Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/08BE27Andjelic

OTKRIVANJE SIGURNOSNIH PROPUSTA U *WEB* APLIKACIJAMA METODOM PENETRACIONOG TESTIRANJA

DETECTING SECURITY VULNERABILITIES IN WEB APPLICATIONS USING PENETRATION TESTING

Doroteja Anđelić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad sadrži osnovne koncepte penetracionog testiranja, opisujući sve faze u metodologiji, od dobijanja informacija za identifikovanje mogućih slabih tačaka do eksploatisanja ranjivosti. Opisane su pronađene ranjivosti u realnim web aplikacijama Juice Shop i Peruggia. Za sprovođenje testiranja i otkrivanje potencijalnih sigurnosnih ranjivosti upotrebljeni su alati DIRB, Nikto, Dotdotpwn i Burp Suite.*

Ključne reči: *penetraciono testiranje; sigurnost i bezbednost web aplikacija; hakovanje*

Abstract – *This paper contains basic concepts of penetration testing, describing all stages in the methodology, from obtaining information to identify possible weak points to exploiting vulnerabilities. Found vulnerabilities in real-world web applications Juice Shop and Peruggia are described. DIRB, Nikto, Dotdotpwn and Burp Suite tools were selected to conduct testing and detect potential security vulnerabilities.*

Keywords: *Penetration testing; Web Application security; Hacking*

1. UVOD

Imajući u vidu da je prioritet isporučiti proizvod na vreme, programeri ne mogu da testiraju svoje aplikacije u potpunosti sa aspekta sigurnosti. Samim tim stvara se potreba za profesionalcima koji su specijalizovani za testiranje sigurnosti. Ovi profesionalci nazivaju se penetracioni testeri.

Njihovim uključivanjem u razvojni proces, oni preuzimaju odgovornost testiranja aplikacije. Gledajući na sistem sa tačke gledišta napadača, izvršioci penetracionog testa identifikuju i eksploatišu postojeće ranjivosti.

U ovom radu su opisani osnovni koncepti penetracionog testiranja, kao i sve faze u metodologiji, od dobijanja informacija za identifikovanje mogućih slabih tačaka do eksploatisanja ranjivosti. Pronađene su neke od postojećih ranjivosti u realnim *web* aplikacijama *Juice Shop* i *Peruggia*.
Penetraciono testiranje predstavlja proaktivan način testiranja aplikacija i raznih sistema.

NAPOMENA:
Ovaj rad proistekao je iz master rada, čiji je mentor bio dr Darko Čapko, vanr. prof.

Cilj testiranja jeste pronalaženje što više nedostataka, propusta i ranjivosti u sigurnosti aplikacije i izbegavanje rizika koje izazivaju ovi nedostaci. Jedan od zadataka penetracionog testera jeste i da, kada pronađe i verifikuje ranjivost, posavetuje programere kako da isprave uočenu grešku i spreče njeno ponovno javljanje. Testiranje, zahvaljujući korišćenju istih alata i tehnika koje bi koristio i zlonameran napadač, nalikuje stvarnom napadu. U najboljem slučaju propusti će se otkriti i ublažiti pre nego što napadač uoči i iskoristi slabost. Testiranje se vrši na kontrolisan način.

Veliki posao penetracionog testiranja više je umetnost nego nauka. Penetraciono testiranje, pored veštine i iskustva, iziskuje specijalni vid shvatanja koje se ne može sistematizovati [1].

2. SIGURNOST SOFTVERA

Softverski problem predstavlja centralni i kritični aspekt problema računarske sigurnosti. Kvar softvera sa posledicama u sigurnosti uključuje greške u implementaciji, kao što su *buffer overflow* i nedostaci u dizajnu (loše upravljanje greškama). Sve češće se koriste i eksploatišu softverske greške ne bi li se pristupilo ciljanom sistemu [2].

