Penetracijsko testiranje

Šešet, Antonio

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

PENETRACIJSKO TESTIRANJE

Rijeka, Rujan 2024.

Antonio Šešet

0069082419

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

PENETRACIJSKO TESTIRANJE

Mentor: prof. dr. sc. Mladen Tomić

Rijeka, Rujan 2024.

Antonio Šešet

0069082419

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14.03.2024.

Zavod: Zavod za računarstvo
Predmet: Računalne mreže

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Antonio Šešet (0069082419)**
Studij: Sveučilišni prijediplomski studij računarstva (1035)

Zadatak: **Penetracijsko testiranje / Penetration Testing**

Opis zadatka:

Napraviti pregled problematike penetracijskog testiranja. Objasniti vrste i faze penetracijskog testiranja. Istražiti i prikazati analizu aktualnih alata koji se koriste za testiranje te njihovih mogućnosti i područja primjene. Na stvarnom primjeru virtualne okoline izložiti provedbu testa te analizirati dobivene podatke i rezultate.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Mentor:
prof. dr. sc. Mladen Tomić

Predsjednik povjerenstva za završni ispit:
prof. dr. sc. Miroslav Joler

IZJAVA

kojom ja, Antonio Šešet, JMBAG: 0069082419 student Tehničkog fakulteta Sveučilišta u Rijeci, kao autor završnog rada s naslovom: Penetracijsko testiranje:

1. Izjavljujem da sam završni rad izradio/la samostalno pod mentorstvom prof. dr. sc. Mladena Tomića. U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u radu citirao sam i povezao s korištenim bibliografskim jedinicama sukladno odredbama Pravilnika o završnom radu, završnom ispitu i završetku preddiplomskih sveučilišnih studija Tehničkog fakulteta u Rijeci.

2. Dajem odobrenje da se, bez naknade, trajno objavi moj završni rad u roku od 30 dana od dana obrane na nacionalnom repozitoriju odnosno repozitoriju visokog učilišta, sukladno obvezi iz odredbe članka 58. stavka 5. Zakona o visokom obrazovanju i znanstvenoj djelatnosti (NN 119/22).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog rada. Ovom izjavom, kao autor dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim studentima i djelatnicima ustanove.

Antonio Šešet

SADRŽAJ

1. UVOD

1.1. Povijest Web sigurnosti i rani početci Interneta

Još od samih začetaka ideje Interneta čija je preteča bio ARPANET, jedan od najvažnijih ciljeva bio je kako osigurati sigurnost infrastrukture i podataka, te omogućiti sigurnu i zaštićenu komunikaciju svim korisnicima koji su koristili taj sustav. Pitanje sigurnosti je bilo od iznimne važnosti za ARPANET s obzirom da je cjelokupni sustav bio namijenjen isključivo za potrebe američke vlade i vojske, a kasnije je bio dostupan za korištenje u akademske svrhe povezane s američkom vladom, no nikad nije bio dostupan širokom pučanstvu kao što je to Internet danas. Unatoč tome što ARPANET danas više nije u upotrebi, ideje o zaštiti i sigurnosti koje su začete još davnih 1960-ih i 1970-ih kada se ARPANET razvijao prenesene su i u moderno doba i urezale su put modernoj web sigurnosti.

S aspekta sigurnosti, ARPANET je imao veliku prednost naspram današnjeg Interneta jer je bio zatvorena mreža gdje je pristup bio dozvoljen samo autoriziranim korisnicima koji su prije svakog korištenja ARPANET-a morali proći autentifikaciju kako bi se mogli služiti tom mrežom što je ujedno i uvelike pridonosilo sigurnosti na samoj mreži.[1] Danas bi takva sigurnosna procedura bila nezamisliva običnom korisniku koji Internet koristi za usluge kao što su World Wide Web, e-pošta, društveni mediji ili instant poruke.

No kako je vrijeme prolazilo, brojni članovi akademske zajednice koji su sudjelovali na razvoju ARPANET-a su uvidjeli brojne prednosti takvog sustava za razmjenu informacija koje bi se mogle koristiti i izvan vojnih okvira u svrhu lakše razmjene informacija među običnim ljudima.

Početkom 1990-ih nastao je Internet kakvog znamo danas kada je svijetu predstavljen World Wide Web (WWW) kojeg je razvio Tim Berners-Lee, znanstvenik u CERN-u. WWW je običnim ljudima i poduzećima omogućio jednostavnu navigaciju i razmjenu informacija koristeći web preglednike.[2]

Paralelno s razvojem WWW-a, 1990-ih došlo je do ogromnog rasta broja osobnih računala u svijetu, a ujedno je veliki broj kompanija i ljudi počeo nabavljati osobna računala i servere kako bi se mogli umrežiti sa svijetom i lakše komunicirati i razmjenjivati podatke, te je tako broj korisnika Interneta strmoglavo rastao.

No kako je sve veći broj ljudi počeo koristiti Internet, počela se javljati sve veća potreba da se uvedu napredniji sigurnosni mehanizmi koji bi zaštitili sigurnost i privatnost korisnika koji koriste Internet, kako običnih ljudi tako i poduzeća.

1.2. O važnosti zaštite web aplikacija u današnjem dobu i web sigurnosti

Danas ogroman broj poduzeća i pojedinaca koristi Internet kako bi pružali svoje usluge. Neki od čestih primjera su banke koje imaju tzv. Internet bankarstvo, drugi primjer bile bi brojne trgovine koje prodaju svoju robu preko tzv. „Internet trgovina“, postoje još razne društvene mreže i velik broj različitih usluga i servisa koji svoje djelatnosti obavljaju upravo putem Interneta, ali ono što je zajedničko svemu navedenom je da su to zapravo web aplikacije.

Kada govorimo o web aplikacijama, govorimo o osnovnoj infrastrukturi koja se koristi za poslovanje, razmjenu informacija i komunikaciju. Uz web aplikacije blisko vežemo pojam web sigurnosti u koju ubrajamo sve alate, strategije i djelovanja koja služe kako bi se rad web aplikacija mogao normalno i neometano izvoditi, te kako bi se zaustavile prijetnje kao što su hakiranje, krađa identiteta, zloupotreba podataka i ostali oblici napada koji mogu nanijeti materijalnu i nematerijalnu štetu pojedincima i kompanijama koji koriste i pružaju usluge na Internetu.

Ciljevi koje web sigurnost mora ispuniti su zaštita osjetljivih podataka, očuvanje povjerenja korisnika, usklađenost s propisima, održavanje neprekinutog poslovanja i rada web aplikacije i sprječavanje financijskih gubitaka. Stoga je razumijevanje principa i metoda web sigurnosti od iznimne važnosti za sigurnost interneta i ljudi koji ga koriste, a posebice za programere i razvojne inženjere koji razvijaju web aplikacije koje će se u budućnosti naći na internetu.

Postoje brojni alati, mehanizmi i tehnike koje se danas koriste kako bi se osigurala i ispitala sigurnost web aplikacija, neke od njih su Skeniranje ranjivosti, Sigurnosne revizije, tzv. „*Bug bounty*“ programi (programi gdje hakeri dobivaju nagradu ako pronađu sigurnosni nedostatak u aplikaciji) itd., no u sklopu ovog rada opisivati će se Penetracijsko testiranje kao jedan od tih mehanizama.

2. PENETRACIJSKO TESTIRANJE

2.1. Što je penetracijsko testiranje i čemu služi?

Penetracijsko testiranje jedan je od brojnih mehanizama koji se danas koriste kako bi se osigurala sigurnost i neometan rad web aplikacija koje se nalaze na internetu. Alternativni nazivi koji se koriste za ovaj mehanizam su još etičko hakiranje, „*engl. White hat hacking*“ i „ *engl. pen-testing*“.

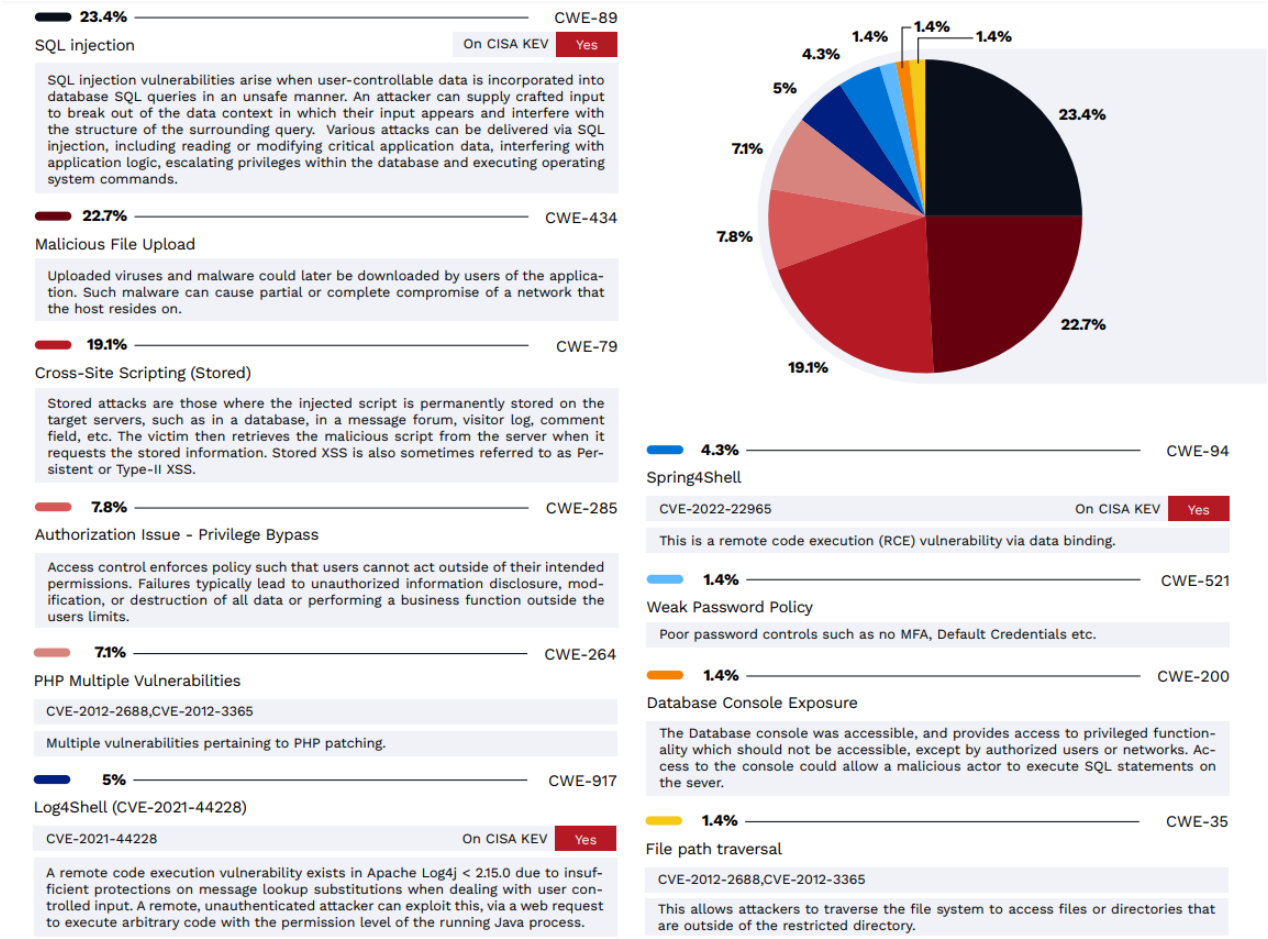
Glavna ideja penetracijskog testiranja je otkriti ranjivosti nekog računalnog sustava i ukloniti ih na vrijeme, odnosno prije nego što bi ih stvarni napadači mogli iskoristiti. Preciznije, kroz simuliranje stvarnog kibernetičkog napada nastojimo ustanoviti postoje li u sustavu ranjivosti koje napadači mogu iskoristiti za napade. Ukoliko ranjivosti postoje, one se evidentiraju i prijavljuju, te se daju preporuke kako ih otkloniti i spriječiti slične propuste.

Ranjivosti u sustavu su većinom posljedica nepažnje ili neznanja, te se najčešće manifestiraju kao sigurnosni propusti koji se potkradaju prilikom razvoja programskog proizvoda. Ono što ovakve propuste čini težima za otkriti jest da ukoliko sam programer nije svjestan da čini propust, često ne može dobiti direktnu povratnu informaciju da postoji sigurnosna nepravilnost kao što bi dobio da primjerice ima sintaktičku pogrešku u kodu. Uz te sigurnosne nepravilnosti, nedovoljna ili loša implementacija sigurnosnih mehanizama, a u rijetkim slučajevima i namjeran propust doprinose broju ranjivosti koje napadači mogu iskoristiti. Penetracijskim testiranjem nastojimo otkriti te ranjivosti koje bi napadačima dozvolile i olakšale izvođenje napada.

Prema podacima iz izvještaja „*2023 Vulnerability Statistics Report, 8th Edition*“ kompanije Edgescan koja se bavi pružanjem usluga testiranja sigurnosti i upravljanjem ranjivostima u IT sistemima, jedna od najčešćih vrsta napada i propusta je „*SQL injekcija*“ koja omogućuje neovlašten pristup bazi podataka, te izvršavanje zlonamjernih skripti na SQL serveru koje zauzvrat omogućuju pristup podacima koji ne bi smjeli biti dostupni ili vidljivi. Ukoliko do toga dođe, napadači mogu brisati podatke ili ih mijenjati i na taj način steći neovlaštene privilegije ili poremetiti rad sistema.

Nadalje, „*Reflected XSS (Cross site scripting)“* jedna je od najčešćih ranjivosti i nastaje kada aplikacija nesigurno prikazuje podatke iz zahtjeva te na taj način omogući napadaču da pokrene zlonamjerni JavaScript kod u pregledniku korisnika

Uz navedene propuste, prijenosi zlonamjernih datoteka, ranjivosti u sustavu za autorizaciji i upravljanju sesijama, loše konfigurirane sigurnosne postavke, ranjivosti u komponentama trećih strana (vanjske knjižnice, API-jevi itd.) uvelike olakšavaju napadačima u stjecanju neovlaštenog pristupa i nanošenju štete vlasnicima web aplikacija i kranjim korisnicima.[3]

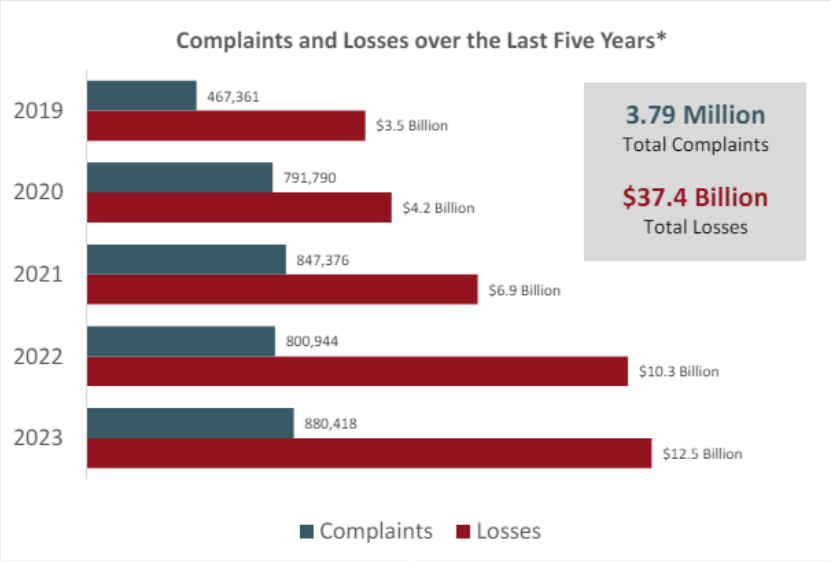


Slika 2.1. Grafički prikaz kritičnih sigurnosnih ranjivosti iz dokumenta "2023 Vulnerability Statistics Report - 8th edition" kompanije Edgescan

Sam proces penetracijskog testiranja složen je i kompliciran, mora biti isplaniran s puno pažnje i uzeti u obzir velik broj čimbenika koji su važni za programski proizvod koji se testira, a osobe koje su zadužene za planiranje i provođenje samog napada moraju biti upoznate s najnovijim tehnologijama koje stvarni napadači koriste, te moraju posjedovati veliku količinu znanja o tehnologijama koje se koriste za razvoj programskih proizvoda koje testiraju kao i o ranjivostima istih.

