Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE31Marin

RAZVOJ APLIKACIJE ZA TESTIRANJE BEZBEDNOSTI WEB APLIKACIJA

DEVELOPMENT OF APPLICATION FOR TESTING WEB APPLICATIONS SECURITY

Ivana Marin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljen je razvoj aplikacije za testiranje bezbednosti web aplikacija u Python programskom jeziku, kroz prethodno analizirane koncepte bezbednosti i ranjivosti web aplikacija i penetracionog testiranja.*

Ključne reči: *Bezbednost web aplikacija; Penetraciono testiranje; Ranjivosti web aplikacija*

Abstract – *This paper presents the development of application for testing Web applications security in Python programming language, through previously analyzed concepts of web application security and vulnerabilities and penetration testing.*

Keywords *Web application security; Penetration testing; Web application vulnerabilities*

1. UVOD

Danas, u 21 veku, milioni ljudi širom sveta stupaju u interakciju sa web aplikacijama svakog dana, bilo da su to društvene mreže, kupovina, bankarstvo, pošta ili potraga za informacijama.

Kompanije i organizacije kao proizvođači žele da svojim korisnicima omoguće najkvalitetniji softver i odgovornost koju imaju prema njima, a i prema tržištu je velika. Jedan od ključnih faktora jeste bezbednost web aplikacije. Bezbednost se već dugi niz godina definiše kao krucijalan faktor u izgradnji web aplikacija. Softveri, kao što su web aplikacije, svakodnevno su izložene hakerskim napadima, pokušajima neovlašćenog pristupa, krađi osetljivih informacija…

To su samo neke od kriminalnih aktivnosti koje se nalaze u okruženju web aplikacija. Stoga je neophodno obezbediti najveći nivo zaštite i omogućiti da aplikacija uspešno funkcioniše, da zaštiti sebe i svoje podatke u svakom trenutku, a naročito ukoliko dođe do malicioznih napada. Danas, više nego pre, kompanije ulažu u zaštitu svojih aplikacija i korisnika. Jedan od najboljih izbora za to predstavlja penetraciono testiranje web aplikacija.

2. ISTRAŽIVANJE O RANJIVOSTIMA WEB APLIKACIJA

Sa sigurnosnog aspekta, ranjivost predstavlja manu ili slabost u dizajnu, implementaciji, operaciji, koja može biti iskorišćena za kompromitovanje aplikacije.

Pretnja predstavlja bilo šta što može naškoditi aplikaciji, koristeći njenu ranjivost. To može biti maliciozni napadač koji dolazi u pristup podacima iz baze podataka.

Jedan od najvećih problema u bezbednosti predstavlja reaktivan pristup umesto proaktivnog. Reaktivnost znači da se problem već dogodio i da posledice treba sanirati. Proaktivnost treba da bude pravi pristup i kao rezultat toga, bezbednosni problemi biće otkriveni u ranijim fazama razvoja, a ne kada se desi napad [1]. Stoga je veoma važno integrisati bezbednost kroz sve faze razvoja softvera, od samih definisanja zahteva, preko dizajna, implementacije, testiranja, pa sve do puštanja u produkciju i održavanja softvera [1]. Međutim, čak i uz ispraćene sigurnosne prakse, velika je verovatnoća da će ostati sigurnosnih propusta.

Sa povećanjem broja organizacija koje posluju na mreži, web aplikacije i web serveri predstavljaju odličnu priliku za napadače. Za interakciju sa web aplikacijom, potreban je samo pregledač. Tokom godina, web aplikacije čuvaju ključne podatke kao što su lični podaci i finansijske evidencije. Kada se web aplikacija ne testira na ranjivosti i napadač dobije pristup podacima korisnika, to može ozbiljno uticati na vrednost brenda kompanije, ako korisnik pokrene postupak protiv kompanije zbog nedovoljnog rada na zaštiti njegovih podataka. To takođe može dovesti do gubitka prihoda, jer će se mnogi korisnici "preseliti" kod konkurenata koji bi obezbedili bolju sigurnost [2].

Stručnjaci iz OWASP zajednice sastavili su ono što se smatra konačnom standardnom listom glavnih ranjivosti web aplikacija. Neke od najpoznatijih i najčešćih ranjivosti sa ove liste su: *SQL Injection, Broken Authentication, Broken Access Control, Cross-site Scripting* (XSS) [3].

3. ISTRAŽIVANJE O PENETRACIONOM TESTIRANJU WEB APLIKACIJA

Penetraciono testiranje se definiše kao proaktivni način testiranja web aplikacija simulacijom napada, slično stvarnom napadu, postavljajući testera u ulogu napadača (hakera) [2]. Ovakvo testiranje vezano je ugovorom između testera i vlasnika aplikacije (klijenta) koja će se testirati [2].