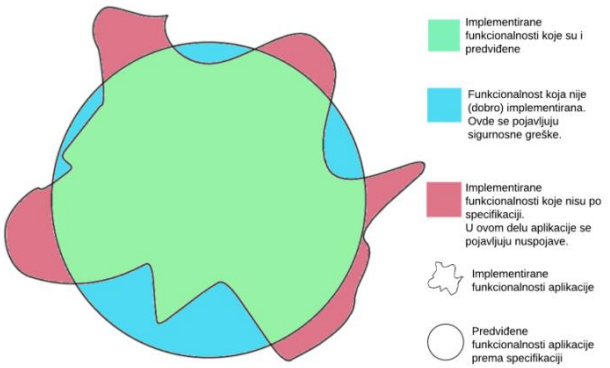
Pod sigurnim softverom podrazumeva se njegovo pravilno funkcionisanje uprkos zlonamerim napadima.

Jednostavnije je zaštititi softver koji nema grešaka nego softver pun ranjivosti. Zato je bitno obratiti pažnju na dizajn softvera, njegovu sigurnost, kao i edukovanje programera i arhitekata kako se gradi siguran softver. Nastajanje sigurnog softvera uključuje procese dizajniranja, izrade i testiranja softvera. Na ovaj način se identifikuju i brišu problemi u samom softveru. Dobija se softver koji je spreman da izdrži i podnese napad proaktivno [2]. U praksi se vrlo često izvode penetracioni testovi nakon faze razvoja softvera, ne bi li se naknadnom analizom utvrdio potencijalni stepen ranjivosti softvera.

2.1 Sigurnosni propusti u softveru

Razvoj sigurnog softvera podrazumeva uključivanje sigurnosnog aspekta u životni ciklus razvoja softvera.

Slika 1 ilustruje prirodu nuspojava većine softverskih ranjivosti. Krugom je predstavljeno predviđeno ponašanje softvera koje se obično definiše specifikacijom. Amorfan oblik nanesen na krug predstavlja stvarnu implementiranu funkcionalnost aplikacije i njeno pravo ponašanje. Idealno bi bilo da se implementirane funkcionalnosti preklapaju sa funkcionalnostima iz specifikacije, što se u praksi skoro nikada ne dešava [3].



Slika 1. *Predviđeno i implementirano ponašanje softvera [3]*

Zahvaljujući lošem kodiranju, funkcionalnosti programa nehotično utiču jedna na drugu i greške neizbežno ulaze u sistem. Područja kruga koja kodirana aplikacija ne pokriva (prikazano plavom bojom) predstavljaju prostor gde se pojavljuju greške. Kada se jednom ispolji nekorektno ponašanje aplikacije, te greške je prilično lako otkriti. Implementirane funkcionalnosti koje nisu predviđene specifikacijom predstavljaju nenamerne i potencijalno opasne funkcionalnosti softvera u kojima se javljaju nuspojave aplikacije. Ove funkcionalnosti su slikovito predstavljene kao područja koja izlaze van kružne regije (prikazano crvenom bojom) [3].

Implementirane funkcionalnosti koje nisu planirane specifikacijom, kao i nuspojave normalnog funkcionisanja aplikacije, ostavljaju prostor za pojavu sigurnosnih propusta [3].

2.2 *OWASP Top* 10 – 2017

Što je softver složeniji, postizanje sigurnosti je teže, a poteškoće eksponencijalno rastu. Identifikovanjem najozbiljnijih sigurnosnih rizika *web* aplikacije, *OWASP* (*Open Web Application Security Project*) je probudio svest programera i menadžera. Na ovaj način postao je sigurnosni standard aplikacija. Pored identifikovanja rizika, *OWASP* ukazuje i na njihove uticaje i protivmere. Rizici koji se nalaze u okviru projekta *OWASP Top* 10 su [4]:

1. *Injection* – ubrizgavanje koda
2. *Broken Authentication* – slaba (neadekvatna) autentifikacija
3. *Sensitive Data Exposure* – otkrivanje poverljivih podataka
4. *XML External Entities (XXE)* – *XML* spoljno ubrizgavanje entiteta
5. *Broken Access Control* – neovlašćena kontrola pristupa
6. *Security Misconfiguration* – pogrešna sigurnosna konfiguracija
7. *Cross-Site Scripting (XSS)* – ubrizgavanje skripte
8. *Insecure Deserialization* – nezaštićena deserijalizacija
9. *Using Components with Known Vulnerabilities* – korišćenje komponenti sa poznatim ranjivostima
10. *Insufficient Logging and Monitoring* – nedovoljno beleženje akcija i nadgledanje rada sistema

Najkritičnije ranjivosti u *web* aplikacijama su greške ubrizgavanja koda. Interaktivne *web* aplikacije koriste korisnički unos, obrađuju ga i vraćaju rezultat korisniku. Kada je aplikacija ranjiva na ubrizgavanje, ona prihvata unos od korisnika bez pravilne ili čak bilo kakve validacije i obrađuje ga. Zlonamerni unos će prevariti aplikaciju primoravanjem osnovnih komponenata da izvrše zadatke za koje aplikacija nije programirana.