2.2. Potreba za penetracijskim testiranjem

U zadnjih nekoliko godina količina kibernetičkih napada strmoglavo je porasla. Iako su kibernetički napadi postojali još od samih začetaka javnog Interneta, početak naglijeg rasta napada dogodio se u razdoblju pandemije korona virusa kada je veliki broj poduzeća morao prijeći na Internet poslovanje i od tad je nastavio rasti svake godine. Prema podacima američkog FBI-ja u dokumentu „*Internet Crime Report 2023"* se može vidjeti da se materijalna šteta u 2023. godini u odnosu na 2019. gotovo učetverostručila i iznosila je 12,5 milijardi američkih dolara, a ako pridodamo štetu od četiri godine prije dolazimo do ukupne štete koja iznosi čak 37.4 milijardi američkih dolara. Ova brojka se temelji na prosječno 758 tisuća prijava koje FBI zaprimi svake godine.[4]



Slika 2.2 Grafički prikaz podataka o napadima iz dokumenta Internet Crime Report 2023.

Iako financijski gubici nisu jedini pokazatelj posljedica kibernetičkih napada, predstavljaju dobar i lako shvatljiv primjer koliko štete takvi napadi mogu uzrokovati. Uz financijske gubitke, dodatni oblici štete mogu biti otkrivanje povjerljivih informacija, nedostupnost usluge, gubitak povjerenja klijenata, gubitak važnih podataka i poremećaj u normalnom radu sustava.

Brojne kompanije, pojedinci ali čak i države institucije[5] su bili žrtve ovakvih napada u bližoj prošlosti s ciljem kako bi napadači stekli određenu materijalnu korist ili ostvarili vlastitu agendu. Upravo zbog ovog negativnog trenda u porastu napada, javlja se sve veća potreba za korištenjem sigurnosnih mehanizama a s time i penetracijskim testiranjem kao jednog od tih mehanizama.

3. POSTUPAK I IZAZOVI PENETRACIJSKOG TESTIRANJA

3.1. Postupak penetracijskog testiranja

Prije samog testiranja, nužno je izraditi plan testiranja koji će jasno definirati koji su ciljevi testiranja i koje metode će se koristiti. Prvo se planira opseg testiranja; koji dijelovi proizvoda su najbitniji za testiranje, koliko dugo će testiranje trajati, ali isto tako se pri planiranju uzima u obzir da samo testiranje ne smije narušiti ostale poslove poduzeća i poslovne operacije koje se trebaju normalno odvijati dok traje testiranje.

Nakon što se razvije i odobri plan kako će se testiranje provoditi, kreće se s postupkom. Prvi korak je s alatima za skeniranje identificirati moguće ranjivosti i prikupiti informacije o sistemima koji su cilj testiranja. Alati koji se najčešće koriste za ovaj korak su alati za mapiranje mreže i skeniranje portova.

Kada su sve informacije prikupljene i sve ranjivosti identificirane, kreće se s iskorištavanjem istih kako bi testeri dobili neovlašteni pristup ili izazvali drugu vrstu štete. Primjeri drugih vrsta štete mogu biti ubacivanje zlonamjernih skripti, izmjena i brisanje podataka u bazi podataka, obustavljanje rada aplikacije itd. Cilj ovog dijela penetracijskog testiranja je pokazati kako bi stvarni napadači mogli iskoristiti te ranjivosti protiv vlasnika programskog proizvoda i nanijeti mu nekakav oblik štete.

Po završetku testiranja testeri rade detaljno izvješće u kojem navode sve ranjivosti koje su pronašli, detaljno opisuju načine kako su iskoristili pronađene ranjivosti i ono najvažnije: daju preporuke kako ranjivosti ukloniti ili kako ih umanjiti ukoliko uklanjanje istih nije moguće.

Kada naručitelj testiranja dobije izvješće, radi na ispravljanju ranjivosti i eventualnih ostalih nedostataka u programskom proizvodu. Nakon što se ranjivosti isprave, kreće se u ponovno testiranje kako bi se utvrdilo jesu li se ranjivosti zaista uklonile i jesu li su eventualno pojavile nove ranjivosti ili sigurnosni nedostatci.

3.2. Izazovi penetracijskog testiranja

U prethodnom potpoglavlju je ukratko opisano kako izgleda postupak penetracijskog testiranja, no sam postupak pred testere stavlja velik broj tehničkih, etičkih i ostalih izazova koji se moraju savladati kako bi se testiranje uspješno izvršilo.

Kada govorimo o tehničkim izazovima, govorimo o činjenici da se testeri danas moraju suočiti s web aplikacijama koje su svakim danom sve složenije. Danas se standardno za razvoj web aplikacija koriste složene arhitekture koje se sastoje od više razina od koje svaka ima svoju važnu ulogu, ali sljedeće tri razine su „temelj“ većine web aplikacija: korisničko sučelje (engl. „*frontend*“), pozadinski dio sustava (engl. „*backend*“) i baza podataka koja služi za pohranu podataka koji su nužni za rad web aplikacije

Uz navedene razine, mogu se još koristiti razni vanjski API-jevi, cloud usluge itd.. Svaka od navedenih razina može sadržavati neku ranjivost i uvesti dodatne ranjivosti u cjelokupan sustav. Stoga je od ključne važnosti da testeri imaju znanje o cijeloj arhitekturi, ali i tehnologijama koje ju čine i njihovim ranjivostima. Iako se većina web aplikacija danas temelji na sličnoj arhitekturi koja je prethodno opisana, nisu sve web aplikacije napravljene korištenjem istih tehnologija i alata. Naime, postoji veliki broj različitih alata, tehnologija i pristupa koji se koriste za razvoj, bilo da je riječ o programskom jeziku, frameworku ili sigurnosnim alatima.

Tester koji obavlja testiranje web aplikacije mora posjedovati znanje o svim navedenim čimbenicima i mora biti sposoban brzo učiti i prilagođavati se novim trendovima i tehnologijama na tržištu. Upravo je to znanje i sposobnost praćenja promjena potrebno kako bi se uspjelo pokriti sve moguće sigurnosne aspekte aplikacije, čak i ona mjesta koja su manje očita i skrivena, sve s ciljem da se sva područja aplikacije pravilno testiraju i da se ne izostave nikakve ranjivosti prilikom testiranja.

Važno je napomenuti da pored „običnog“, odnosno ljudskog testiranja, testeri danas također koriste automatizirane alate koji se zovu skeneri i oni služe za pronalazak i identifikaciju slabosti unutar web aplikacija, programa i općenite IT infrastrukture. Skeneri automatski pretražuju web aplikaciju s ciljem pronalaska ranjivosti koje bi napadači mogli iskoristiti. Njihov rad se sastoji od mapiranja web aplikacije (prolazak kroz sve stranice i obrasce, te funkcionalnosti), identifikacije ranjivosti (prethodno opisane poput „SQL injekcije“, XSS itd.) i izvještavanju. Značajno ubrzavaju rad testera, konzistentni su i imaju širok opseg testiranja ali ih je isto potrebno stalno nadograđivati i unaprjeđivati kako bi mogli otkriti nove prijetnje i ranjivosti koje su se tek pojavile.

Kako je ranije navedeno za testere, isto vrijedi i za skenere; da bi skener bio što učinkovitiji važno je da čovjek ili tim koji ga razvija bude dobro educiran i informiran o svim mogućim ranjivostima s kojima se skener može susresti, te da pravilno konfiguriraju i razviju skener.

Potrebno je biti svjestan činjenice da skeneri mogu dati pogrešne rezultate u obliku lažno pozitivnih rezultata (engl. „*false positive*“, pojave kada skener javlja grešku iako je nema) i mogu iziskivati intervenciju testera. Tester mora poznavati prednosti i mane skenera koji koristi, sve aspekte web aplikacije koju testira i biti sposoban prepoznati lažne pozitivne rezultate.[6]

Uz prethodno opisane tehničke izazove, važno je obratiti pažnju na etičke i legalne izazove s kojima se testeri susreću. Svaki tester koji provodi penetracijsko testiranje mora imati pravne ovlasti i odgovarajuće dozvole od vlasnika aplikacije koja se testira kako bi proveo testiranje. Mora se jasno definirati kako će se testiranje provoditi, na koje dijelove aplikacije testeri moraju obratiti posebnu pozornost i kojim dijelovima aplikacije testeri smiju pristupiti.[7]

Od iznimne je važnosti da testeri odgovorno rukuju s podacima koje otkriju prilikom testiranja, kao i s izvještajima koji se kasnije rade. Preciznije, ako se testira aplikacija koju korisnici već koriste i nije u fazi razvoja, već sadrži osjetljive podatke korisnika, bitno je da podaci na koje testeri naiđu ne budu zloupotrebljavani, te da se zadrži i ne naruši privatnost krajnjih korisnika. Veoma je bitno da testeri i poduzeća ne bi prilikom testiranja, a i općenito, smjeli kršiti odrednice GDPR-a (engl. „*General Data Protection Regulation*“), dokumenta Europske unije koji propisuje niz pravila vezana za rukovanje podacima korisnika na internetu.[8]

Pod ostale izazove možemo nabrojati vremenski pritisak s kojim se testeri susreću; za testiranje je gotovo uvijek propisan strogi vremenski rok unutar kojeg se moraju kvalitetno i efikasno odraditi svi prethodno opisani koraci penetracijskog testiranja. Vrlo je bitno imati dobro razvijen plan kako bi se svi zadaci ispunili na vrijeme i kako se ne bi probili vremenski rokovi.

Komunikacija s klijentima je isto tako važan čimbenik. Testeri moraju cijelo vrijeme komunicirati s klijentom, jasno i jednoznačno dogovoriti kako će se testiranje provoditi, na vrijeme rješavati nejasnoće, te na razumljiv i jasan način klijentu predati rezultate testiranja, objasniti rezultate i ponuditi rješenja kako otkloniti ranjivosti ukoliko one postoje; sve na način kojeg će klijent lako razumjeti.

Testeri moraju odabrati ispravne alate za testiranje koji najbolje odgovaraju aplikaciji koju testiraju, te ih ispravno primijeniti prilikom testiranja. Naposlijetku, danas postoji veliki broj alata za penetracijsko testiranje, stoga je važno da testeri budu upoznati s najnovijim alatima i trendovima u industriji.

4. VRSTE PENETRACIJSKOG TESTIRANJA

Kako postoje različiti alati s kojima se može provoditi testiranje, tako postoji i više vrsta penetracijskog testiranja koje su testerima na raspolaganju. Razlikujemo tri vrste penetracijskog testiranja: „Black box“ testiranje, „White box“ testiranje i „Gray box“ testiranje. Svako od ovih testiranja ima posebne karakteristike koje ih čine različitima.

4.1. Black box testiranje

Black box testiranje je oblik testiranja koji podrazumijeva simuliranje napada u kojem vanjski napadač pokušava steći nedozvoljen pristup aplikaciji ili sustavu bez da ima ikakva prethodna saznanja o istima. Prilikom simulacije napada testeri prvo pretražuju javno dostupne informacije i koriste prethodno opisane alate za testiranje, tzv. skenere kako bi pronašli ranjivosti. Javno dostupne informacije se pronalaze na internetu, korištenjem standardnih Internet pretraživača poput tražilice Google.[9]

Nakon što se pretraže javno dostupne informacije, kreće se sa skeniranjem otvorenih portova i servisa na mreži. Alati koji se često koriste za ovu svrhu su Nmap i Masscan.[10] Portovi se skeniraju sa svrhom da testeri otkriju sve moguće pristupne točke u aplikaciju; otvoreni portovi mogu predstavljati prisutnost mrežnih usluga. Primjerice, otvoreni port može značiti da se njime služi baza podataka, FTP server ili da na njemu postoji web server. Nakon što su portovi skenirani, nastoji se precizno identificirati prethodno navedene usluge, drugim imenom servise, na otvorenim portovima na kojima mogu biti potencijalni ciljevi za daljnje ispitivanje.

Kada su usluge na otvorenim portovima identificirane, primjerice baza podataka, kreće se u detaljniju analizu web aplikacija. Za ovaj zadatak može se koristiti alat kao što je Burp Suite, koji je ujedno i najviše korišten i izuzetno moćan alat za sigurnosno testiranje aplikacija. Burp Suite je automatizirani alat, tzv. skener, koji omogućuje istraživanje i manipuliranje HTTP/HTTPS prometom kako bi se otkrile ali i iskoristile ranjivosti u sigurnosti.

Burp Suite pruža sljedeće usluge: presretanje i izmjena prometa između klijenta i servera, automatsko indeksiranje web stranica kako bi se stvorila mapa strukture web aplikacije, skeniranje ranjivosti u svrhu pronalaska uobičajenih propusta poput SQL injekcije, Cross-Site Scripting i ostalih, te za analizu i manipulaciju parametrima unosa kako bi se ispitao način na koji aplikacija obrađuje podatke koji nisu predviđeni.[11]

Black box metoda predstavlja najrealističniju imitaciju napada, nije najučinkovitija, ali je zato najsloženija za provesti kako za stvarne napadače, tako i za testere koji ih oponašaju jer nemaju nikakve direktne informacije o aplikaciji; unutarnja struktura koda i implementacijski detalji nisu poznati. Zbog toga je teško pokriti sve moguće dijelove koda, pogotovo ako postoji više grananja u nekom dijelu koda. Ukoliko je moguće, dobra je praksa testerima na kraju testiranja omogućiti uvid u izvorni kod kako bi mogli utvrditi dodatne ranjivosti ukoliko nisu otkrivene prilikom samog Black box testiranja.[9]

4.2. White box testiranje

Za razliku od Black box testiranja u kojem testeri nemaju nikakve izravne informacije o izvornom kodu i arhitekturi web aplikacije koju testiraju, kod White box testiranja to nije slučaj. Prilikom White box testiranja, testeri imaju potpuni pristup svim informacijama o sustavu što uključuje izvorni kod, arhitekturu aplikacije, mrežnu konfiguraciju i povezane vanjske komponente ukoliko one postoje.[12]

Velika prednost naspram Black box pristupa je ušteda vremena koje bi testeri inače morali utrošiti na prikupljanje podataka s interneta, otkrivanje portova i analizu servisa koji se nalaze na njima. Testeri od početka testiranja raspolažu sa svim informacijama vezanim za aplikaciju koje su im potrebne i na taj način mogu provesti testiranje koje će imati bolju pokrivenost cijele aplikacije. Testiranje se vrši na dostupnom kodu i testeri cijelo vrijeme imaju uvid u kod, stoga mogu puno lakše odrediti ranjivosti i eventualne ostale greške. Isto tako, lakše je razviti automatizirane testove i prilagoditi skenere koji su u potpunosti prilagođeni aplikaciji koja se testira.

Nažalost, iako ovaj pristup nudi dosta prednosti naspram ostalih pristupa testiranju, on isto ima svoje mane. Testeri koji provode testiranje na aplikaciji moraju imati duboko znanje o aplikaciji koju testiraju, ali isto tako će na raspolaganju imati veliki broj informacija o svim dijelovima aplikacije. Velika količina informacija može uzrokovati da testeri previde eventualne ranjivosti koje bi napadači možda mogli iskoristiti. White box testiranje je isto tako dosta skuplje u odnosu na druge vrste testiranja.

Ukoliko se testirana aplikacija još nalazi u fazi razvoja to može biti i dobro i loše. Dobro je što aplikacija još nije dostupna napadačima i stoga nema prijetnje, a ranjivosti se mogu otkloniti na vrijeme, tj. prije nego što bi ih napadači iskoristili.

Loše je što se aplikacija može promijeniti u svakom trenutku dok se razvija i zato je važno da testeri prate sve promjene i njihove posljedice koje bi mogle imati utjecaj na ranjivost cijele aplikacije. Radi svih ovih čimbenika, White box testiranje iziskuje veliku količinu vremena što isto može biti nedostatak.[12]

Proces White box testiranja sastoji se od više aktivnosti i uvjeta. Prvi uvjet je da tester razumije alate i tehnologije koje će koristiti prilikom testiranja. Nadalje, tester mora dobro definirati i odrediti područje i opseg testiranja aplikacije, te poznavati programske jezike, alate i tehnologije koji se koriste za razvoj aplikacije koja se testira.

Drugi korak je da se provjeri, odnosno analizira kod koji će se testirati, i da se u potpunosti razumije dio koda koji se testira.

