši skeniranje samo zavisnosti koje se nalaze u projektu i generiše izveštaj o ranjivosti i omogućava pregled statistike, što troši manje resursa i vremena

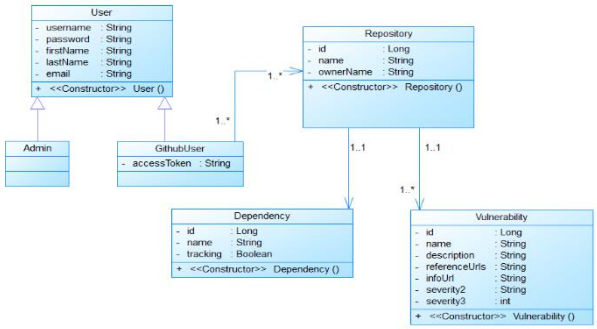
Autori rada [12] predstavljaju preciznu metodologiju koja odgovara potrebama industrije za pouzdano pronalaženje ranjivih zavisnosti u softveru otvorenog koda [12]. Ono što ova metodologija pruža jeste razlikovanje opsega zavisnosti i razmatranje direktnih i tranzitivnih zavisnosti. Pored toga, rešenje identifikuje i zavisnosti čiji je razvoj zaustavljen i uzima u obzir da zavisnosti koje se održavaju i objavljuju istovremeno predstavljaju jednu celinu prilikom analize. Za njihov rad koristili su 200 najpopularnijih biblioteka otvorenog koda koji koristi SAP *(*nem. *Systeme, Anwendugen und Produkte in der Datenverarbeitung)* u svom softveru. Zaključak do kojeg su došli jeste da se veliki procenat ranjivosti može rešiti jednostavnim ažuriranjem zavisnosti. Utvrđeno je da 20% zavisnosti koje ima ranjivost nije postavljeno (instalirano) [12], kao i da je za mali procenat zavisnosti zaustavljen dalji razvoj. Sistem za praćenje ranjivosti skenira samo direktne zavisnosti što daje dovoljno dobar izveštaj o ranjivostima koje postoje bez da troši količinu vremena koja je potrebna za opisanu metodologiju iz rada [12].

4. MODEL SISTEMA

4.1 Model sistema za praćenje ranjivosti u softveru

Ovaj sistem moguće je da koriste svi korisnici koji samostalno ili u saradnji razvijaju određeni softverski proizvod. Sistem je osmišljen tako da registrovani korisnici mogu da dodaju svoje Github repozitorijume na kojima im se nalazi projekat i za koje žele da vrše upravljanje ranjivostima. Postoji odvojeni administratorski deo aplikacije gde se dodaju korisnici kako bi mogli da pristupe ovom sistemu. Svaki registrovani korisnik ima mogućnost da doda repozitorijum, skenira repozitorijum, pregleda spisak svih pronađenih ranjivosti za skenirane repozitorijume, pregleda statistiku vezane za ranjivosti pronađene u svim repozitorijumima kao i u pojedinačnim. Pored toga, ima mogućnost da dodeli pravo pristupa za upravljanje ranjivostima drugim korisnicima kako bi oni mogli da prate ranjivosti njegovog sistema (ukoliko vrše saradnju) kao i da vrši praćenje zavisnosti koje se koriste za projekat i za koje će dobijati izveštaje. Sam model sistema za upravljanje ranjivostima i funkcionalnost skeniranja repozitorijuma opisana je u sledećem odeljku. Na slici 3 prikazan dijagram klasa opisanog sistema. U sistemu mogu da postoje dva tipa korisnika, oni su opisani sledećim klasama: *Admin* i *GithubUser*, obe klase nasleđuju klasu *User*. *GithubUser* dodatno ima polje

accessToken koje predstavlja token kojim se omogućava pristup repozitorijumu. Repozitorijum je opisan klasom *Repository* i svaki *GithubUser* može da ima jedan ili više repozitorijuma. Pored ovoga postoje klase koje opisuju zavisnost i ranjivost i to su klase *Dependency* i *Vulnerability*, respektivno. Svaki repozitorijum može da ima jednu ili više zavisnosti kao i jednu ili više ranjivost.

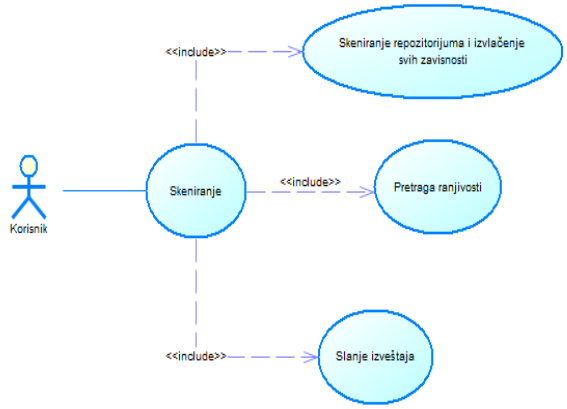


Slika 3. *Dijagram klasa sistema*

4.2 Koraci u radu sistema

Sistem za praćenje ranjivosti u softveru prilikom generisanja izveštaja o ranjivostima prolazi kroz tri faze (slika 4):