Funkcije aplikacije koje se odnose na autentifikaciju i upravljanje sesijom često su implementirane pogrešno, omogućavajući napadačima da kompromituju lozinke, ključeve, tokene sesija [4]…

Jedan od glavnih ciljeva bezbednosti informacija je zaštita poverljivosti podataka. Mnoge *web* aplikacije ne štite na pravi način kredencijale, brojeve kreditnih kartica, zdravstvene informacije i ostale lične podatke. Napadači tako slabo zaštićene podatke mogu ukrasti ili izmeniti. Osetljivi podaci bez dodatne zaštite, kao što je enkripcija, mogu biti ugroženi i zahtevaju posebne mere predostrožnosti kada se razmenjuju sa pretraživačem. Kriptografija, praksa komunikacije i dešifrovanje tajnih zapisa ili poruka, koriste se za zaštitu poverljivosti i integriteta podataka [5].

XXE predstavlja ranjivost *web* sigurnosti koja omogućava napadaču da se meša u obradu i analiziranje *XML* podataka. Do napada dolazi kada slabo konfigurabilni *XML* parser obrađuje ulaz koji sadrži referencu na eksterni entitet. Na ovaj način napadač može da pregleda datoteke na serveru, komunicira sa sistemima kojima i sama aplikacija pristupa, otkrije poverljive podatke, falsifikuje zahtev na strani servera [4]…

Ograničenja u vezi sa onim što autentifikovani korisnici mogu da rade često se ne sprovode pravilno. Napadači koriste ove mane kako bi pristupili neovlašćenim funkcijama ili podacima. Za sprečavanje ranjivosti najčešće se koristi autorizacija koja se zasniva na ulogama.

Pogrešna sigurnosna konfiguracija je jednostavan način za eksploatisanje *web* stranice. Definiše se kao neuspešna implementacija svih bezbednosnih kontrola *web* aplikacije ili kao implementacija sa greškama. Ovo je obično rezultat podrazumevane ili nepotpune konfiguracije, otvorenog *cloud storage*-a, pogrešno konfigurisanog *HTTP* zaglavlja ili poruke o greškama koja sadrži osetljive informacije [4].

XSS predstavlja slabost koja se oslanja na validaciju ulaza, slično *SQL injection*-u, a nastaje kada se korisnički unos nepravilno filtrira [6]. To omogućava napadaču da ubrizga zlonameran kod koji se kasnije izvršava u pretraživaču žrtve.

Nesigurna deserijalizacija je ranjivost koja se javlja kada se nepouzdani podaci koriste za zloupotrebu logike aplikacije, sprovođenje *Denial-of-Service* (*DoS*) napada ili izvršavanje proizvoljnog koda nakon serijalizacije. Primer eksploatisanja ranjivosti predstavlja uspešno deserijalizovanje objekta, njegovo modifikovanje, a zatim i ponovno serijalizovanje.

Sve češće i jednostavne *web* stranice sadrže komponente od kojih zavise. Biblioteke i drugi softverski moduli rade sa istim privilegijama kao i aplikacija. Ako se iskoristi ranjiva komponenta, ovakav napad dovodi do gubitka podataka ili preuzimanja servera [4].

Nedovoljno logovanje i nadgledanje se donekle razlikuje od prethodnih devet rizika. Iako ne dovodi do direktnog upada, zakasnelo i nepravovremeno otkrivanje upada predstavlja neuspeh. Premda stopostotna sigurnost nije

realan cilj, postoje načini da se aplikacija redovno prati kako bi se određene mere preudzele usled neočekivanih događaja.