Nakon toga se pišu testni slučajevi koji bi mogli otkriti eventualne ranjivosti u aplikaciji. Važno je da tester ima na umu koje bi se sve ranjivosti mogle pojaviti. Zatim se pokreću testni slučajevi koji su rađeni za otkrivanje tih ranjivosti. Pored korištenja korisnički definiranih testova, tester na raspolaganju ima alate SonarQube i Checkmarx koji su statički analizatori s kojima se mogu automatizirano provjeriti sigurnosni propusti unutar koda. U ovom koraku je cilj kao i u Black box testiranju otkriti uobičajene ranjivosti poput SQL injekcije, XSS-a itd.

Uz pogreške u kodu, važno je analizirati mrežne dijagrame, tj. vidjeti kako je cijeli sustav strukturiran, kako pojedine komponente aplikacije međusobno komuniciraju kako bi se izbjegle strukturne ranjivosti i sigurnosni propusti. U ovom koraku se isto provjerava kako su konfigurirani sigurnosni mehanizmi aplikacije poput VPN-a, vatrozida i ostalih mjera ukoliko su one prisutne.

Posljednji korak je samo penetracijsko testiranje, odnosno simulira se napad kao i kod Black box pristupa. Pokušava se probiti sigurnosne mehanizme aplikacije i ostvariti neovlašten pristup iako postoji duboko znanje o funkcioniranju same aplikacije. Isto tako se koristi tehnika zvana „fuzzing“ s kojom se aplikaciji šalje nepredviđene ili slučajne podatke da se vidi kako će ih obraditi i hoće li se na taj način isto eventualno otkriti neke ranjivosti.

Na kraju se radi analiza i bilježenje rezultata, te se oni detaljno proučavaju i na osnovu njih se daju adekvatna rješenja i preporuke koje bi otklonile ranjivosti.

4.3. Gray box testiranje

Kako se vjerojatno može i zaključiti iz samog imena, Gray box testiranje je vrsta testiranja koja je po svojim karakteristikama kombinacija Black box i White box testiranja. Alternativni naziv na koji se može naići u literaturi je prozirno testiranje, odnosno „Translucent testing“ na engleskom. Kod Gray box testiranja testeri imaju djelomične informacije i nepotpuno znanje o karakteristikama aplikacije koju testiraju, te nemaju neograničen pristup cijelom izvornom kodu. Pod karakteristike aplikacije ubrajamo informacije o infrastrukturi, izvornom kodu i arhitekturi same aplikacije. Zatim testeri koriste svoja vlastita saznanja o aplikaciji kako bi obavili što bolji posao prilikom traženja i izvještavanja o ranjivostima.[13]

Korištenjem ovog pristupa, testerima se omogućuje da iskoriste ograničene informacije koje imaju kako bi se testiranje bolje usmjerilo. Iako raspolažu s manje informacija nego kod White box testiranja gdje je kao jedan od problema navedeno da previše raspoloživih informacija o aplikaciji može testiranje odvesti u krivom smjeru, kod Gray box testiranja to nije slučaj. Naime, s obzirom da testeri imaju otprilike jednak broj informacija koji bi mogli imati i napadači to ih usmjerava da pokušavaju što više razmišljati kao stvarni napadači..

Gray box testiranje se provodi u pet faza koje imaju dosta sličnosti s Black box i White box testiranjem. Prva faza je planiranje i analiza zahtjeva u kojoj testeri moraju shvatiti opseg aplikacije; kakva je arhitektura aplikacije, koji programski jezici, knjižnice, frameworkovi i alati se koriste za razvoj aplikacije. Testeri s razvojnim programerima imaju intervjue, dobivaju uvid u tehničku dokumentaciju, zahtijevaju osnovne informacije vezane uz aplikaciju poput izmišljenih korisničkih računa i odgovarajućih prava pristupa za iste, ali i dalje nemaju uvid u kompletan izvorni kod.

Zatim slijedi analiza prikupljenih informacija i planiranje s ciljem da se razvije strategija koja će iskoristiti dostupne informacije kako bi se testiranje što bolje provelo i otkrilo čim više ranjivosti. Potrebno je identificirati kritične komponente unutar aplikacije koje će iziskivati detaljnije testiranje i razviti testne slučajeve koji će ciljati specifične ranjivosti na temelju informacija s kojima testeri raspolažu.

Nakon što se razvila strategija testiranja, kreće se sa skeniranjem aplikacije i otkrivanjem ranjivosti. Testeri nastoje identificirati sve ranjivosti aplikacije koristeći informacije koje su stekli u prethodnim fazama. Testiraju se poznate ulazne točke i API-jevi koji se koriste, te se provjeravaju sigurnosne postavke i konfiguracija aplikacije kako bi se otkrile eventualne ranjivosti.

Testeri mogu ručno testirati aplikaciju ali isto tako mogu koristiti alate koji se se već spominjali i koji su prethodno opisani: Nmap i Burp Suite. Nmap se koristi za skeniranje mreže, te identificiranje otvorenih portova i servisa koji se nalaze na njima. Burp Suite je automatiziran alat za penetracijsko testiranje i sam može pokrenuti veliki broj testova koji mogu ispitati ranjivosti ali isto tako Burp Suite može manipulirati HTTP i HTTPS prometom.

Jednom kad su potencijalne ranjivosti otkrivene, testeri rade na eksploataciji istih kako bi odredili kakav utjecaj one imaju na sigurnost aplikacije i hoće li doći do narušavanja sigurnosti. Testeri isto tako na uvid mogu dobiti sigurnosne logove koje ima aplikacija, kao i zabilježene reakcije sustava na pokušaje napada. U ovom koraku se često koriste alati kao Metasploit Framework u kojem već postoji otprilike devetsto načina na koje se može testirati neka ranjivost unutar sustava. Jedna od mnogih korisnih značajki je da tester isto tako može sam unutar programa Metasploit Framework ili Burp Suite definirati tzv. „payload code“ odnosno kod koji će se izvršiti ukoliko se uspije ostvariti neautorizirani pristup dijelovima aplikacije koji ne bi smjeli biti dostupni. Isto se može koristiti tehnika zvana „fuzzing“ koja je prethodno opisana i koja za cilj ima utvrditi ponašanje aplikacije kada se u nju pošalju podaci koji su nepredviđeni ili imaju slučajne vrijednosti.[14]

Kao i kod prethodno opisanih vrsti testiranja, posljednji korak je pisanje izvještaja i informiranje naručitelja testiranja o pronađenim ranjivostima, kako one utječu na rad aplikacije, te kako ih je moguće otkloniti ili ublažiti.

Kada se usporedi Gray box testiranje s White box i Black box testiranjima, vidi se da ono nudi najbolje od oba svijeta. Pruža učinkovitije i realističnije testiranje jer simulira napad s ograničenim internim znanjem kojeg bi napadači mogli steći ako dobivaju informacije unutar razvojnog tima ili ako uspiju osigurati pristup tim informacijama nekim drugim, nelegalnim putem. Isto tako, testeri su na neki način prisiljeni razmišljati kao napadači što može osigurati bolju pokrivenost kritičnih dijelova aplikacije. No uz prednosti koje nudi, postoje i nedostatci u vidu da testeri nemaju potpun spektar informacija i saznanja kao što bi imali kod White box testiranja i njihov rad uvelike ovisi o kvaliteti informacija koje dobiju od razvojnog tima ili koje sami otkriju u fazama prikupljanja informacija i otkrivanja ranjivosti koje će se eksploatirati.

5. PREGLED I USPOREDBA ALATA KOJI SE KORISTE ZA PENETRACIJSKO TESTIRANJE

U prethodnom poglavlju prilikom opisivanja različitih vrsta penetracijskog testiranja bili su spomenuti određeni alati koji su na raspolaganju testerima; Nmap (Network Mapper), Burp Suite i Metasploit Framework koje će se u ovom poglavlju detaljnije opisati. Uz navedene alate, opisivati će se još alati John the Ripper (JtR) i Wireshark koji je bio korišten u sklopu kolegija Računalne mreže. Svaki alat se koristi za specifičnu zadaću unutar cijelog postupka penetracijskog testiranja i skupa uvelike olakšavaju i ubrzavaju rad testera ako se pravilno koriste.

5.1. Nmap

Nmap, odnosno Network Mapper, je besplatni i open source alat koji se koristi za otkrivanje mreže i ispitivanje sigurnosti. Koriste ga brojni testeri, te sistem i mrežni administratori za ispunjavanje zadaća poput popisivanja mrežnih resursa (popisivanje mrežnih uređaja, softvera, veza i ostalih resursa unutar računalne mreže koji su pod njihovom ovlasti), upravljanje rasporedima nadogradnje usluga i nadzor dostupnosti hosta ili usluge.[15]

Koristi sirove IP pakete i iz njih otkriva koji su poslužitelji dostupni na mreži, koje usluge daju poslužitelji (ime aplikacije i njihova verzija), koja vrsta paket filtera i vatrozida se koristi, koji operacijski sustav koristi neki od poslužitelja itd.. Razvijen je s fokusom da se u vrlo kratkom vremenu skeniraju velike mreže ali isto dobro radi i za otkrivanje pojedinačnih poslužitelja. U zajednici internetske sigurnosti smatra se za jedan od najvažnijih alata radi svojih naprednih značajki i prilagodljivosti širokom spektru okolnosti.

S aspekta penetracijskog testiranja, najčešće se koristi prilikom faze otkrivanja kako bi testeri stekli razumijevanje mrežne strukture u kojoj se cilj nalazi. Nmap se koristi da se otkriju kritični sustavi unutar mreže i čije se ranjivosti mogu iskoristiti za napade.

Funkcionalnosti Nmap-a koje testeri koriste su:

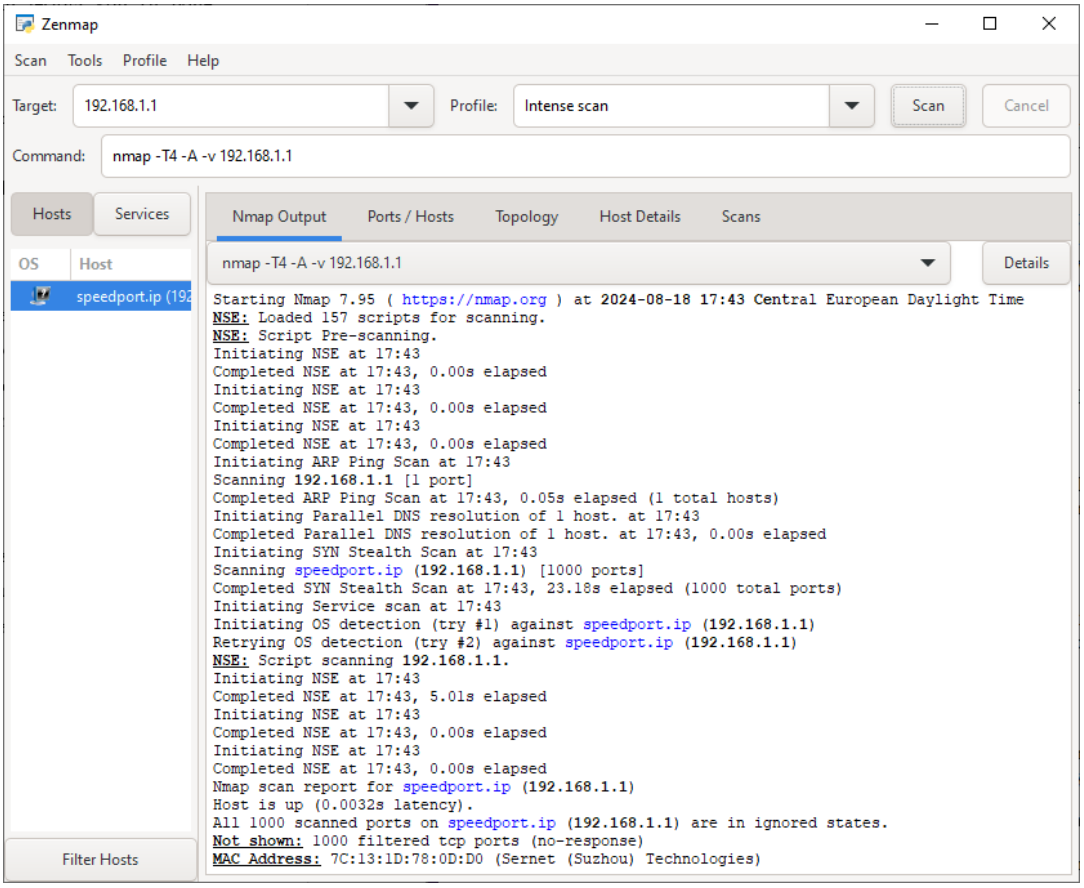
- Skeniranje portova s ciljem identifikacije otvorenih portova na ciljanom sustavu koristeći metode:
    - TCP SYN (Stealth) skeniranja (još poznato po nazivu „half-open“ skeniranje) u kojem se šalje SYN paket ciljanom portu i onda se analizira odgovor; SYN-ACK označava otvoreni port, dok RST označava zatvoren port. Ukoliko nema odgovora port je zatvoren ili koristi vatrozid. Preporučen i najćešće korišten način skeniranja.

- TCP connect je skeniranje u kojem se koristi puna TCP povezanost za otkrivanje otvorenih portova. Port se skenira tako da se uspostavi potpuna povezanost s poslužiteljem (SYN, SYN-ACK, ACK); jednom kada se port otvori napadači imaju uspostavljenu vezu sa servisom koji može imati neke od poznatih ranjivosti poput SQL injekcije, XSS-a itd. koje mogu iskoristiti. Veza se zatvara slanjem FIN paketa. Ovo skeniranje je ujedno i najrizičnije za stvarne napadače jer se s uspostavljanjem potpune TCP povezanosti može lakše otkriti i zabilježiti izvor napada.[16]
- UDP skeniranje koristi se za skeniranje usluga koje koriste UDP protokoli kao što su DHCP (port 53), DNS (portovi 161/162) i SNMP (portovi 67/68). Ovaj tip skeniranja je sporiji i kompliciraniji u odnosu na TCP skeniranja i stoga ga neki mrežni administratori i testeri ignoriraju, ali kao tester ga je bitno izvršiti jer napadači iskorištavaju sve mogućnosti. Funkcionira tako da se šalje UDP paket na svaki ciljani port. Paketi koje šaljemo na većinu ciljanih portova će biti prazni, a za neke portove će se morati koristiti posebni paketi specifični za protokol na tom portu.[16]
- Navedene metode skeniranja koriste poznate TCP i UDP protokole s kojima se možemo susresti u gotovo svim mrežama na svijetu i iskorištavaju njihove ranjivosti. Uz navedene metode skeniranja postoji još metoda koje su prilagođene za razne okolnosti testiranja i mogu se pronaći u Nmap dokumentaciji.
- Ovo je ujedno i najbitnija Nmap značajka koja se koristi.

- Host discovery:
  - Oktrivanje aktivnih uređaja na mreži koristeći tehnike kao što su ping skeniranje (ICMP echo), TCP ping, ARP skeniranje na lokalnim mrežama itd. Nmap nudi različite mogućnosti skeniranja hostova koje su krojene prema zahtjevima administratora, testera i običnih korisnika. Primjerice, sistem administratoru će biti dovoljno samo napraviti ICMP ping da pronađe hostove na svojoj internoj mreži, dok će penetracijski tester možda morati koristiti sve gore navedene tehnike kako bi izbjegao vatrozid.[17]
- OS Detection:
  - Analiziranje odgovora na pakete koje smo slali s ciljem da bi se identificirao operativni sustav koji se koristi i njegova verzija. To se radi kako bi se steklo bolje razumijevanje ciljanog sustava i kako bi se napravili testovi koji su bolje prilagođeni.[18]

- Scripting engine (NSE):
    - Jedan od najjačih i fleksibilnih značajki Nmap-a, NSE omogućuje korisnicima da sami naprave vlastite skripte ili koriste postojeće kako bi automatizirali složene zadatke poput skeniranja ranjivosti, izvođenje raznih napada ili detektirali zlonamjeran softver.[19]
    - Pisanje skripti se radi u Lua programskom jeziku.

Nmap je fleksibilan alat koji ima veliki broj opcija i prilagođavanja različitim potrebama, kompatibilan je sa svim poznatijim operacijskim sustavima (Linux, Windows, macOS) i ima aktivnu zajednicu koja konstantno radi na unaprjeđenju alata, te pregršt dokumentacije.



Slika 5.1. Grafičko korisničko sučelje (GUI) programa Nmap, tzv. "Zenmap"

No kako bi ga mogli učinkovito koristiti, potrebna je velika količina znanja o web sigurnosti i računalnim mrežama. Potrebno je poznavanje protokola, mrežne strukture, slojeva i metoda skeniranja koje se koriste. Isto tako, Nmap može biti detektiran i žrtva napada ili cilj testiranja može onemogućiti daljnji pristup.