Penetracioni testovi se pokazuju korisnim u pronalaženju ranjivosti u organizaciji i proveravanju da li će ih napadač moći iskoristiti da bi stekao neovlašćen pristup. Iz

---

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

izveštaja o testiranju mogu se proceniti potencijalni uticaji na organizaciju i predložiti kontramere za smanjenje rizika [4].

Pristupi penetracionom testiranju uključuju sledeće tipove: crna, bela i siva kutija [5].

Penetracioni test crna kutija ne zahteva prethodne informacije o ciljnoj mreži ili aplikaciji i izvodi se kao stvarni napad hakera [6]. U ovom slučaju testeru nisu date informacije o unutrašnjem radu web aplikacije, niti o izvornom kodu ili softverskoj arhitekturi [6]. U penetracionom testu bela kutija, testeru se pruža pun pristup i informacije o unutrašnjoj strukturi sistema - izvornom kodu [7]. Izvorni kod aplikacije omogućava da se izvrši statička/dinamička „analiza izvornog koda“. Penetraciono testiranje siva kutija, nalazi se između testiranja crne i bele kutije, gde su neke informacije pružene, a neke skrivene [8]. Testerima se pruža dovoljno informacija za početak,

Kada je reč o kategoriji testiranja, važno je osvrnuti se i povući paralelu na tipove hakera koji postoje: crni, beli i sivi šešir.

Haker sa crnim šeširom svoje znanje koristi u negativne svrhe. Vođen je ličnom dobiti, uključujući zaradu iznudom ili drugim obmanjujućim metodama prikupljanja novca [9][10]. Haker sa belim šeširom se često naziva stručnjakom za bezbednost [10]. Takve hakere zapošljava organizacija i dozvoljeno im je da napadaju organizaciju kako bi pronašli ranjivosti koje bi napadač mogao iskoristiti [10]. Haker sa sivim šeširom predstavlja sredinu između belog i crnog šešira [9]. Na primer, haker sivog šešira radio bi kao stručnjak za bezbednost u organizaciji [9]. Međutim, može ostaviti ulaz da bi mu kasnije mogao pristupiti.

Postoji više metodologija penetracionog testiranja, ali je princip testiranja u svima veoma sličan. U ovom radu analiziran je pristup sa četiri faze:

Prva faza je prikupljanje informacija ili izviđanje i njen cilj je naučiti što više o klijentima [11]. Nakon svih prikupljenih informacija o meti, sledi faza skeniranja ranjivosti, koja obuhvata razumevanje kako će ciljna aplikacija odgovoriti na razne pokušaje upada [10]. Kada se završi faza skeniranja ranjivosti sledi eksploatacija ili iskorišćavanje ranjivosti. Ova faza predstavlja proces sticanja kontrole nad sistemom [10].

Za fazu eksploatacije veoma je važno da prethodne faze budu kompletirane, jer se uspeh ekploatacije zasniva na prethodno prikupljenim informacijama. Nakon završene eksploatacije neophodno je napraviti rezime u obliku izveštaja o izvršenom penetracionom testiranju [10].

Kad god je to moguće, prilikom pisanja detaljnog izveštaja o penetracionom testiranju, treba uključiti predloge za rešavanje problema koji su otkriveni. Pružanje rešenja za svaki problem koji se otkrije je vitalni deo svakog detaljnog izveštaja [10].

Uz temeljno analiziran teorijski deo, u nastavku je data njegova primena, kroz razvoj aplikacije kao alata koji će testirati bezbednost nekoliko web aplikacija.

4. RAZVOJ APLIKACIJE ZA TESTIRANJE BEZBEDNOSTI WEB APLIKACIJA KROZ PISANJE PENETRACIONIH TESTOVA

Aplikacija za testiranje implementirana je kao skup ručno pisanih skripti u *Python* programskom jeziku, unutar *Visual Studio Code* okruženja i namenjena je za sledeće vrste napada:

- *SQL Injection* (na prvom mestu OWASP TOP 10 liste)
- *Broken Authentication* (na drugom mestu OWASP TOP 10 liste)
- *Broken Access Control* (na petom mestu OWASP TOP 10 liste)

Testovi su implementirani uz pomoć *requests* i *BeautifulSoup Python* biblioteka.