3. PENETRACIONO TESTIRANJE

Penetraciono testiranje predstavlja jednu od najopsežnijih metoda za pronalaženje ranjivosti i povećanje zaštite računarskih sistema i aplikacija. Iako se koriste alati, trikovi i tehnike koje koriste i zlonamerni hakeri, penetraciono testiranje je legalno uz dozvolu cilja [7].

3.1 Faze penetracionog testiranja

Proces penetracionog testiranja se može *razbiti* na niz koraka ili faza. Svi koraci zajedno čine obimnu metodologiju za upotpunjavanje penetracionog testa. Detaljnom analizom izveštaja koji predstavljaju odgovore na incidente, zaključuje se da *black boxing* hakeri prate proceduru kada napadaju ciljanu metu. Hakeri sakupljaju informacije, identifikuju ranjivost, a zatim je eksploatišu, odnosno izvršavaju napad. Nakon prodiranja u ciljani sistem, izazov je ostati što duže skriven. Ovakav organizovan pristup testera drži fokusiranog i usmerava ga ka sledećem koraku koji je olakšan zahvaljujući rezultatima iz prethodne faze [8].

Metodologija deli kompleksan proces na seriju manjih, bolje upravljivih zadataka. Postoje različite metodologije i sve sadrže između četiri i sedam faza. Iako broj faza varira, svaka metodologija obezbeđuje kompletan pregled celog procesa penetracionog testiranja [8].

3.1.1 Četvorofazni proces penetracionog testiranja

U okviru četvorofazne metodologije izdvajaju se sledeće faze: izviđanje, skeniranje, eksploatacija ranjivosti i održavanje pristupa **Error! Reference source not found.**. Slikovit primer ova četiri koraka može se predstaviti okrenutim trouglom (slika 2).

Slika 2. *Slikovit prikaz četiri faze penetracionog testiranja [8]*

Izlaz svake faze je širok, a i koristi se kao ulaz naredne faze. Pomerajući se ka poslednjoj fazi, dobijaju se sve specifičniji detalji koji testera vode do cilja. Svaka informacija i svaki detalj o meti se sakuplja i skladišti.

Početna faza, koja je potpuni kontrast od faza koje slede, predstavlja sprovođenje temeljnog istraživanja o svim javno dostupnim informacijama.

Skeniranje uključuje pronalaženje mogućih otvora ili ranjivosti pomoću ručnog testiranja ili automatizovanog skeniranja.

Treća faza obuhvata eksploataciju ranjivosti, kompromitovanje cilja i dobijanje pristupa.

Održavanje pristupa ili posteksploatacija podrazumeva podešavanje sredstava za eskalaciju privilegija na eksploatisanom elementu ili instaliranje zadnjih vrata.

Nakon završetka četvrte faze, neophodno je sumirati sve pronalaske u obliku izveštaja penetracionog testiranja.

4. NAPADANJE NEDOSTATAKA U *WEB* APLIKACIJAMA

Zbog velikog broja *web* stranica na internetu i povećanja broja organizacija koje svoje poslove obavljaju *online*, *web* aplikacije i *web* serveri su atraktivne mete za napadače.

Investiranje resursa u pisanje bezbednog koda je efikasan metod za minimiziranje ranjivosti *web* aplikacija. Međutim, pisanje bezbednog koda je lako objasniti, ali teško implementirati.

4.1 Alati

Izvršioci testa procenjuju sigurnost sistema primenjujući različite alate za penetraciono testiranje. Korišćenje alata nudi više prednosti. Ako se efikasno koriste, mogu da obave veći deo posla koji je neophodan za kasniju analizu. Alat nikad ne može i ne treba u potpunosti da odmeni testera. Svaku vrstu ranjivosti koju je pronašao alat, neophodno je ručno proveriti. Glavne prednosti alata su brzina i jednostavan način otkrivanja određene sigurnosne ranjivosti.

Alati koji su korišćeni prilikom testiranja *web* aplikacija su *DIRB*, *Nikto*, *Dotdotpwn* i *Burp Suite*.