5.2. Burp Suite

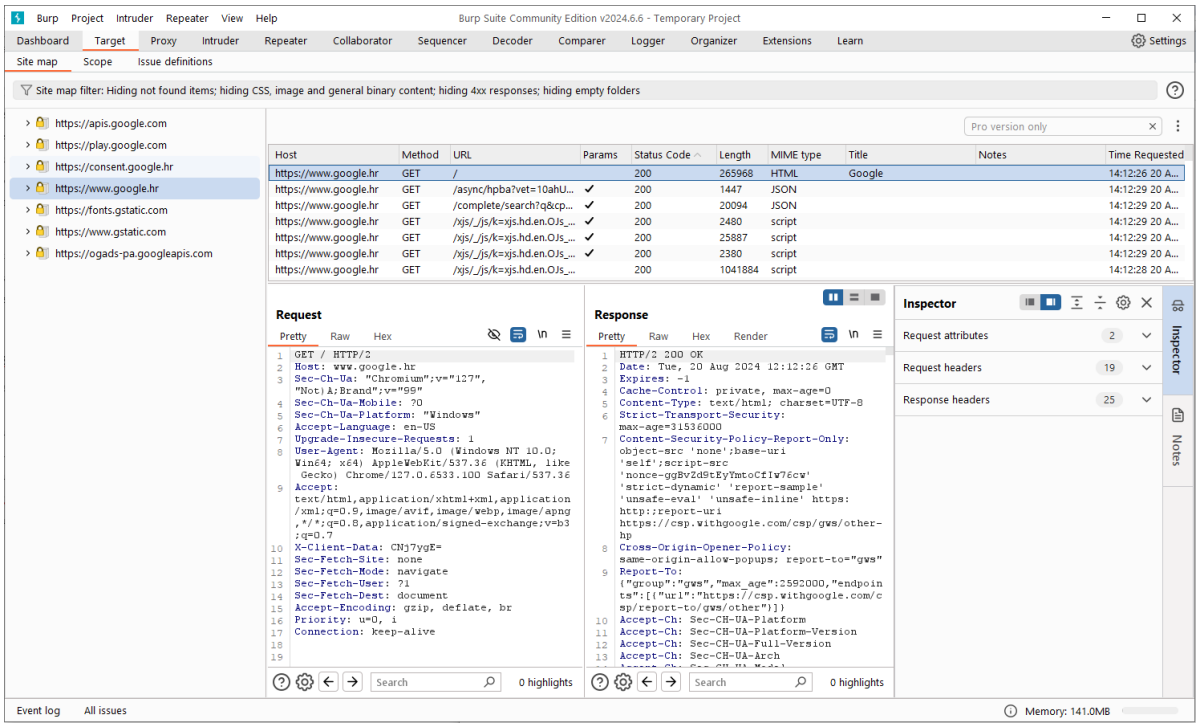
Sljedeći alat u arsenalu testera je Burp Suite, alat koji nudi brojne automatizirane i ručne metode testiranja, te služi za testiranje sigurnosti web aplikacija. Za razliku od Nmap-a koji se više koristi za početno mrežno skeniranje i identifikaciju otvorenih portova i servisa, Burp Suite se koristi za dubinsku analizu i testiranje sigurnosti web aplikacija, te bi se ova dva alata trebala koristiti komplementarno. Testeri bi trebali Burp Suite koristiti za identificiranje i iskorištavanje ranjivosti poput SQL injekcije, procjenu sigurnosti aplikacije itd.

Uvelike se koristi u području web sigurnosti, a razvila ga je kompanija PortSwigger i trenutno postoje tri inačice: Community, Professional i Enterprise. Community je besplatna inačica koja nema sve mogućnosti koje imaju inačice Professional i Enterprise koje je potrebno kupiti. Za potrebe ovog rada koristiti će se Community inačica.[20]

Funkcionalnosti programa Burp Suite:

- Proxy za presretanje
    - Glavna funkcionalnost Burp Suite-a; omogućuje presretanje, pregledavanje i izmjenu HTTP/HTTPS zahtjeva i odgovora između web preglednika i aplikacije koju se promatra
    - Koristi se kako bi se iz podataka koje aplikacija šalje i prima moglo izvući informacije
- Spidering
    - Automatski se indeksiraju web aplikacije kako bi se otkrile sve dostupne stranice i funkcionalnosti, čak ako i aplikacija koja se promatra koristi materijale s drugih stranica, te druge stranice će isto biti mapirane.
    - Služi za mapiranje aplikacije prije nego li se izvrši testiranje
- Intruder
    - Alat s kojim je moguće pakete koji su se presreli pomoću proxya kasnije iskoristiti za napad na način da se u njih doda tzv. payload, odnosno da se izmjeni postojeći paket i u njega dodaju kod ili podaci koji bi mogli iskoristiti ranjivosti u web aplikaciji
- Vulnerability scanner
    - Automatizirani skener koji prolazi kroz web aplikaciju i pokušava otkriti postoje li uobičajene ranjivosti koje bi se mogle iskoristiti poput SQL injekcija, XSS-a, itd.

- Repeater
    - Omogućuje da testeri sami šalju izmijenjene zahtjeve web aplikaciji i na taj način mogu analizirati kako aplikacija odgovara na različite ulazne vrijednostil
- Ekstenzije
    - Burp Suite podržava dodatke koji omogućuju da se u sam alat doda još funkcionalnosti, omogućuje integraciju s drugim alatima i dodavanje specifičnih module za napredne testove



Slika 5.2. Grafičko korisničko sučelje programa Burp Suite

Burp Suite je odličan i svestran alat koji nudi pregršt funkcionalnosti za testiranje web aplikacije unutar jednog programa. Omogućava kvalitetno testiranje sigurnosti web aplikacija sa svojim automatiziranim testovima, a testeri ga mogu dodatno prilagođavati svojim potrebama i proširivati njegove funkcionalnosti.

Iako Community inačica nudi pregršt korisnih funkcionalnosti, ne pruža sve mogućnosti Burp Suite-a koje bi se dobile u plaćenim verzijama ovog programa; npr. u Community verziji nije moguće spremati svoj rad i rezultate testiranja kako bi se lako nastavilo testiranje kada se opet upali program. Iako sam programa ima veoma intuitivno i jednostavno sučelje, potrebno je educirati se kako bi se mogao koristiti i imati određeno iskustvo u polju web sigurnosti.

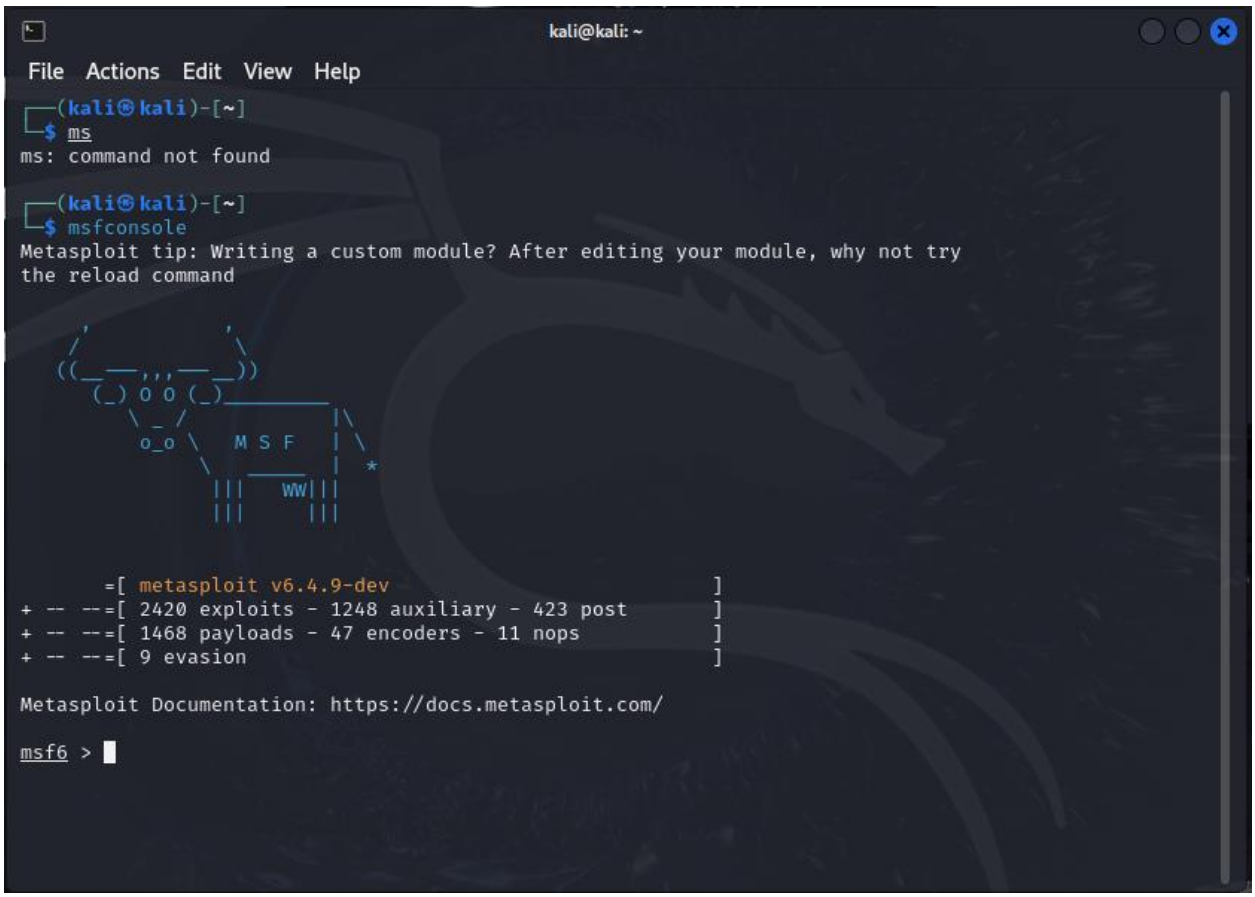
5.3. Metasploit Framework

Metasploit Framework je open source alat, točnije framework koji se koristi za penetracijsko testiranje. Omogućuje testerima da prilagođavaju napade, razvijaju i izvršavaju eksploatacije koje ciljaju ranjivosti neke web aplikacije.[21] Može se koristiti u svim fazama penetracijskog testiranja; pri skeniranju i u fazi prikupljanja informacija, u fazi eksploatacije u kojoj se iskorištavaju ranjivosti za izvođenje zlonamjernih skripti pa čak i nakon što se faza eksploatacije izvrši, Metasploit omogućuje testerima, ali i stvarnim napadačima da upravljaju sustavom s daljine.

Glavne funkcionalnosti programa Metasploit:

- Eksploatacija ranjivosti
    - Primarna funkcionalnost programa, sa samim programom već dolazi velik broj postojećih skripti za iskorištavanje raznih poznatih ranjivosti, isto tako korisnici mogu kreirati svoje skripte ili izmijeniti postojeće.
- Payload
    - Nakon što se ranjivosti iskoriste i testeri, odnosno napadači, dobiju pristup web aplikaciji, mogu iskoristiti tzv. „Payload“ opciju.
    - Payload je zlonamjerni kod koji se može ubaciti u web aplikaciju s ciljem da se npr. poremeti rad web aplikacije ili preuzme kontrola nad web aplikacijom.
    - Postoje takvi payloadi koji omogućavaju napadačima da sa svoga računala izvršavaju naredbe na kompromitiranom računalu žrtve.
- Encoding
    - Koristi se kako bi testeri sakrili ili kodirali payload s ciljem da se izbjegne detekcija zlonamjernog koda.
- Moduli za korištenje nakon stjecanja pristupa
    - Unutar programa postoje brojni gotovi alati koji se mogu koristiti za istraživanje web aplikacije; njene strukture i koda, ali i za prikupljanje informacija i ubacivanje prethodno opisanog payloada.
- Modularnost
    - Metasploit je modularan framework koji omogućava korisnicima da ga dodatno prilagode svojim potrebama, tj. da u njega dodavaju module koji su im potrebni.
    - Može se dodati nove module, skripte za iskorištavanje ranjivosti, „payload-e“, pa čak integrirati i ostale alate poput Nmapa u sam framework što uvelike može ubrzati rad testera jer ima sve potrebne alate na jednom mjestu.[22]

Prednosti koji Metasploit nudi su različite; sveobuhvatan je alat koji nudi sve potrebne funkcionalnosti za cijeli proces penetracijskog testiranja. Može se lako prilagoditi potrebama testera, a zahvaljujući svojoj modularnosti u njega je moguće integrirati druge alate, razviti vlastite module i skripte i upravo zbog toga je nezaobilazan alat prilikom penetracijskog testiranja. Uz navedene prednosti, ima široku zajednicu koja ga koristi, bogatu dokumentaciju koja nudi odgovore na sva pitanja ali i redovita ažuriranja.



Slika 5.3. Tekstualno korisničko sučelje programa Metasploit

Međutim, Metasploit može biti izazovan novim testerima radi velikog spektra mogućnosti koje nudi, te kao i prethodno opisani alati, zahtjeva određenu dozu predznanja kako bi se mogao koristiti; potrebno je razumjeti koncepte web sigurnosti i određene tehničke detalje aplikacija koje testiramo. Koristi tekstualno korisničko sučelje. Nažalost, kao i brojni alati za penetracijsko testiranje, može se koristiti i u ilegalne svrhe kao alat napadača.

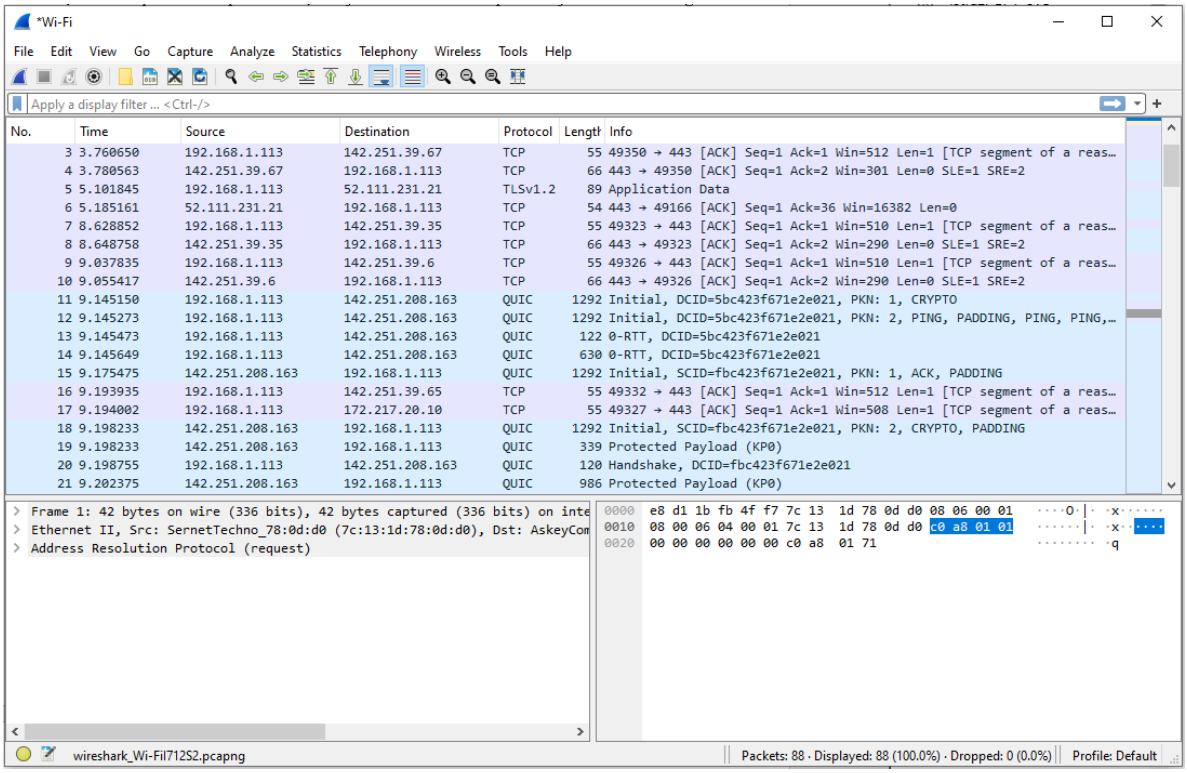
5.4. Wireshark

Wireshark je najpoznatiji alat za analizu mrežnog prometa, točnije za presretanje i analizu paketa u stvarnom vremenu koji se kreću nekom vezom.[23] Prilikom penetracijskog testiranja može dati uvid u potencijalne ranjivosti ovisno o protokolu koji koriste paketi i daje dublje razumijevanje o mrežnim protokolima koje koristi web aplikacija. Pomoću Wireshark-a može se identificirati sigurnosne prijetnje poput neovlaštene aktivnosti na mreži ili pokušaje napada. Isto tako, može se koristi za analizu zlonamjernih paketa koji žele iskoristiti određenu ranjivost u sustavu.

