Penetraciono testiranje

Penetraciono testiranje spada u etičko hakovanje i podrazumeva testiranje računarskih sistema, mreže ili veb aplikacija kako bi se pronašle ranjivosti koje haker može da iskoristi i zloupotrebi. Penetraciono testiranje se može automatizovati pomoću softverskih alata ili se može vršiti ručno. U oba slučaja, proces obuhvata prikupljanje informacija o ciljanoj aplikaciji pre testiranja, identifikaciju mogućih ulaznih tačaka i pokušaj ulaska u sistem. Na kraju je potrebno dokumentovati pronađene ranjivosti u obliku izveštaja.

Izveštaji koje pentesting izgeneriše treba da daju povratnu informaciju organizacijama čije aplikacije se testiraju kako bi trebalo postaviti prioritete što se tiče bezbednosti. Takođe, mogu da pomognu developerima kako bi napravili što bezbednije aplikacije. Namera je motivisati developere da prošire postojeće znanje u domenu bezbednosti kako bi se detektovane ranjivosti sanirale u budućim projektima. Za penetraciono testiranje rent-a-car sistema će se koristiti sqlmap i owaspZAP alati.

SqlMap

Ovaj open-source alat za pentestiranje omogućava automatizaciju detektovanja i zloupotrebe SQL injection nedostataka i preuzimanje servera baze podataka. Neke od mogućnosti ovog alata su:

- Puna podrška za mySQL, postgresSQL, microsoftSQL, MariaDB, H2 (i mnoge druge) sisteme za upravljanje bazom podataka
- Direktno povezivanje sa bazom podataka pomoću DBMS kredencijala, IP adrese i porta i naziva baze podataka
- Može da vrši enumeraciju korisnika, heševa lozinki, permisija, uloga, tabela i kolona
- Automatski prepoznaje format hash lozinke i podržava dictionary-based napade

OwaspZAP

Besplatan open-source alat za početnike u pentestiranju koji omogućava otkrivanje ranjivosti u web aplikacijama. ZAP kreira proxy web server kroz koji prolazi sav „saobraćaj“ veb aplikacije. Za presretanje zahteva i otkrivanje ranjivosti se koristi automatski skener, koji ima mogućnost generisanja izveštaja. ZAP se ponaša kao middle-man između browsera i veb aplikacije kako bi presretao poruke i menjao njihov sadržaj po potrebi.

U nastavku su navedene detektovane potencijalne ranjivosti pomoću ovih alata i priložena su rešenja. Za testiranje su korišćena tri tipa korisnika: prodavac sa validnom lozinkom, koji ima autorizovan pristup određenom delu sistema, kupac koji takođe ima autorizovan pristup delu sistema i jedan nevalidan korisnik koji ne postoji u sistemu. Testirane su sve postojeće putanje koje su prošle kroz proxy, sa ispravnim i neispravnim Authorization hederom. Takođe, testirani su i zahtevi koji ne sadrže Authorization header ili je on podešen na null. Svi neovlašćeni zahtevi su vraćeni sa 401 i 403 porukom greške što ukazuje na to je je kompletan sistem autorizovan i

nije moguće pristupiti metodama i stranicama ukoliko korisnik nije prijavljen sa odgovarajućim kredencijalima i ima definisane permisije za tu stranicu, komponentu ili metodu.

- **Path Traversal (nivo rizika: VISOK)**

Ova vrsta napada omogućava hakerima da pristupe fajlovima, direktorijumima i komadama koje se potencijalno nalaze van korenskog direktorijuma veb aplikacije. Najčešće se napadi vrše sa nizom karaktera kao što su "../" i njegove alternative. Pored ovog, česta je upotreba i nevalidnih UniCode-encoding unosa koji umesto kose crte sadrže % (%2e%2e%2f, ..%255c…).

- Detektovani problemi: https://localhost:8099/advertisement/addNew/1
  parametri koji nisu zaštićeni od ovakvog unosa - *kidsSeats, max_price, mileage, mileageLimit, min_price, rating, startDate, city*

- Rešenje: Iz data transfer objekata koji se šalju i prihvataju kroz http zahteve su uklonjeni parametri koji se ne koriste (max_price, min_price). U svakoj metodi gde se ovi parametri koriste, prethodno je pozvana validacija nad poljima pomoću regex izraza. Svaki izraz je formiran tako da proverava dužinu, tip unosa, opseg, dozvoljene vrednosti i karakteristike pojedinačnog parametra. Sprovedena je provera validacije svakog polja u svakom objektu aplikacije, i svakog parametra koji se šalje kroz HTTP zahtev. Validacija svakog parametra se vrši pre pristupa bazi podataka.