4.2 Testiranje ranjivih *web* aplikacija *Juice Shop* (*JS*) i *Peruggia*

Ukoliko penetracioni tester nema izričito pismeno odobrenje vlasnika, skeniranje, testiranje ili eksploatisanje ranjivosti na serverima i aplikacijama na internetu su ilegalni u većini država.

OWASP održava *Juice Shop*, *web* aplikaciju otvorenog koda, koji podržava učenje zasnovano na izazovima. Aplikacija je namerno ranjiva na deset najčešćih ranjivosti koje je identifikovao *OWASP* [4].

Pred navedena četiri alata postavljena su tri izazova – pronalazak putanje */ftp*, preuzimanje sakrivene datoteke *eastere.gg* i pronalaženje pravog *easter egg*-a (tabela 1).

Tabela 1. *Alati i eksploatisane ranjivosti JS aplikacije*

	Pronalazak putanje */ftp*	Preuzimanje datoteke *eastere.gg*	Pronalazak pravog *easter egg*-a
DIRB	✓	×	×
Nikto	✓	×	×
Dotdotpwn	×	×	×
Burp Suite	✓	✓	✓

Putanju */ftp* su otkrili svi osim *Dotdotpwn* alata. *Dotdotpwn* je koristio datoteku sa dužim spiskom putanja koje su dodate na glavnu putanju, a samim tim je i testiranje iziskivalo više vremena. Alati *DIRB* i *Nikto* su za kraće vreme uradili efektivniji posao. Uspeli su da detektuju zanimljivu putanju */ftp* i penetracionom testeru ukazali na tokove daljeg testiranja. *Burp Suite*, osim izlistanih svih potencijalnih putanja, nudi mogućnost prikazivanja *HTTP* zahteva.

U okviru pronađene putanje *http://localhost:3000/ftp* izlistane su datoteke sa različitim ekstenzijama. Moguće je preuzeti i pregledati samo one sa ekstenzijom *.pdf* ili *.md*. Jedan od izazova predstavlja preuzimanje i pregledanje sadržaja datoteke *eastere.gg*. Presreten je

zahtev sa *URL*-om *http://localhost:3000/ftp/eastere.gg* i zahvaljujući *Burp Suite*-ovom *Repeater*-u ekstenzija datoteke je u putanji promenjena u *.pdf* nakon čega se datoteka uspešno preuzima. U *HTTP* odgovoru i u datoteci *eastere.gg* nalazi se poruka sa kodom koji krije pravi *easter egg*.

Poslednji zadatak – pronalaženje pravog *easter egg*-a uspeo je da eksploatiše samo *Burp Suite*. Kod dobijen iz *HTTP* odgovora, prilikom preuzimanja datoteke *eastere.gg*, dekodiran je iz formata *base64*. Dobijena *ASCII* putanja ne vodi do rešenja, pa je pronadjena putanja dekodovana uz pomoć *ROT13* tehnike maskiranja. Otkrivena putanja otvara pravo *vaskršnje* jaje u obliku *3D* planete.

Web aplikacija *Peruggia* predstavlja društvenu mrežu na kojoj se postavljaju slike i komentari.

Alati *DIRB*, *Nikto* i *Burp Suite* su detektovali potencijalno zanimljive putanje, dok ih *Dotdotpwn*, koristeći *directory traversal* napad, nije pronašao (tabela 2).

Tabela 2. *Uspešnost testiranja Peruggia aplikacije*

	Pronalazak skrivenih putanja
DIRB	✓
Nikto	✓
Dotdotpwn	×
Burp Suite	✓

DIRB je detektovao devet zanimljivih putanja i na nekima od njih pronađeni su kredencijali korisnika sa administratorskim pravima.

Nikto je generisao pregledan izveštaj u kojem se navodi verzija servera, nedostatak *HttpOnly tag*-a na *cookie*-u, zastarele verzije softvera, kao i nedostatak adekvatne zaštite u zaglavlju.

Dotdotpwn nije pronašao potencijalno ranjive lokacije koje se nalaze iza korenskog direktorijuma.

Burp Suite je potvrdio postojanje putanja koje su već pronađene uz pomoć alata *DIRB* i *Nikto*. *Burp Intruder* automatizuje slanje više zahteva na server, zamenjujući selektovane vrednosti korisničkim unosom, i zapisuje sve odgovore kako bi se analizirali kasnije.