Osim za svrhe penetracijskog testiranja, koristi se za otklanjanje poteškoća u nekoj mreži, analizu, razvoj softverskih i komunikacijskih protokola, te u edukacijske svrhe. Korišten je i u sklopu kolegija „Računalne mreže“. Besplatan je i otvorenog koda, te se konstanto unaprjeđuje i razvija zahvaljujući velikoj bazi njegovih korisnika. Postoji ogroman broj literature i edukacijskih videa kako ga koristiti.

Glavne funkcionalnosti programa Wireshark:

- Presretanje prometa
    - Wireshark u stvarnom vremenu presreće pakete koji se kreću nekom vezom, te prikazuje pakete u čitljivom obliku.
- Podrška za velik broj protokola koji se koriste
    - Podržava velik broj mrežnih protokola, uključujući TCP/IP, HTTP, DNS, FTP i brojne druge.
- Dekodiranje protokola
    - Može dekodirati i prikazati sadržaj paketa što testerima omogućuje detaljnu analizu svake komponente mrežnog prometa vezne za testiranu web aplikaciju
- Filtriranje paketa
    - Filteri unutar programa omogućavaju korisniku da prate samo promet za koji smatraju da im je bitan.
    - Paketi se mogu filtrirati po protokolu koji paketi koriste, komunikaciji između specifičnih IP adresa ili portova itd.
- Jednostavno korisničko sučelje
    - Wireshark ima grafičko sučelje koje je jednostavno za razumjeti i koristiti, te se u kratkom vremenu može svladati korištenje ovog korisnog programa



Slika 5.4. Grafičko korisničko sučelje programa Wireshark

Prednosti Wiresharka su da omogućuje detaljnu analizu mrežnog prometa na vezi koja se promatra i daje uvid u rezultate na jednostavan i razumljiv način. Besplatan je za korištenje i podržan je na svim operacijskim sustavima (Linux, Windows, macOS).

Ukoliko se prati i presreće promet na vezi koja ima velik protok paketa, količina informacija koja će se prikazati na sučelju može biti ogromna i potrebno je dobro namjestiti filtere kako bi izolirali podatke koji su bitni, a za odabrati pravilne filtere potrebno je imati određeno znanje o web sigurnosti i internetskim protokolima.

Nadalje, s aspekta penetracijskog testiranja, Wireshark je alat koji služi samo za analizu i ne posjeduje nikakve alate za sprječavanje prijetnji ili iskorištavanje ranjivosti, već ih može samo identificirati.

5.5. John the Ripper

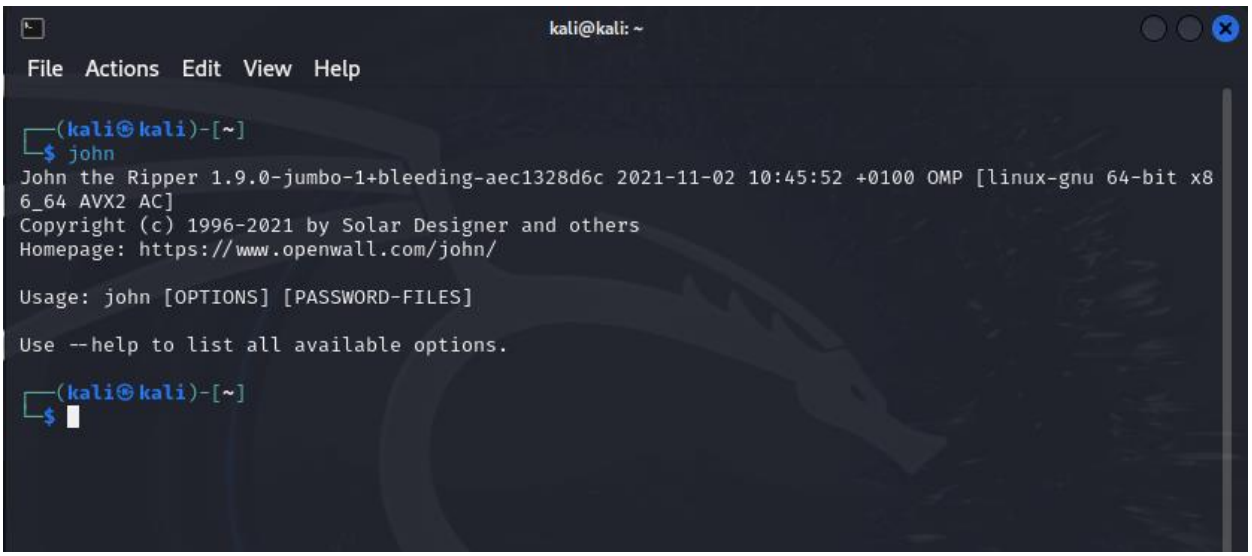
Posljednji alat koji će se opisivati je John the Ripper, još poznat pod akronimom JtR. Razvila ga je kompanija Openwall. Ovaj alat je razvijen za probijanje lozinki. JtR je snažan alat koji podržava razne hash algoritme i sposoban je provesti „brute force“ napade, napade rječnika i ostale sofisticiranije napade.[24]

Prvotna namjena ovog programa je testiranje sigurnosti, odnosno snage lozinki u aplikacijama i operacijskim sustavima, ali i za oporavak lozinki. Koriste ga sigurnosni stručnjaci i testeri s ciljem identificiranja slabih lozinki koje se koriste unutar sustava a koje bi napadači mogli lako probiti. Ovaj postupak je važan pogotovo za testiranje lozinki korisnika koja imaju viša prava – administratore.[25]

Glavne funkcionalnosti programa JtR:

- Podržava više formata:
    - Podržava širok raspon šifriranih lozinki i formata; Windows (LM, NTLM), Unix (DES, MD5) i brojne druge.
- Posjeduje različite metode napada:
    - „Dictionary attack“ – metoda napada na lozinke gdje se koristi popis često korištenih lozinki kako bi se pronašla ispravna lozinka
    - „Brute force„ napad – metoda napada gdje se sustavno isprobavaju sve moguće kombinacije lozinki dok se ne pronađe ispravna, zahtijeva dosta vremena.
    - Napadi temeljeni na unaprijed pripremljenim tzv. „Rainbow tablicama“; unaprijed izračunate tablice koje sadrže hash vrijednosti i odgovarajuće lozinke, čime se ubrzava proces pronalaženja lozinke bez potrebe za kontinuiranim izračunavanjem hash vrijednosti tijekom napada.
- Korisnički popisi lozinki
    - Mogu se koristiti korisnički definirani popisi lozinki za probijanje lozinki, te se mogu koristiti dodatne permutacije kako bi se povećala šansa za probijanjem
- Paralelizacija
    - Koristi se više dretvi procesora i distribucija posla preko više procesorskih jedinica (CPU) s čime se može znatno ubrzati proces probijanja lozinke

Brz je i učinkovit, pogotovo kada se koriste optimizirani korisnički popisi lozinki. Otvorenog koda je i prilagodljiv specifičnim potrebama korisnika, te ima veliki broj podržanih formata lozinki i metoda napada koje može koristiti.[25]



Slika 5.5. Tekstualno korisnčiko sučelje programa John the Ripper

Za određene metode napada poput „brute force“ i „dictionary attack“, potrebno je puno računalnih resursa, pogotovo za kompliciranе lozinke. Ilegalna upotreba ovog alata van svrhe penetracijskog testiranja i bez odobrenja može imati legalne posljedice.

5.6. Usporedba alata i njihove uloge

Svaki od navedenih i opisanih alata ima važnu ulogu u cjelokupnom procesu penetracijskog testiranja, te se navedeni alati odlično nadopunjuju ukoliko se pravilno koriste. Testeri i stručnjaci za sigurnost trebali bi znati koristiti sve navedene alate i vješto se služiti s njima kako bi mogli lakše otkriti i neutralizirati sigurnosne prijetnje i ostale ranjivosti unutar sustava kojeg testiraju.

Nmap je alat koji je idealno koristiti u početnim fazama testiranja kako bi stekli bolji uvid mrežne strukture aplikacije i kada se mora mapirati mrežno okruženje web aplikacije, te odrediti potencijalne ranjivosti koje će se kasnije iskorištavati.

Nakon što je mrežno okruženje web aplikacije mapirano, može se koristiti alat poput Burp Suite-a ili Metasploit Framework-a, ovisno o potrebama. Svaki od alata ima svoje prednosti, ali i nedostatke te je potrebno odlučiti koji će testeri koristiti ovisno o potrebama testiranja. Burp Suite je više orijentiran ka samim web aplikacijama dok je Metasploit Framework „općenitiji“ alat za penetracijsko testiranje. Također, Burp Suite koristi grafičko korisničko sučelje dok se za Metasploit Framework većina posla obavlja preko naredbenog retka, a postoji i grafičko korisničko sučelje.

Burp Suite se koristi za dubinsku analizu web aplikacije i njeno testiranje. Burp Suite omogućuje identificiranje i iskorištavanje ranjivosti unutar web aplikacije. Primarno je orijentiran na testiranje sigurnosti samih web aplikacija a ne na iskorištavanju ranjivosti i u okruženju aplikacije. Sadrži brojne korisne ugrađene alate koje je moguće proširivati s korisničkim dodacima ukoliko su potrebne dodatne mogućnosti. Posjeduje alate za skeniranje ranjivosti, te alate za „brute force“ napade i prethodno opisani tzv. „fuzzing“.

Metasploit Framework je prilagodljiva platforma za penetracijsko testiranje unutar koje možemo integrirati dodatne alate poput Nmapa i besplatan je za razliku od Burp Suite-a. Služi za iskorištavanje ranjivosti koje su otkrivene na svim razinama: mrežnoj razini, sistemskoj i aplikacijskoj razini. Omogućuje ubacivanje tzv. payloada u aplikaciju; programskog koda koji može poremetiti rad aplikacije, ili čak preuzeti kontrolu nad njom i omogućiti testeru ili napadaču da upravlja njome s udaljenosti.

Sadrži veliku postojeću bazu skripti i payloada koje se mogu upotrijebiti za iskorištavanje ranjivosti, te posjeduje alate za skeniranje i otkrivanje mreža, automatski generator alata za phishing i ostale napade.

Wireshark je „pasivni“ alat koji se koristi za duboku analizu mrežnog prometa i može se koristiti za svrhu identificiranja neobičnih aktivnosti ili potencijalnih napada na web aplikaciju ili mrežu u kojoj se nalazi. Iako se s njim ne može nikako direktno utjecati na web aplikaciju, može se koristiti za stjecanje dodatnog uvida u način kako web aplikacija funkcionira, koje protokole koristi, te na taj način steći dodatne informacije koje će pomoći u samom procesu penetracijskog testiranja. Ukoliko aplikacija ne koristi dobru enkripciju podataka unutar paketa, moguće je čak i otkriti povjerljive podatke.

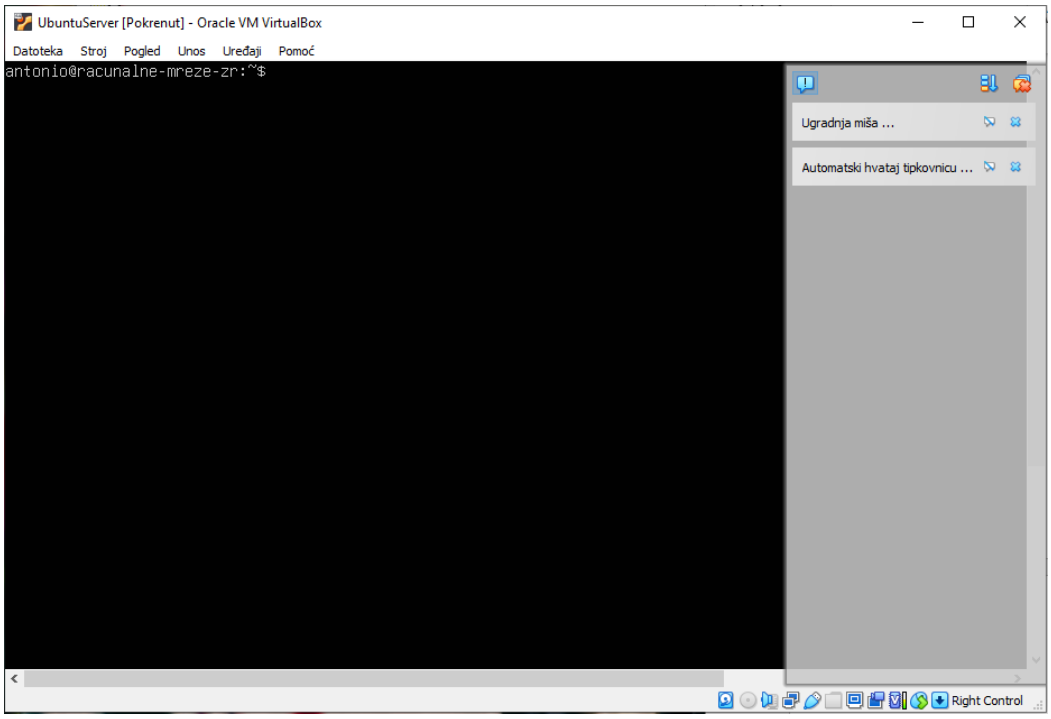
Posljednji opisivani alat bio je John the Ripper, alat koji se koristi za testiranje snage i probijanje lozinki. Od neophodnog je značaja kako bi se procijenilo koliko su lozinke unutar web aplikacije otporne na probijanje. Važno je ispitivati sigurnost lozinke pogotovo za one korisnike koji imaju višu razinu prava unutar aplikacije i pristup osjetljivim podacima s čijim bi probijanjem lozinke napadači mogli nanijeti bilo kakav oblik štete, materijalne ili ostalih.

6. PRIMJER PENETRACIJSKOG TESTIRANJA

U ovom poglavlju opisivati će se primjer penetracijskog testiranja na aplikaciji DVWA (Damn Vulnerable Web Application) koja služi za učenje i testiranje web sigurnosti. Razvio ju je Ryan Dewhurst koji na GitHub-u koristi korisničko ime digininja. Njena svrha je da bude pomoć sigurnosnim stručnjacima u smislu da isprobaju i unaprjeđuju svoje vještine u sigurnom i legalnom okruženju, te da bolje shvate procese osiguravanja web aplikacija od napadača. DVWA je PHP/MySQL web aplikacija koja ima velik broj ranjivosti, te posjeduje mogućnost konfiguracije stupnja ranjivosti same aplikacije. Za potrebu ovog rada koristiti će se stupanj „low“, odnosno najniži stupanj sigurnosti.[26]

6.1. Postavljanje virtualnih okruženja i instalacija potrebnih programa

Prvi korak je postaviti virtualno okruženje u kojem će se provoditi samo testiranje. Za potrebe testiranja koristiti će se program VirtualBox tvrtke Oracle kako bi se pokretale dvije virtualne mašine. Virtualna mašina s koje će se testirati ranjivu aplikaciju imati će na sebi Kali Linux koji dolazi s unaprijed instaliranim alatima za penetracijsko testiranje. Druga virtualna mašina, odnosno ciljna mašina će na sebi imati Ubuntu Server na kojem će se nalaziti ranjiva aplikacija DVWA.



Slika 6.1. Korisničko sučelje za Ubuntu Server

Nakon što su navedeni operacijski sustavi instalirani na virtualne mašine, može se krenuti s postupkom namještanja okoline za testiranje na istima.

Prvi korak je na virtualnu mašinu koja na sebi ima Ubuntu Server (dalje u tekstu: server) instalirati programe Apache, MySQL, PHP i na kraju DVWA aplikaciju. Apache će se koristiti za hosting stranice, MySQL za rad s bazom podataka, te PHP za pokretanje PHP dijela DVWA aplikacije. Navedene programe može se jednostavno instalirati korištenjem sljedeće naredbe (Napomena: sve naredbe u terminalu se unose bez navodnika " "):

„sudo apt install apache2 mysql-server php libapache2-mod-php php-mysql -y"

Nakon uspješne instalacije navedenih alata, može se instalirati DVWA koristeći sljedeće naredbe:

„cd /var/www/html"

„sudo git clone https://github.com/digininja/DVWA.git"

„sudo chown -R www-dana:www-dana /var/www/html/DVWA"

Pomoću gore navedenih naredbi preuzima se DVWA aplikacija korisnika digininja s repozitorija na GitHub-u.