- **SQL injection (nivo rizika: VISOK)**

Vrsta napada u kojem se ubrizgava SQL kod koji može da manipuliše podacima u korištenoj bazi podataka, naruši njihov integritet ili ih učini nedostupnim.

- Detektovan problem:
  https://localhost:8099/advertisement/search?startDate=1593380700000&endDate=1593749160000&city=ns+AND+1%3D1+--+
  uspešno izvršen napad nad city parametrom sa [AND 1=1 --] i [AND 1=2 –]

- Rešenje: Ovo je jedini parameter zahteva kod kog je detektovan sql napad, proverena je validacija koja prihvata samo karaktere za vrednost ovog parametra i pozvana je na odgovarajućem mestu. Pored validacije na bekendu, urađen je i escaping ovog parametra pre pristupa bazi podataka.

- **Buffer Overflow (nivo rizika: SREDNJI)**

Ovaj tip greške karakteriše prekomerno upisivanje u memorijske lokacije pozadinskih procesa kojima nije dozvoljeno pristupati. Najčešće ovaj tip napada dovodi do zaustavljanja aplikacije ili njenog nepredviđenog ponašanja.

- Detektovan problem: https://localhost:8099/advertisement/addNew/1
  Ugrožen parameter je carClass, zahtev se vratio sa 500 Internal server error kodom greške za zahtev čiji je content-length: 2398

- Rešenje: Java programski jezik spade u grupu interpretiranih porgramskih jezika koji su imuni na ovu vrstu napada, osim ukoliko je sam interpreter ugrožen. S obzirom da je kompletan kod backend aplikacije napisan u Javi, rizik od prolaza ovog napada je smanjen na minimum. Kako bi se aplikacija dodatno obezbedila, za svaki response je izvršena provera dužine odgovora koji dolazi sa backend-a

➢ Cross-Domain Misconfiguration (nivo rizika: SREDNJI)

Pogrešna CORS konfiguracija na veb serveru dozvoljava čitanje zahteva sa proizvoljnih domena trećih strana (third party domains) koristeći neovlašćene API-je na ovom domenu. Veb pretraživači koji se danas koriste podržavaju Same-Origin politiku i onemogućavaju trećim stranama da čitaju odgovor ovakvih zahteva, što donekle smanjuje rizik od napada i svrstava ga u srednji nivo.

- Detektovan problem: Access-Control-Allow-Origin: *

- Rešenje: postavljen Access-Control-Allow-Origin header svakog zahteva na https://localhost:8099 kako bi se dozvolilo slanje i čitanje zahteva samo sa tog domena. Dodatno, u konfiguracionoj klasi aplikacije je postavljen allowOrigins parameter na vrednost url-a sa kod dolaze zahtevi (a to je localhost:8099)

➢ X-Frame-Options Header Not Set (nivo rizika: SREDNJI)

X-Frame-Options header je deo HTTP odgovora i može da se koristi kao indikator koji će pretraživaču dati dozvolu da renderuje stranice u `<frame>`, `<iframe>`, `<embed>` ili `<object>` tagovima. Postavljanje ovog hedera onemogućava napadačima da ugrade svoj sadržaj na tuđe stranice.

- Detektovan problem: ne postoji X-Frame-Options header u HTTP response-u

- Rešenje: X-Frame-Options: DENY
  S obzirom da naša aplikacija ne zahteva renderovanje u okviru nekog <iframe> taga, postavljen je X-Frame-Options heder na DENY vrednost što će sprečiti renderovanje u <iframe> kako sa drugih sajtova, tako i iz naše aplikacije i onemogućiti izvršavanje Click-jacking napada.