Aplikacije koje se testiraju, u okviru pen test kućne laboratorije, pokrenute su na lokalu i sa aspekta bezbednosti predstavljaju aplikacije sa različitim nivoima zaštite, date sa ciljem testiranja bezbednosti i otkrivanja ranjivosti:

- *OWASP Juice Shop* – aplikacija sa predviđenim ranjivostima i izazovima različitih težina, namenjena vežbanju bezbednosti na web aplikacijama
- *Damn Vulnerable Web App* (DVWA) - namerno ranjiva web aplikacija, sa različitim nivoima težine i ciljem da pomogne stručnjacima za bezbednost da testiraju svoje veštine i alate, a programerima da bolje razumeju proces zaštite web aplikacije. Za potrebe testiranja odabran je visok nivo zaštite.
- *Mega Travel Booking* – web aplikacija za rezervaciju smeštaja, rađena kao studentski projekat

Pored manuelnih testova analizirani su i automatizovani testovi, uz pomoć poznatih, postojećih alata: *SqlMap* (za *SQL Injection*), *Wfuzz* (za *Broken Authentication*) i *Burp Suite Authz* (za *Broken Access Control*)

Kako bi se skupili svi neophodni podaci pre napada, konfiguriše se *proxy* u pregledaču i u *Burp Suite Proxy*-ju se prati filtriran saobraćaj između dve strane.

4.1 Test za *SQL Injection*

SQL Injection se sastoji od ubacivanja malicioznog SQL upita putem forme u aplikaciji. Uspešan napad rezultuje pristupom osetljivim podacima unutar baze podataka, kao i potencijalnom modifikacijom istih.

Ideja za testiranje *SQL Injection*-a kroz aplikaciju u *Python*-u pomenute tri aplikacije je sledeća: Sastaviti listu najpoznatijih malicioznih upita/fraza i za svaku od njih testirati jedno ili više unosnih polja u aplikaciji, kako bi se pokazalo da li je aplikacija ranjiva na ovaj napad, i ako jeste pokušati doći do podataka o korisnicima.

4.2 Test za *Broken Authentication*

Zaobiđena autentifikacija predstavlja ranjivost koja omogućava napadaču da prođe ili zaobiđe autentifikacione kontrole. On ima priliku da kompromituje lozinke, token sesije i da koristi implementacione mane kako bi preuzeo identitet korisnika. Do kompromitovanja sistema usled ove ranjivosti dolazi usled više faktora: korisnikovi kredencijali nisu bezbedno sačuvani ili su lako predvidivi, sesije se ne invalidiraju nakon određenog perioda…

Testiranje autentifikacione kontrole kroz test aplikacije zasniva se na sprovođenju *brute-force* napada na predviđene tri aplikacije. Polazna tačka za ovaj tip automatizovanog napada biće top 10000 najslabijih lozinki.

4.3 Test za *Broken Access Control*

Zaobiđena kontrola pristupa predstavlja ranjivost koja omogućava napadaču da pristupa neovlašćenim ili neautorizovanim funkcijama ili podacima kao što su pristup tuđem nalogu, pregled osetljivih podataka, izmena korisnikovih podataka, menjanje prava pristupa… Kontrole pristupa mogu se podeliti u tri široke kategorije: vertikalne, horizontalne i zavisne od konteksta:

- Vertikalne kontrole pristupa omogućavaju različitim vrstama korisnika pristup različitim delovima funkcionalnosti aplikacije [12]. U najjednostavnijem slučaju, ovo obično uključuje podelu između običnih korisnika i administratora
- Horizontalne kontrole pristupa omogućavaju korisnicima pristup određenom podskupu resursa iste vrste korisnika [12].
- Kontrole pristupa zavisne od konteksta osiguravaju da je pristup korisnika ograničen na ono što je dozvoljeno obzirom na trenutno stanje aplikacije [12].

Prilikom testiranja kontrole pristupa fokus će biti na eskalaciji horizontalnih privilegija i pristupu tuđim podacima.

5. REZULTATI TESTIRANJA

5.1 *SQL Injection*

Rezultati sa testiranja *SQL Injection*-a za polje pretrage *Owasp Juice Shop* aplikacije pokazuju da je moguće doći do podataka o korisnicima uz pomoć *union* naredbe, u kombinaciji sa podacima koji su dobijeni iz izuzetaka na *backend*-u. *SqlMap* alat uz odgovarajuće parametre u komandnoj liniji dolazi do tabela korisnika, ali bez mogućeg pristupa samim podacima.

Kada je u pitanju DVWA aplikacija, *Python* skripta, kreirana uz pomoć dva zahteva, s obzirom da se napad odigrava u dva različita prozora, rezultuje uspešnim napadom i dolaskom do podataka o korisnicima (imena i prezimena). *SqlMap* alat takođe dolazi do podataka o korisnicima, a u ovom slučaju i do tabela cele baze podataka, kao i korisnikovih lozinki.