Zatim je potrebno kreirati MySQL bazu podataka koju će koristiti DVWA aplikacija. Prvo je potrebno otvoriti sučelje za rad s MySQL. To se čini koristeći sljedeću naredbu s kojom se otvara sučelje kao root korisnik:

„sudo mysql -u root -p"

Nakon što se dobio pristup sučelju, prvo je potrebno stvoriti novu bazu podatka pod imenom „dvwa" u kojoj će se nalaziti podaci i korisnici. Naredbe koje se moraju unijeti su sljedeće:

1. „CREATE DATABASE dvwa;"
2. „CREATE USER 'dvwauser'@'localhost' IDENTIFIED BY 'password';"
3. „GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwauser'@'localhost';"
4. „FLUSH PRIVILEGES;"
5. „EXIT;

Izvršavanjem ovih naredbi:

1. Stvoriti će se baza podataka „dvwa",
2. Kreirati će se novi korisnik u MySQL sistemu s imenom „dvwauser„ koji će se moći prijavljivati u aplikaciju samo s mašine na kojoj se nalazi i sama aplikacija.
3. Zatim se korisniku „dvwauser" daju sve ovlasti nad bazom podataka, tj. da može izvršavati sve CRUD operacije.

4. Naredbom FLUSH PRIVILEGES osvježava se interni sustav privilegija za MySQL s čime prethodno dodijeljena prava odmah stupaju na snagu.
5. Izlazi se iz sučelja za MySQL

Sljedeći korak je konfigurirati DVWA sa sljedećim naredbama:

1. „cd /var/www/html/DVWA/config“
2. „sudo cp config.inc.php.dist config.inc.php“
3. „sudo nano config.inc.php“

Ovim naredbama se prvo postavljamo u direktoriji aplikacije DVWA koji sadrži konfiguracijske datoteke, zatim se datoteku „config.inc.php.dist“ koja je šablona za konfiguriranje prekopira u novu datoteku „config.inc.php“ koju će DVWA stvarno koristiti. Potom se otvori novu datoteku „config.inc.php“ u nano editoru i promijeni potrebne vrijednosti sukladno uputama koje se mogu pronaći na DVWA GitHub repozitoriju.

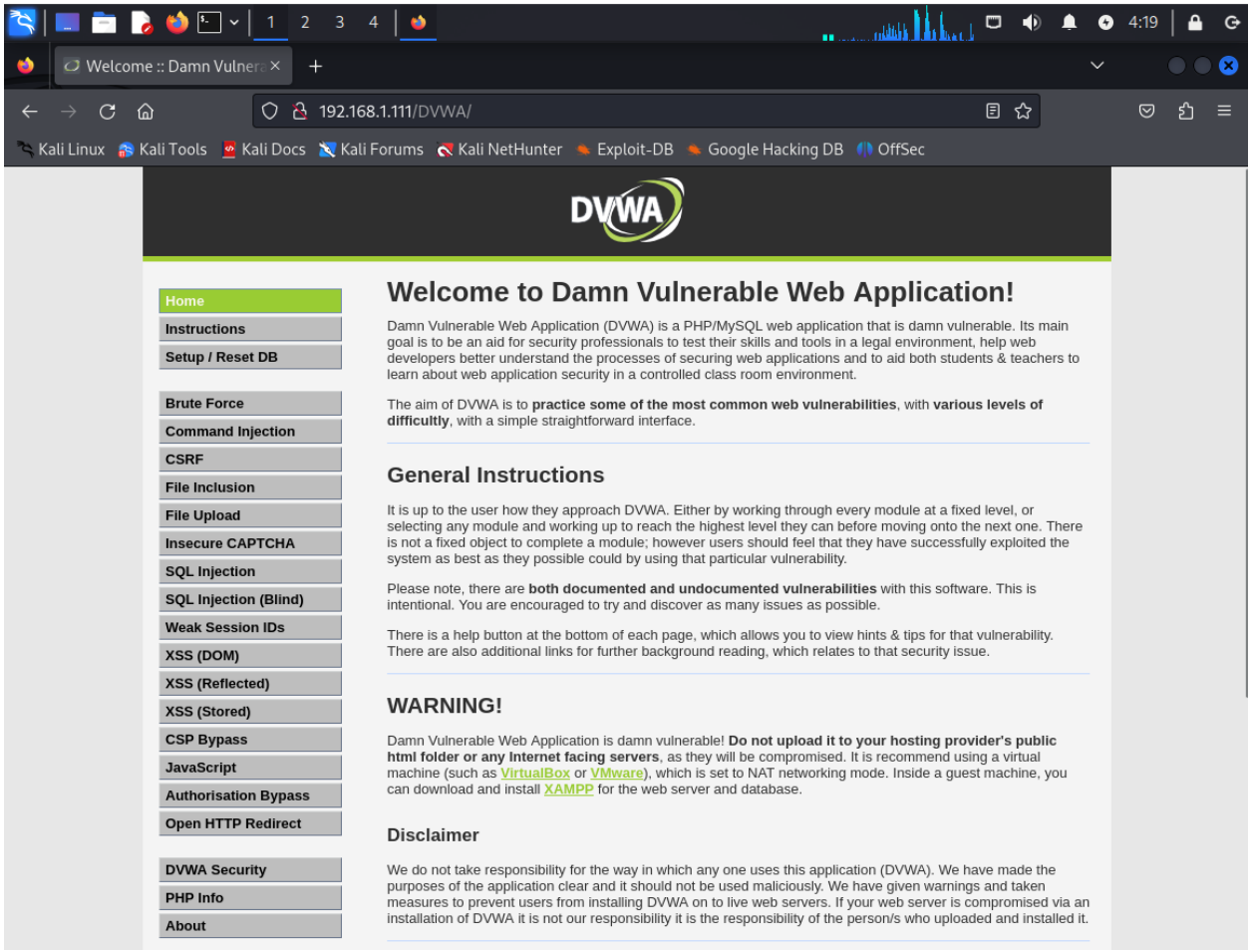
Nakon ovih naredbi potrebno je izvršiti sljedeće naredbe kako bi se pokrenuli programi Apache i MySQL:

1. „sudo systemctl start apache2“
2. „sudo systemctl start mysql“

Važna napomena je da se prilikom postavljanja virtualnih mašini obrati pozornost na mrežne postavke virtualnih mašina. Prilikom postavljanja virtualnih mašini, zadana postavka za mrežne adaptere je bila NAT (Network Adress Translation) što je izazivalo poteškoće u radu, te su se obje mašine morale postaviti na opciju „Premošćen Adapter“ kako bi virtualne mašine ispravno komunicirale.

Nakon izvršenja svih prethodno navedenih naredbi i ukoliko je sve ispravno postavljeno, može se pristupiti DVWA aplikaciji s druge virtualne mašine koja pokreće Kali Linux. To se čini tako da se u pregledniku na Kali OS-u unese IP adresu na kojoj se nalazi ranjiva web aplikacija koja se hosta na drugoj virtualnoj mašini. U ovom slučaju je to adresa 192.168.56.102/DVWA/ ali može varirati ovisno o tome kako je virtualna mreža konfigurirana. Na toj adresi nalazi se obrazac za ulogirati se u samu aplikaciju.

U aplikaciju se ulogirava koristeći korisničko ime „admin“ i lozinku „password“. Navedeni korisnički podaci su namjerno ovakvi kako bi se kasnije koristili za demonstriranje kako odabir loše lozinke i korisničkog imena može biti ranjivost.



Slika 6.2. Korisničko sučelje aplikacije DVWA na Kali linuxu

Nakon što se ulogiramo u aplikaciju, vidjeti će se web stranica koja na sebi sadrži uputstva i pravne odredbe, a u izborniku s lijeve strane se nalaze linkovi na stranice koje sadrže specifične ranjivosti koje se mogu iskorištavati u svrhu penetracijskog testiranja.

6.2. Postupak testiranja

Nakon što je sve postavljeno, može se početi sa samim postupkom penetracijskog testiranja. Testiranje će se provesti koristeći alate koji su prethodno opisani u petom poglavlju, a vrsta testiranja koja će biti provedena je Gray box. Vrsta testiranja je Gray box zbog toga što autor ovog rada nije autor aplikacije koju testira i nema saznanja o cjelokupnom kodu, ali i zbog toga što se prilikom same konfiguracije aplikacije DVWA i servera steklo znanje o aplikaciji koje se može iskoristiti za samo testiranje što bi u stvarnim okolnostima odgovaralo stanju da napadači ili testeri imaju dostupno interno znanje o aplikaciji. Odnosno količina informacija s kojima se raspolaže prilikom testiranja odgovara uvjetima prilikom Gray box testiranja.

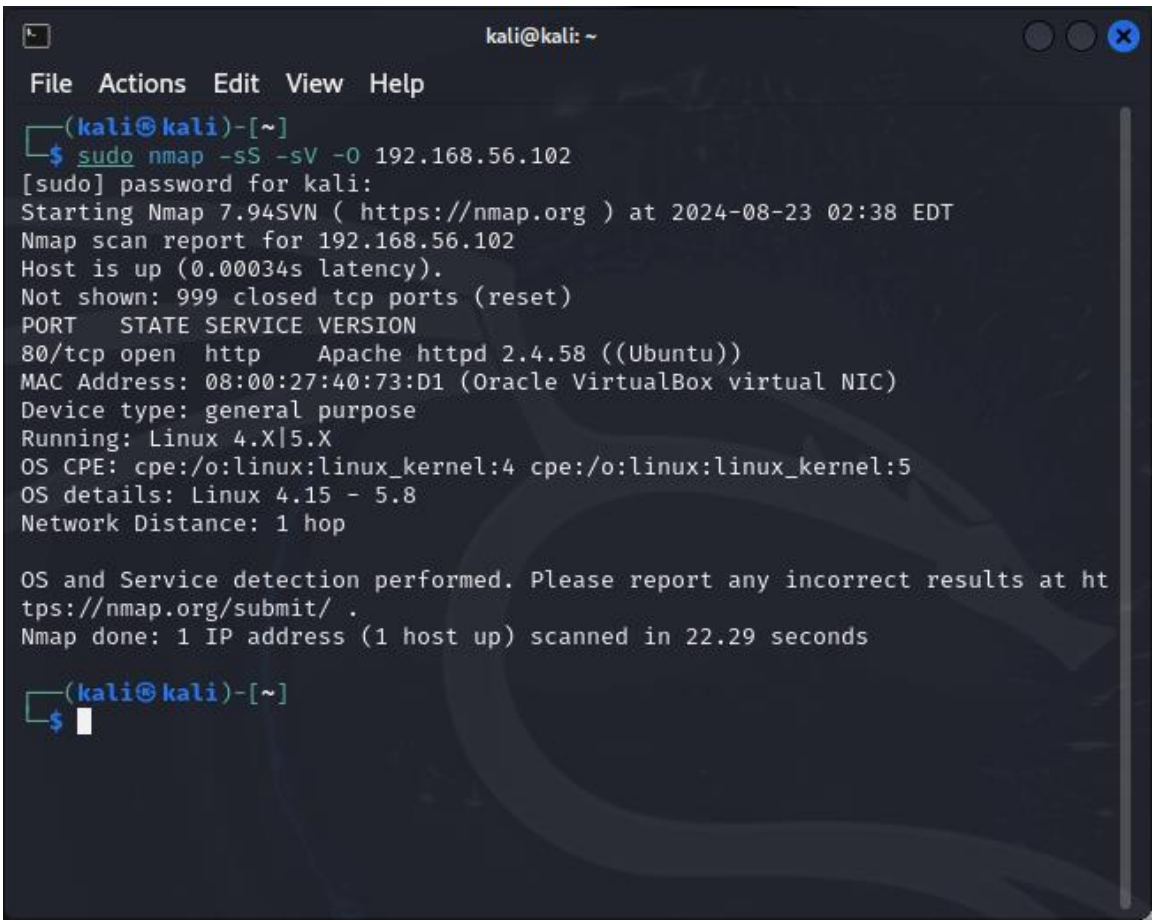
Prednost korištenja Kali OS-a je što dolazi s brojnim alatima za penetracijsko testiranje, ali samo sljedeći će biti potrebni za postupak testiranja: Nmap, Burp Suite, Wireshark i John the Ripper.

Prvi korak će biti da skeniranje mreže s alatom Nmap kako bi se otkrili otvoreni portovi i servisi u web aplikaciji. Kako bi se to napravilo, koristiti će se najpoznatiji oblik skeniranja, tzv. „SYN scan“ i dodatne parametre koji će dati informacije o verziji servisa koji se nalaze na portu i koji operacijski sustav se koristi.

Naredba koja se unosi u terminal na Kaliju je sljedeća:

„sudo nmap -sS -sV –O 192.168.56.102“

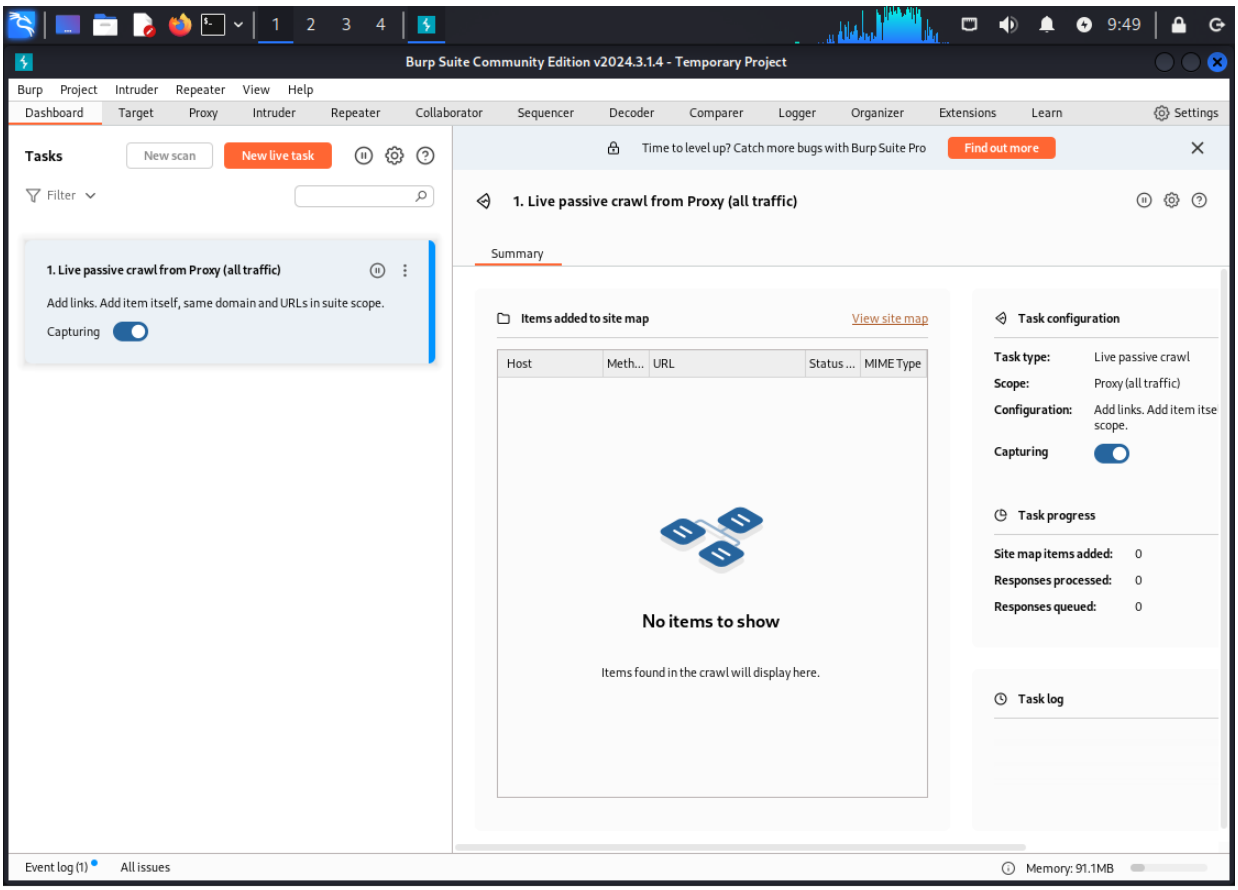
pri čemu parametar „-sS“ govoro da je riječ o Syn skenu „-sV“ daje informacije o verziji servisa, parametara „-O“ daje informaciju o operacijskom sustavu koji se koristi, a navedena IP adresa 192.168.56.102 pripada serveru na kojem se nalazi aplikacija.