5. ZAKLJUČAK

Penetraciono testiranje omogućava da se softver ili određena moguća ciljna tačka hakera posmatra očima neprijatelja – zlonamernih hakera. Proces testiranja iziskuje dodatno vreme da bi se *zakrpio* sistem pre potencijalnog stvarnog napada i udarca.

Kroz ovaj rad je predstavljeno testiranje sigurnosti *web* aplikacija metodom penetracionog testiranja. Za sprovođenje testiranja i otkrivanje potencijalnih sigurnosnih ranjivosti izabrane su *Juice Shop* i *Peruggia web* aplikacije. Opisane su pronađene ranjivosti i alati pomoću kojih se došlo do njih.

Pokazano je da u datim primerima alati *DIRB*, *Nikto* i *Burp Suite* uspešno pronalaze ranjivosti, dok se *Dotdotpwn*, koji koristi *directory traversal* napad, nije pokazao kao prikladan. *Burp Suite* sa svojom širokom lepezom funkcionalnosti predstavlja veoma efektivan alat.

Ukoliko se ne otkrije ni jedna ranjivost prilikom testiranja, dokazano je samo da ne postoje ranjivosti za date scenarije, ali ne i da ranjivosti ne postoje. Prolazak penetracionog testa pruža vrlo malo sigurnosti da je aplikacija imuna na sve napade. Aplikacija je imuna samo na napade na koje je testirana.

Skrivena priroda sigurnosnih grešaka je razlog zašto su neophodne specifične tehnike za testiranje. Penetraciono testiranje prkosi tradicionalnom modelu verifikacije specifikacije aplikacije i umesto toga identifikuje neodređene i nesigurne nuspojave ispravne funkcionalnosti aplikacije [9].

Imajući u vidu da su u ovom radu predstavljeni pronalasci i načini eksploatacije samo nekih postojećih ranjivosti aplikacija, dalji rad bi podrazumevao identifikovanje i eksploataciju preostalih slabosti. Pohod na ranjivosti i eksploatacija nedostataka mogu se izvršiti nad ostalim ranjivim *web* aplikacijama koje se nalaze u okviru projekta *OWASP BWA*. Jedan od izazova predstavlja testiranje realne aplikacije, ali uz pristanak i dozvolu mete. Kako bi se postigla veća efikasnost, skup alata za testiranje mogao bi se proširiti.

Jedan od najvećih atributa penetracionog testiranja i hakovanja jeste što se nikada ne doseže kraj.

6. LITERATURA

[1] James P. McDermott, „Attack net penetration testing”, *NSPW*, pp. 15-21, 2000.

[2] Gary Mcgraw, *Software security: building security in*, Addison-Wesley Professional, vol. 1, 2006.

[3] Herbert H. Thompson, „Why security testing is hard”, *IEEE Security and Privacy*, vol. 1, no. 4, pp. 83-86, 2003.

[4] *OWASP Top* 10 – 2017, *The Ten Most Critical Web Application Security Risks*

[5] Gilberto Najera Gutierrez i Juned Ahned Ansari, *Web penetration testing with Kali Linux*, Packt Publishing, 2018.

[6] Joseph Muniz i Aamir Lakhani, *Web penetration testing with Kali Linux*, Packt Publishing, 2013.

[7] Ajinkya A. Farsole, Amruta G. Kashikar i Apurva Zunzunwala, „Ethical Hacking”, *International Journal of Computer Applications (IJCA)*, vol. 1, no. 10, pp. 14-20, 2010.

[8] Patrick Engebretson, *The basics od hacking and penetration testing: Ethical hacking and penetration testing made easy*, Elsevier, 2013.

[9] H. H. Thompson, „Application penetration testing”, *IEEE Security and Privacy*, vol. 3, no. 1, pp. 66-69, 2005.

Kratka biografija

Doroteja Anđelić rođena je 1995. godine u Novom Sadu. Diplomirala je 2018. godine na Fakultetu tehničkih nauka (studijski program Elektroenergetski softverski inženjering) sa prosečnom ocenom 9,71. Master akademske studije (studijski program Primenjeno softversko inženjerstvo) upisala je školske 2018/2019. godine.