Treća aplikacija, *Mega Travel Booking,* za polje pretrage smeštaja, pokazuje otpornost na maliciozne SQL upite iz manuelnog testa i za svaki zahtev vraća odgovor 422 – *Unprocessable entity.* Takođe i za *SqlMap* alat.

Kako bi se izbegle posledice *SQL Injection*-a, neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Validacija i sanitizacija ulaznih polja: definisanje odgovarajuće sintakse za ulazno polje forme i eskejpovanje (transformisanje) specijalnih karaketera poput ‘ / \ ” u sigurnu formu.
- Korišćenje alata/okruženja za objektno-relaciono mapiranje podataka. Obuhvata korišćenje parametrizovanih upita, pa pisanje upita za bazu u kodu nije neophodno.

5.2 *Broken Authentication*

Kod testiranja autentifikacije za *Owasp Juice Shop* i DVWA aplikaciju iskorišćena su dobijena korisnička imena iz testiranja *SQL Injection*-a, a za lozinku se koristi lista top 10000 najslabijih lozinki.

Test za *Owasp Juice Shop* aplikaciju rezultuje pronađenom lozinkom za administratorski mejl i uspešnim prijavljivanjem na sistem pod identitetom administratora. Do istih rezultata dolazi i *Wfuzz* alat.

Testiranje autentifikacije za DVWA aplikaciju nailazi na malo teži pristup, obzirom na prisustvo ANTI-CSRF tokena kod logovanja na sistem. Međutim, test pokazuje da je uz pomoć *BeautifulSoup* biblioteke moguće zaobići ovaj mehanizam zaštite i doći do pogođene lozinke iz liste, što ne važi i za *Wfuzz* alat, koji rezultuje neupešnim napadom.

Kada je u pitanju *Mega Travel Booking* aplikacija u testiranju *SQL Injection*-a nisu otkriveni podaci koji bi se mogli iskoristiti za ovaj napad, kao ni oni koji bi ukazali na *email* adrese korisnika prilikom prikupljanja informacija, pa pristup sa ovim testom nije moguć, kao ni sa testom *Wfuzz* alata.

Kako bi se izbegle evidentne posledice zaobiđene autentifikacije neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Lozinke ne čuvati u otvorenom tekstu, već koristiti više mehanizama. Čest oblik: *hash* funkcije primenjene na tekst lozinke kome se dodaje *salt* (niz *random* znakova) kako bi otežao *brute-force* napad.
- Implementirati provere za slabe lozinke
- Definisati minimalnu i maksimalnu dužinu i kompleksnost lozinke.
- Ograničiti ili odložiti neuspešne pokušaje logovanja. Vršiti *logging* svih neuspeha i obavestiti administratore kada su detektovane sumnjive aktivnosti koje mogu da nagoveste *brute-force* napade, a u tom slučaju ih stopirati.

5.3 *Broken Access Control*

Testiranje kontrole pristupa *Owasp Juice Shop* aplikacije svodi se na pokušaj pristupa tuđoj kupovnoj korpi. Preko ulogovanog korisnika i menjanjem parametra u URL-u test pokazuje da se može pristupiti podacima tuđe kupovne korpe. Podaci dobijeni u rezultatima testa poslužili su za još jedan tip testa – slanje žalbe u ime drugog korisnika, koji rezultuje ponovno narušenom

kontrolom pristupa i kreiranjem žalbe pod drugim identitetom. Alat *Burp Suite Authz* je takođe pokazao zaobilaženje kontrole pristupa, kreiranjem žalbe u ime drugog korisnika.

Kada je reč DVWA aplikaciji, ona ne nudi mogućnost direktnog napada na kontrolu pristupa u vidu eskalacije horizontalnih privilegija i stoga testiranje kontrole pristupa na ovu aplikaciju nije vršeno.

Za *Mega Travel Booking* aplikaciju test je vršen za kreiranje rezervacije, sa idejom umetanja tuđeg tokena prilikom slanja zahteva. Rezultati su pokazali izuzetak na *backend*-u, međutim na osnovu njih ne može se utvrditi da li ispitivana ranjivost postoji u aplikaciji. Kada je reč o *Burp Suite Authz* alatu, test je rezultovao zabranom pristupa traženim resursima (403 – *Forbidden*).