Slika 6.3. Rezultat pokretanja naredbe nmap

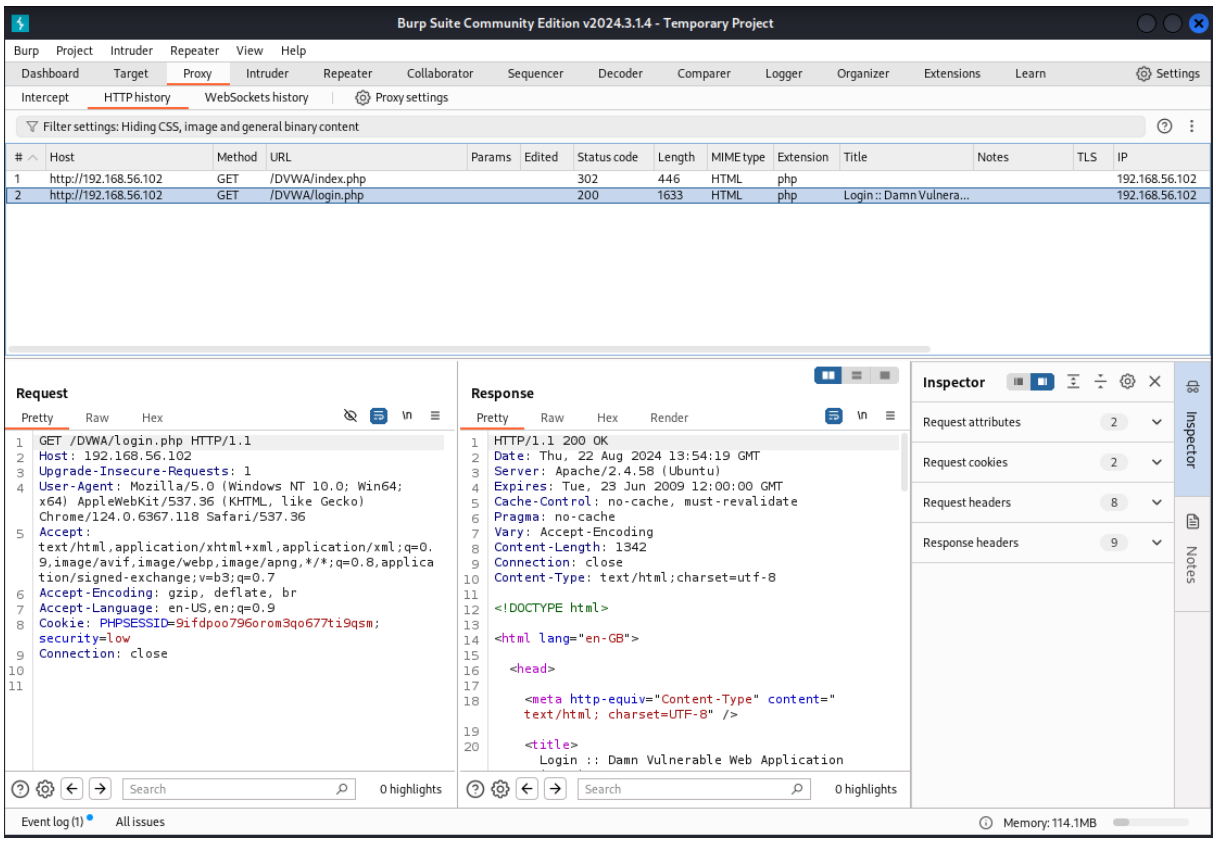
Na slici koja prikazuje izlaz za izvršenu naredbu vidi se da je port 80/tcp otvoren, te da se koristi Apache server.

Sljedeći korak je testirati sigurnost web aplikacije. Ovaj korak se može izvršiti korištenjem Metasploit Frameworka ili Burp Suite, no za potrebe ovog rada koristiti će se Burp Suite. Nakon što se otvori Burp Suite, vidjeti će se zaslon sa slike 6.6.



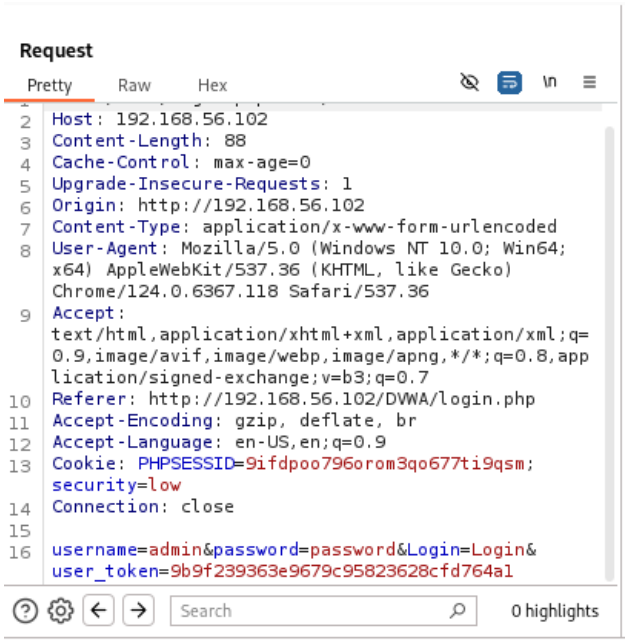
Slika 6.4. Grafičko korisničko sučelje programa Burp Suite

Sada će se koristiti opcija „Proxy“ koju nudi Burp Suite. Opcija Proxy omogućuje da Burp Suite cijelo vrijeme presreće i prikuplja HTTP zahtjeve i odgovore koji se šalju. Klikom na karticu Proxy otvara se novo sučelje na kojem je potrebno kliknuti „Open browser“ kako bi se otvorio modificirani Chrome preglednik iz kojeg će Burp Suite dohvaćati navedene HTTP zahtjeve i odgovore. Nakon što se preglednik pokrene, u njega se unosi URL na kojem se nalazi DVWA, u ovom slučaju je to „192.168.56.102/DVWA/“. Jednom kada se stranica učita može se vratiti u Burp Suite i odabrati karticu „HTTP history“ gdje će se nalaziti svi HTTP zahtjevi i odgovori koje se može proučavati.



Slika 6.5. Prikaz HTTP zahtjeva u rubrici Proxy unutar Burp Suite

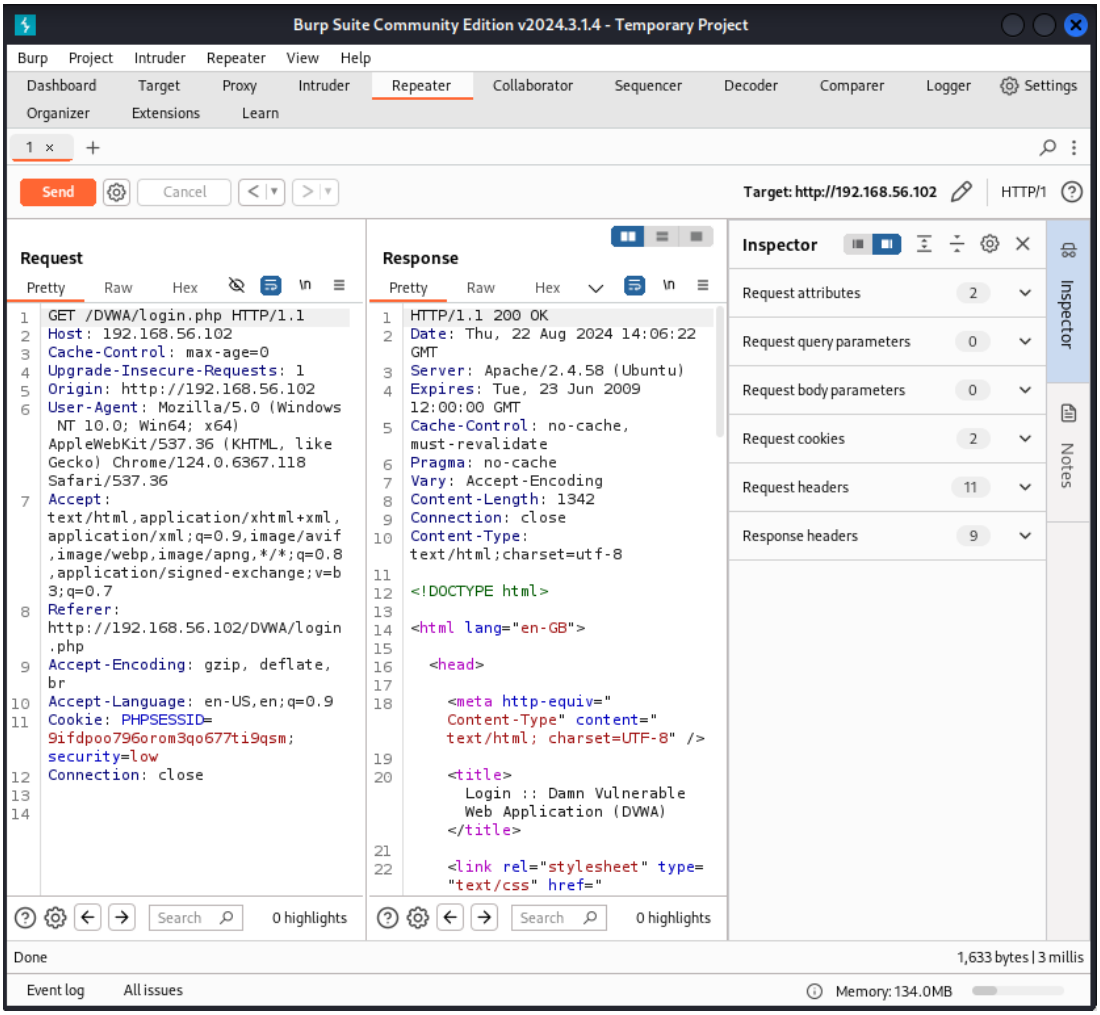
Nakon što se ulogira u DVWA aplikaciju, unutar Burp Suite-a može se vidjeti i HTTP paket koji je poslan prilikom autentifikacije i u kojem se nalaze korisničko ime i lozinka koja je unesena.



Slika 6.6. Prikaz lozinke i korisničkog imena iz dohvaćenog HTTP zahtjeva

Sa slika se može vidjeti da iz zahtjeva koji su se presreli može dobiti dosta informacija o sustavu, ali ukoliko korisničko ime i lozinka ne koriste nikakav oblik heširanja, da se i njih može dohvatiti što je također još jedna ranjivost.

Korak dalje je koristiti funkcionalnost „Repeater“ unutar Burp Suita s kojom se može ponovno slati podatke koji su se presreli. Primjerice u rubrici Proxy se može iz HTTP history rubrike poslati u Repeater neki od HTTP zahtjeva koje imamo i ponovno ga iskoristiti. Može se ponovno slati zahtjeve, te mijenjati parametre kako bi se vidjelo kako će aplikacija reagirati na njih.

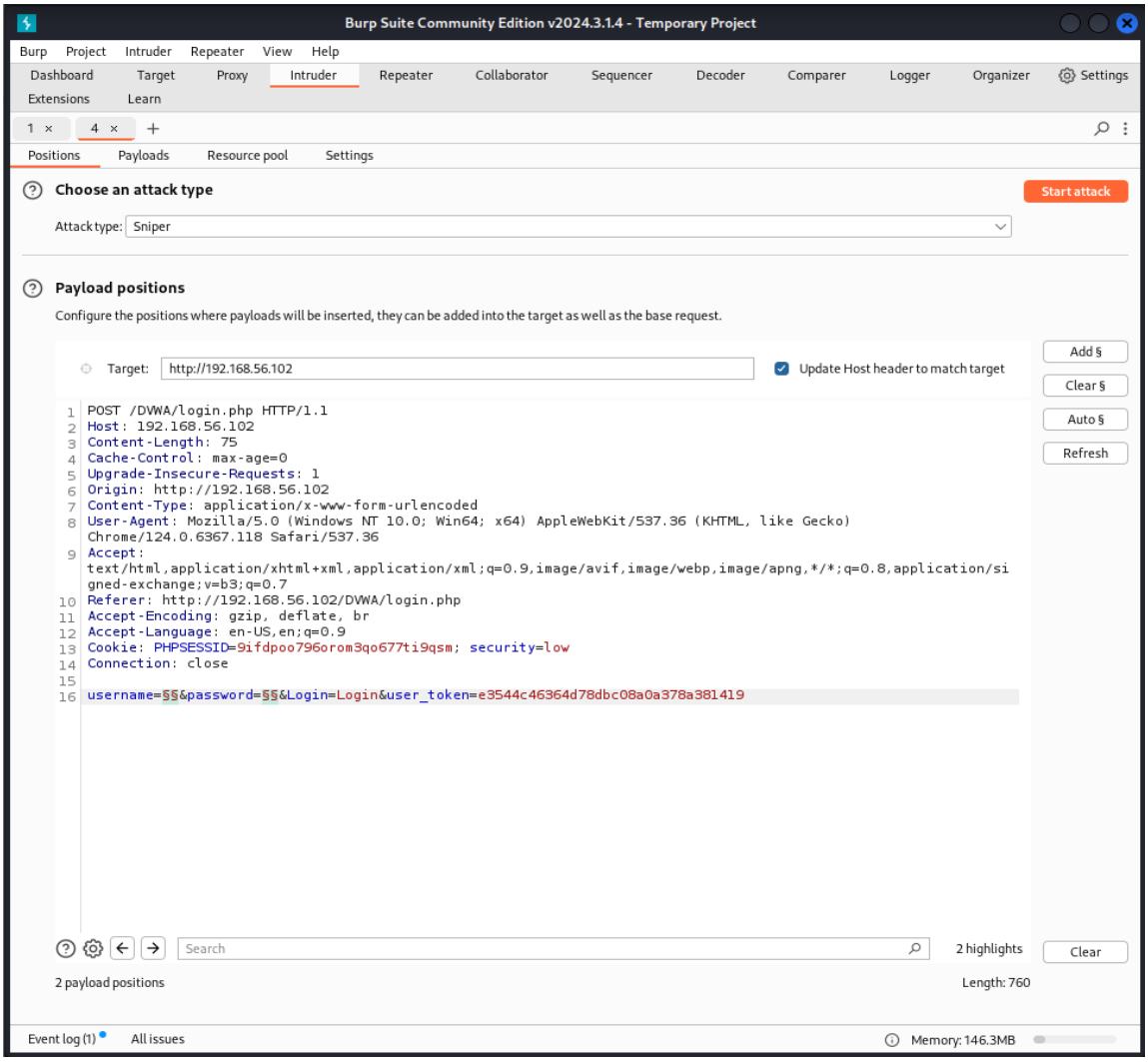


Slika 6.7. Rubrika "Repeater" unutar Burp Suite

Primjerice, pokušao se poslati HTTP zahtjev s kojim se već jedanput ulogiralo u aplikaciju, no nije moguće ponovno se ulogirati koristeći isti HTTP zahtjev jer je CSRF token bio nepravilan. CSRF je token koji se koristi kako bi se web aplikacija zaštitila od tzv. „Cross-Site Request Forgery“ napada s čime se stekla informacija da web aplikacija koristi navedeni sigurnosni mehanizam.

Još jedna korisna opcija koju nudi Burp Suite je tzv. „Intruder“ kojeg se može koristiti za „brute force“ napade kako bi probili lozinke. U Intruder alat može se poslati HTTP zahtjev za ulogiravanje korisnika čiji se zahtjev prethodno presreo u proxyu. Moguće je izmjenjivati zahtjev prema svojim potrebama. Primjerice, unutar zahtjeva, u poljima za korisničko ime i lozinku može se staviti znakove §§ tamo gdje bi stvarni podatak trebao biti. Zatim se u tabu „Payloads“ odabere kakav će se payload koristiti za brute force napad.