1. Skeniranje repozitorijuma i izvlačenje svih zavisnosti
2. Pretraga ranjivosti za pronađene zavisnosti
3. Slanje izveštaja



4. *Faze rada sistema*

4.2.1 Skeniranje repozitorijuma i izvlačenje svih zavisnosti

Kako bi korisnik započeo skeniranje potrebno je da izabere repozitorijum za koji želi da dobije izveštaj o ranjivostima. Ograničenje sistema se svodi na to da vrši skeniranje samo za *Maven* projekte. Nakon što korisnik izabere željeni repozitorijum, sistem vrši autentifikaciju korisnika putem *Github* API-a i pokreće skeniranje. Takođe, ovaj API omogućava sistemu da obavi sve potrebne akcije koje se izvršavaju prilikom skeniranja. Nakon što se izvrši uspešna autentifikacija, sistem parsira POM (eng. *Project Objet Model*) fajl iz projekta i iz njega izvlači listu svih zavisnosti koje se koriste. POM je osnovna jedinica rada u *Maven* projektima. To je XML datoteka koja sadrži informacije o projektu i detalje o konfiguraciji koje *Maven* koristi za izgradnju projekta [13].

4.2.2 Pretraga ranjivosti

Nakon dobijanja liste svih zavisnosti vrši se pretraga nad NVD bazom podataka i pronalaze se sve ranjivosti vezane za te zavisnosti.

Oslonac ove pretrage jeste na NVD API-u koji omogućava pretragu na osnovu ključne reči, gde ključnu reč u ovom slučaju predstavlja naziv zavisnosti.

4.2.3 Slanje izveštaja

Nakon pronalaska svih ranjivosti korisniku se šalje izveštaj na mejl i omogućava mu se da pregleda listu svih pronađenih ranjivosti na front-end delu aplikacije. Taj izveštaj sadrži sledeće informacije: naziv i link ka skeniranom repozitorijumu i listu pronađenih ranjivosti za zavisnosti koje su pronađene u repozitorijumu.

5. IMPLEMENTACIJA SISTEMA

Glavna komponenta ovog sistema jeste komponenta za skeniranje repozitorijuma. Ona se može podeliti na sledeće delove: skeniranje repozitorijuma radi pronalaženja svih zavisnosti (biblioteka), pretraga ranjivosti u NVD bazi podataka za pronađene zavisnosti i slanje mejl izveštaja korisniku za pronađene ranjivosti.

5.1 Komponenta za pronalaženje zavisnosti

Kako bi mogla da se izvrši pretraga za ranjivosti u NVD bazi, prvo je potrebno da se pronađu sve zavisnosti koje postoje unutar repozitorijuma. Metoda koja to izvršava prikazana je na listingu 1.

```
public void
searchForDependencies(RepositoryService
repoService, GitHubClient client, List<String>
dependencies, Repository repository)
        throws IOException,
ParserConfigurationException, SAXException {
    org.eclipse.egit.github.core.Repository repo
=
repoService.getRepository(repository.getOwnerNam
e(), repository.getName());
    ContentsService contentService = new
ContentsService(client);
    List<RepositoryContents> contentList =
contentService.getContents(repo);
    searchRecursive(contentList, repo,
contentService, dependencies);
}
```

Listing 1. *Metoda za pronalaženje zavisnosti*

U ovoj metodi se prvobitno dobavlja repozitorijum sa metodom *getRepository* na osnovu korisničkog imena vlasnika i naziva repozitorijuma, respektivno. Na kraju se poziva metoda *searchRecursive* koja rekurzivno prolazi kroz sve direktorijume unutar repozitorijuma kako bi pronašla fajl koji sadrži sve zavisnosti.

Ova metoda prima sledeće parametre: *contentList* – lista svih direktorijuma i fajlova unutar određenog direktorijuma repozitorijuma, repo – repozitorijum koji se skenira, *contentService* – servis koji omogućava dobavljanje svih direktorijuma i fajlova i *dependencies* – prazna lista u kojoj će se čuvati sve pronađene zavisnosti.

Kako je ovaj sistem ograničen samo na skeniranje *Maven* projekata, potrebno je vršiti rekurzivno kretanje sve dok se ne pronađe fajl pod nazivom *pom.xml* u kom su izlistane sve zavisnosti. Kada se fajl pronađe vrši se parsiranje i izvlačenje svih zavisnosti.

5.2 Komponenta za pretragu ranjivosti

Nakon što se pronađu sve zavisnosti koje se nalaze u repozitorijumu, vrši se pretraga ranjivosti. Pretraga ranjivosti vrši se nad NVD bazom podataka i to je implementirano uz oslonac na zvanični NVD API koji sadrži specifikaciju za način komunikacije sa bazom. Za pretragu nad bazom koristi se pretraga po ključnoj reči, što ovde predstavlja naziv zavisnosti. Za svaku zavisnost šalje se zahtev NVD bazi i dobija se odgovor u JSON formatu čija je šema definisana u specifikaciji NVD API-a. Ovaj JSON format dalje se mapira na model koji je definisan unutar sistema.