Kako bi se izbegle evidentne posledice zaobiđene kontrole pristupa neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Odbiti pristup za sve što nije javno dostupno (npr. neregistrovanim korisnicima).
- Podesiti da svi korisnici, programi, procesi imaju minimalno privilegija koliko je neophodno.
- Primeniti mehanizme kontrole pristupa koji će sprečiti neovlašćen pristup. Jedan od njih je RBAC model, često primenjiv.
- Vršiti *logging* svih neuspešnih kontrola pristupa i obavestiti administratore kad je to potrebno (npr. kad se isti neuspešni zahtevi ponavljaju).

6. ZAKLJUČAK

U ovom radu predstavljen je razvoj aplikacije za testiranje bezbednosti web aplikacija kroz prethodno analizirane koncepte bezbednosti web aplikacija i penetracionog testiranja istih. Testovi kroz koje se razvija aplikacija pisani su u *Python* programskom jeziku. Za testiranje su odabrane jedne od glavnih ranjivosti zastupljene u web aplikacijama danas: *SQL Injection*, *Broken Authentication*, *Broken Access Control*. Testirane su tri web aplikacije u tri scenarija.

Kada je reč o SQL *Injection*-u, testovi su pokazali uspešnu eksploataciju ranjivosti za prvu i drugu aplikaciju, a odabrani *SqlMap* alat kao paralela manuelnim testovima, to pokazuje za takođe iste aplikacije.

Testovi za autentifikaciju su izvedeni kao *brute-force* napadi i rezultuju uspešnim zaobilaženjem autentifikacije za prve dve aplikacije, dok *Wfuzz* alat namenjen *brute-force* napadu to čini samo za prvu. Treći tip testa namenjen kontroli pristupa, pokazao je da je samo prva aplikacija ranjiva za eskalaciju horizontalnih privilegija, i kroz manuelni i kroz *Burp Suite Authz* automatizovani alat.

Ono što se primećuje skoro u svim uspešnim napadima jeste nedostatak evidentiranja događaja, koje bi ukoliko je adekvatno implementirano upozorilo na sumnjive aktivosti.

Na kraju su izložene mere zaštite za testirane ranjivosti.

Može se reći da je za dati scenario i odabrane automatizovane alate, aplikacija u razvoju za testiranje, kao manuelni alat, pokazala bolju efikasnost pri otkrivanju ranjivosti. Ovakav način testiranja pruža testeru mogućnost da bolje analizira situaciju, razmišlja kako i gde bi haker mogao da izvrši napad.

Dalje istraživanje u razvoju aplikacije obuhvata proširenje trenutnih penetracionih testova, poput eskalacije vertikalnih privilegija kod zaobiđene kontrole pristupa i *Credential stuffing* napada kod zaobiđene autentifikacije. Takođe, budući rad uključuje i testiranje drugih ranjivosti poput *Cross-site scripting* napada. Preporučuje se i uvođenje drugih alata, radi poređenja rezultata sa trenutnim stanjem i dolaska do što preciznijih zaključaka.

7. LITERATURA

[1] Ahmed, S., "Secure Software Development: Identification of Security Activities and Their Integration in Software Development Lifecycle", *School of Engineering Blekinge Institute of Technology, Ronneby, Sweden*, 2007

[2] Wolf Halton, Bo Weaver, Juned Ahmed Ansari, Srinivasa Rao Kotipalli, Mohammed A. Imran, *"Penetration Testing: A Survival guide"*, Packt Publishing Ltd, 2016

[3] Owasp TOP 10 – 2017, OWASP Foundation, 2017

[4] Mansour Alharbi, "Writing a Penetration Testing Report", *SANS Institute – Information Security Reading Room*, 2010

[5] https://purplesec.us/types-penetration-testing/#Involve (pristupljeno u septembru 2020.)

[6] https://resources.infosecinstitute.com/the-types-of-penetration-testing/#gref (pristupljeno u septembru 2020.)

[7] Kassem A. Salech, *"Software Engineering"*, J. Ross Publishing, 2009

[8] Joseph Muniz, Aamir Lakhani, *"Web Penetration Testing with Kali Linux"*, Packt Publishing Ltd, 2013

[9] Rafay Baloch, *"Ethical Hacking and Penetration Testing Guide"*, CRC Press, 2017

[10] Patrick Engebretson, *"The Basics of Hacking and Penetration Testig - Ethical Hacking and Penetration Testing Made Easy"*, Elsevier, 2011

[11]Georgia Weidman, *"Penetration Testing – A Hands-On introduction to Hacking"*, William Pollock, 2014

[12] Dafydd Stuttard, Marcus Pinto, *"The Web Application Hacker's Handbook – Finding and Exploiting Security Flaw, Second Edition"*, John Wiley & Sons, Inc, 2011

Kratka biografija:

Ivana Marin rođena je u Novom Sadu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranila je 2020.god.
kontakt: ivanamarin67@gmail.com