➢ Web Browser XSS Protection Not Enabled (nivo rizika: NIZAK)

Veb pretraživači imaju svoj mehanizam zaštite od XSS napada koji treba da bude omogućen. Ovaj problem se detektuje kod zahteva u čijem odgovoru se može naći xss sadržaj. Da bi se omogućio ovakav mehanizam zaštite, potrebno je postaviti X-XSS-Protection: 1.

➢ Cross Site Scripting Weakness (Reflected in JSON Response) (nivo rizika: NIZAK)

XSS napad reflektovan in JSON odgovoru koji može dovesti do ranjivosti korisnika koji će konzumirati ovaj odgovor. U ovom slučaju je svrstan u nizak nivo opasnosti iz razloga što Content-Type nije HTML.

- Detektovani problemi:
  https://localhost:8099/advertisement/addNew/1
  Aftektovani parametri: *kidsSeats, endDate, startDate, mileageLimit*
  https://localhost:8099/pricelist
  aftektovani parametri: *exceedMileag, collisionDW, discount20, discount30, priceDay*
  https://localhost:8099/request
  aftektovani parametri: *startDate, endDate*
  Za sve navedene parametre je uspešno izvršen napad sa ubrizgavanjem skripte
  <script>alert(1);</script>

- Rešenje: Aplikacija već koristi OwaspEncoder klasu koja radi escaping String parametara za html. Pored ove, pozvana je i metoda forJavaString(String input) čija dokumentacija garantuje da je bezbedna za upotrebu za HTML atribute, JSON objekte, javascript skripte i blokove. Svaki parametar ove metode je okružen navodnicima kako bi se escaping uradio na pravilan način. Proširen je poziv ove klase i na parametre drugog tipa (koji nisu string). Dodatno, proverena je i dopunjena validacija ovih parametara na frontu.

  OwaspZAP alat je pokušao i tri vrste XSS persistent napada. Napadi u kojima se ubacena maliciozna skripta trajno cuva na serveru (u bazi podataka, u polju komentara i sl...) i poziva se svaki put kada korisnik "pogodi" odgovarajuću funkcionalnost ili pri svakom ucitavanju stranice. Ovakav napad se smatra rizičnijim, izaziva više štete većem broju korisnika. Da bi se izvrsio persistent XSS napad, potrebno je implementirati maliciozni kod u neko input polje (neki primeri su: forma, pretraga, komentar). Za XSS persistent napade je korišten i ajax spider iz owaspZAP-a sa ukupno pokušanih 1017 zahteva od kojih ni jedan nije prošao uspešno što pokazuje da je aplikacija zaštićena od ove vrste XSS napada.

➢ Application Error Disclosure (nivo rizika: NIZAK)

Stranica na kojoj je detektovan ovaj problem sadrži ispis greške ili upozorenja koji napadaču mogu otkriti lokaciju fajlova i otvara mogućnost napadaču da dublje istraži ispis i izazove ozbiljnije napade.

- Detektovan problem: ispis greške na https://localhost:8099/car

- Rešenje: Proveriti i kod greške u slučajevima kada se zahtev ne izvrši uspešno i u zavisnosti od vrste greške redirektovati korisnika na implementirane stranice za grešku (Error404 i Error500).

➢ Information Disclosure - Debug Error Messages (nivo rizika: NIZAK)

Odgovor sadrži uobičajene debug poruke karakteristične za .ASP NET platformU, IIS ili Apache veb servere.

- Detektovan problem: https://localhost:8099/car ispisuje Internal Server Error debug poruku ako dođe do greške

- Ova aplikacija je podignuta na Apache veb server i moguće je da je ovaj vid ranjivosti false-positive. Kako bismo bili sigurni da se debug poruke neće prikazivati korisnicima potrebno je onemogućiti ispis debug poruka prilikom postavljanja aplikacije u produkciju.

➢ X-Content-Type-Options Header Missing (nivo rizika: NIZAK)

Omogućava starijim verzijama Internet Explorera i Chrome-a MIME-sniffing nad sadržajem odgovora sa backend-a, što potencijalno može da dovede do pogrešne interpretacije odgovora, u drugom formatu u odnosu na očekivani.

- Detektovan problem: zahtevi ne sadrže X-Content-Type-Options header

- Rešenje: postaviti X-Content-Type-Options: nosniff za sve stranice aplikacije, uključujući i one koje služe za error handling jer su i one podložne skreniranju sadržaja i napadima.