5.3 Komponenta za slanje izveštaja

Nakon uspešnog skeniranja repozitorijuma za zavisnosti i pretrage svih ranjivosti za njih, poziva se metoda koja generiše izveštaj koja se šalje korisniku na mejl. Generisani izveštaj sadrži sledeće informacije: naziv repozitorijuma, link ka repozitorijumu i listu svih pronađenih ranjivosti. Svaka ranjivost iz liste sadrži: naziv zavisnosti na koju se odnosi, CVE ID i link ka zvaničnom opisu ranjivosti unutar NVD baze.

6. ZAKLJUČAK

Tema ovog rada je kreiranje sistema za praćenje ranjivosti u softveru. Predstavljen je sistem koji na osnovu skeniranog repozitorijuma i izvlačenja svih zavisnosti kreira izveštaj sa listom javno poznatih ranjivosti vezane za pronađene zavisnosti.

Objašnjeni su neki od mehanizama za otkrivanje javno identifikovanih ranjivosti u softveru. Opisana je Lista ranjivosti (CVE), National Vulnerability Database (NVD), kao i efikasan sistem za ocenjivanje ranjivosti (CVSS).

Pored teorijskog dela, opisana su postojeća rešenja, model sistema i koraci u radu sistema. Prikazana je i implementacija sistema sa opisom tehnologija koje su korišćene za implementaciju, komponenta za skeniranje repozitorijuma na koju se ovaj sistem oslanja.

Preporuka za dalji razvoj veb-aplikacije je poboljšanje performansi pretrage javno identifikovanih ranjivosti. Pretraga se vrši putem NVD API-a gde je rezultat pretrage JSON datoteka sa svim ranjivostima koje su pronađene za zavisnost.

Nakon dobijenog rezultata parsira se JSON datoteka i mapira na model ranjivosti iz sistema. Ova faza oduzima najviše vremena u procesu analize. Razmotriti korišćenje *Elasticsearch*-a za skladištenje i pretragu celokupne NVD baze podataka.

Dalji razvoj veb-aplikacije može da ide u smeru uvođenja dodatnih vrsta projekata za skeniranje. Za sada je moguće samo skeniranje i izvlačenje zavisnosti iz Maven projekata.

7. LITERATURA

[1] Jose Carlos Coelho Martins da Fonseca, Marco Vieira, and Henrique Madeira, "Evaluation of Web Security Mechanisms Using Vulnerability & Attack Injection ", IEEE Transactions on depandable and soft computing, 2013.

[2] https://resources.whitesourcesoftware.com/blog-whitesource/owasp-a9-using-components-with-known-vulnerabilities (pristupljeno pristupljeno u maju 2021)

[3] Serena Elisa Ponta, Henrik Plate, "Detection assessment and mitigation of vulnerabilities in open source dependencies, 2020.

[4] Minzhe Gou, Ju An Wang, "An Ontology-based Approach to Model Common Vulnerabilities and Exposures in Information Security", ASEE Southeast Section Conference, 2009.

[5] https://ce.mitre.org/cve/identifiers/index.html, (pristupljen pristupljeno u maju 2021)

[6] Harold Booth, Doug Rike and Greg Witte, "The National Vulnerability Database (NVD): Overview", 2013.

[7] https://resources.whitesourcesoftware.com/blog-whitesource/the-national-vulnerability-database-explained, (pristupljeno u maju 2021)

[8] Peter Mell, Karen Scarfone, Sasha Romanosky, "An Analysis of CVSS Version 2 Vulnerability Scoring", Third International Symposium on Empirical Software Engineering and Measurement, 2006.

[9] Peter Mell, Karen Scarfone, Sasha Romanosky, A Complete Guide to the Common Vulnerability Scoring System Version 2.0, 2007.

[10] J. A. D. C. A. Jayakody, A. K. A. Perera, G. L. A. K. N. Perera, "Web-application Security Evaluation as a Service with Cloud Native Environment Support", International Conference on Advancements in Computing (ICAC), 2019.

[11] Dimitris Mitropoulos, Vassilios Karakoidas, Panos Louridas, Georgios Gousios, Diomidis Spinellis, "The Bug Catalog of the Maven Ecosystem", 2014.

[12] Ivan Pashchenko, Henrik Plate, Serena Elisa Ponta, Antonino Sabetta and Fabio Massacci, "Vulnerable Open Source Dependencies: Counting Those That Matter", Proceedings of the 12th International Symposium on Empirical Software Engineering and Measurement (ESEM), 2018.

[13]https://maven.apache.org/guides/introduction/introduction-to-the-pom.html (pristupljeno u junu 2021)

Kratka biografija:

Vladimir Cvetanović rođen je u Novom Kneževcu 29.09.1996. godine. Fakultet tehničkih nauka u Novom Sadu, smer Računarstvo i automatika, upisao je 2015. godine. Osnovne akademske studije završio je 2019. godine, nakon čega je upisao master akademske studije na Fakulte tehničkih nauka, smer Računarstvo i automatika.
kontakt: cvetanovic9696 @gmail.com