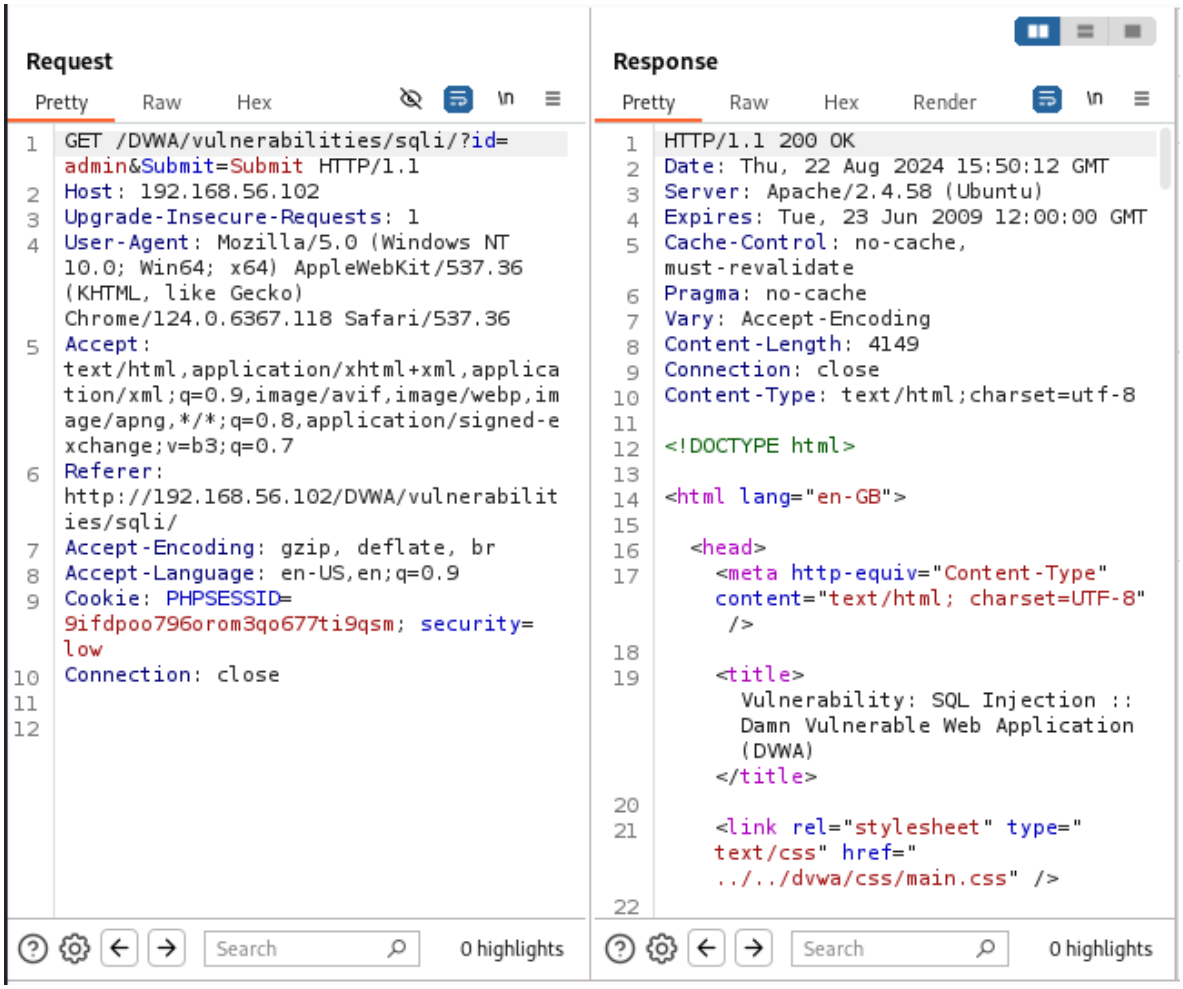
Za potrebe ovog testiranja koristio se Sniper napad i kao payload tzv. „Simple list“ odnosno jednostavna lista u koju su unesene jednostavne šifre i korisnička imena koja bi se mogli pojaviti, ali uneseno je i stvarno korisničko ime i lozinka jednog korisnika. No zbog korištenja CSRF zaštite nije se uspjelo probiti login stranicu jer se za svaku sesiju za login generira novi CSRF token, baš kako bi se spriječio ovaj oblik napada.



Slika 6.8. Intruder sučelje za konfiguraciju i provođenje brute force napada

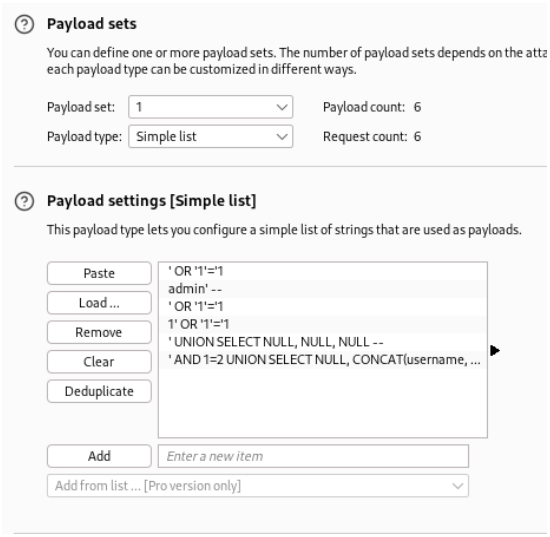
Stoga je i ovaj obrambeni mehanizam aplikacije efikasan jer se tokeni stalno mijenjaju i ne može se iskoristiti one koji su se već jedanput pojavili, već je za svaku novu autentifikaciju potreban novi token. Da aplikacija ne koristi ove tokene, mogla bi se probiti na ovaj način. Međutim, da se koristi Burp Suite Professional koji ima alat za automatsko stvaranje novih CSRF tokena, mogla bi se iskoristiti ta značajka i probiti ovaj sigurnosni mehanizam.

Sljedeća ranjivost koju se testirala bila je SQL injekcija. Prvo se u Proxyu pronašao zahtjev koji je u sebi sadržavao SQL zahtjev:



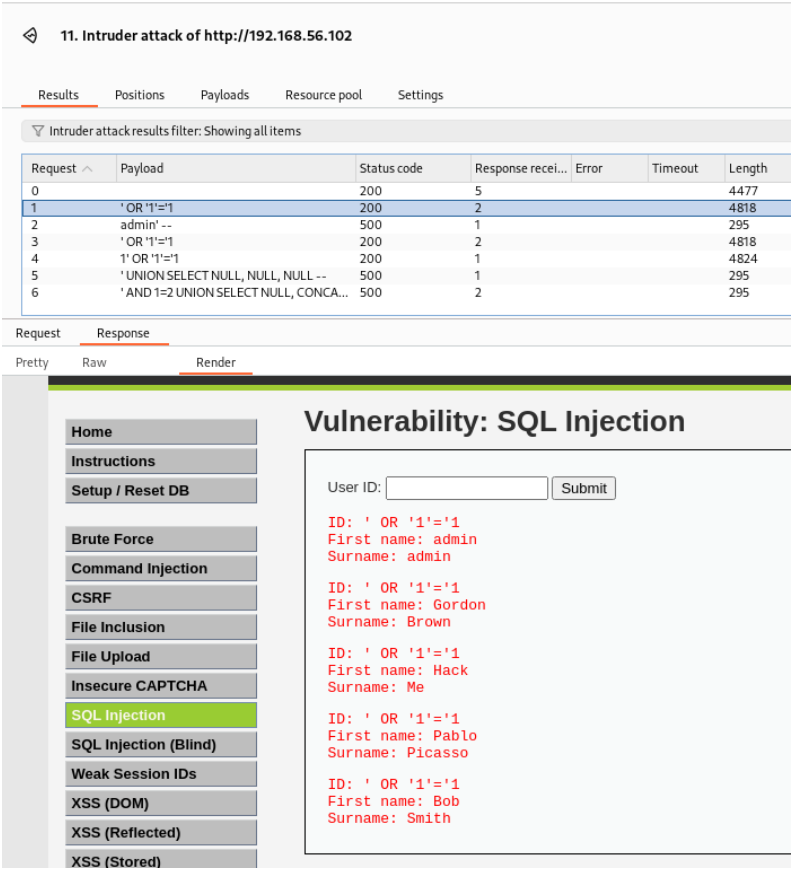
Slika 6.9. SQL zahtjev koji je pronađen u Proxyu

Nakon što je zahtjev otkriven, poslan je u Intruder kako bi se prilagodio za testiranje ranjivosti. Zatim su u dio u kojem se specificira ID korisnika u SQL zahtjevu dodani znakovi §§ koji će se zamjenjivati s upitima (enql. „queries“) koji mogu iskoristiti ranjivost SQL injectiona koristeći izraze unutar payloada koji se šalju. Sam payload se sastoji od jednostavne liste koja u sebi sadrži navedene ranjive upite s kojima se može izvesti SQL injection.



Slika 6.10. Payload za SQL injekcija napad

Nakon što je sve pripremljeno i kad je payload spreman, pokreće se napad klikom na gumb „Start attack“. Nakon izvršenog napada odmah se dobiju rezultate unutar Burp Suite-a gdje se može vidjeti HTML kod ali i renderirana stranicu na kojoj su prikazani podaci kao što se vidi na slici.

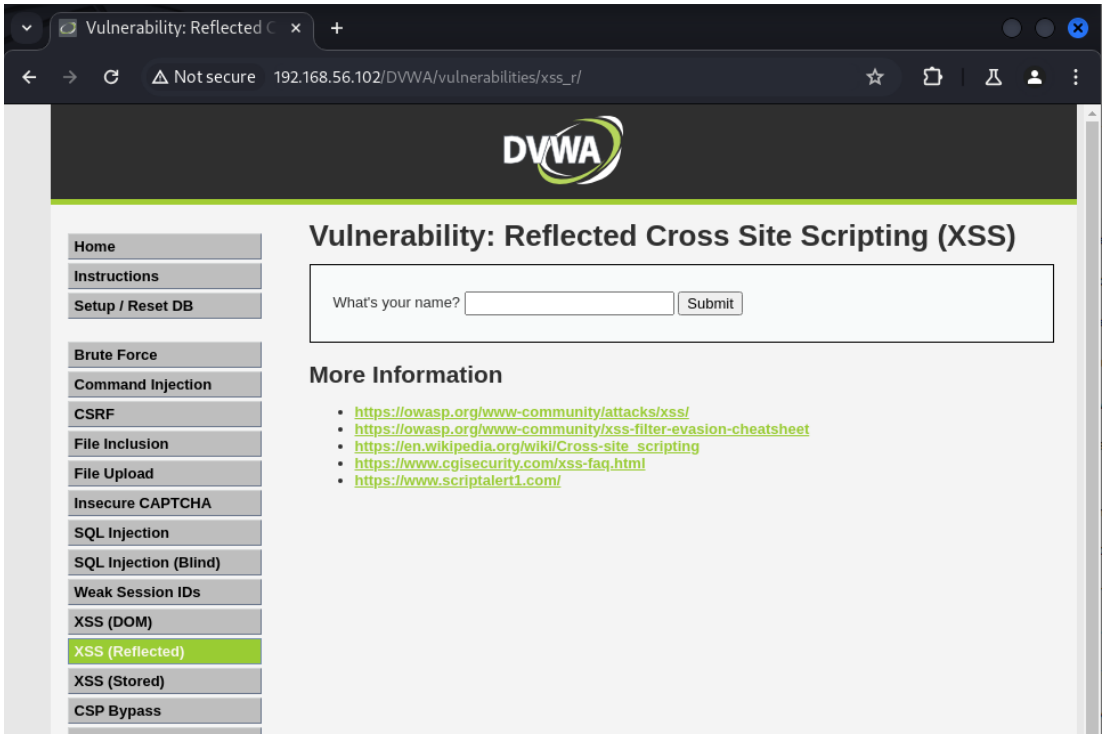


Slika 6.11. Uspješno izvršena SQL injekcija

Na slici 6.13 se vidi posljedica uspješno izvršenog napada za payload „*' OR '1'=1*“, isti rezultat je dobiven i za payload *„ 1' OR '1'=1“*. S ovim je utvrđeno da je aplikacija ranjiva na SQL injekciju. Odnosno, omogućuje se napadaču da zaobiđe autentifikaciju i prikupi osjetljive podatke iz baze podataka bez da se uopće ulogira. U trenutku napada nismo bili ulogirani u aplikaciju, payload koji se poslao ne odgovara imenu niti jednog korisnika, ali se zato dobio ispis svih imena u bazi podataka.

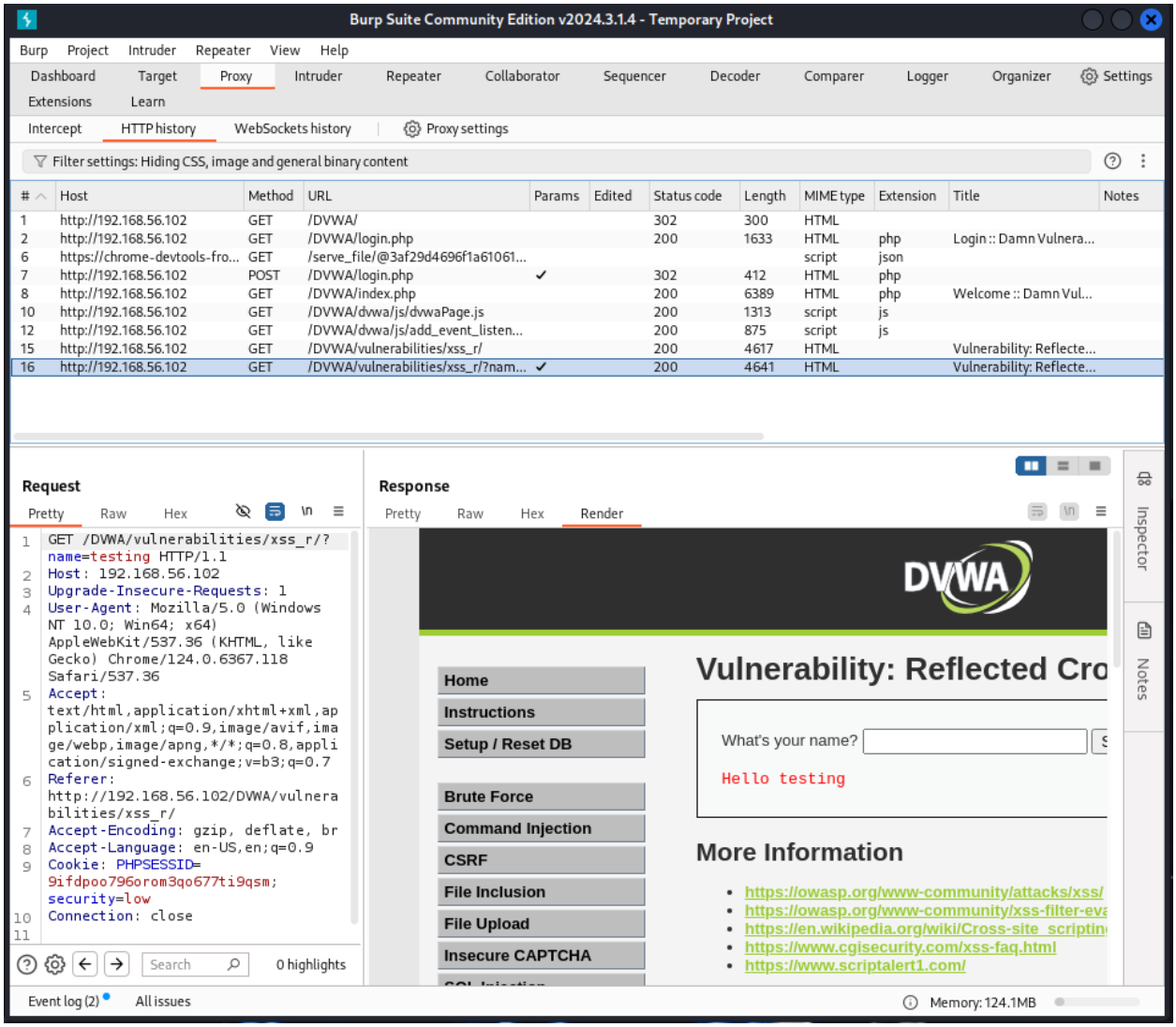
Ovaj propust se mogao izbjeći da se koriste gotovi izrazi za pristup bazi podataka, da se kao ulazna vrijednost mogu primati samo imena korisnika a ukoliko tester ili napadač unese nekakav izraz koji nije dozvoljen da zahtjev bude odbijen. Općenito, važno je da se provjeri i pročisti korisnički unos prije nego što se stvore upiti prema bazi, ali isto tako, potrebno je napraviti provjeru je li korisnik uopće ulogiran u aplikaciju i ima li ovlasti koristiti željene izraze.

Sljedeća ranjivost koja će se testirati je XSS. Kako bi otkrili XSS ranjivosti potrebno je pronaći na stranici ulazna polja koja mogu predstavljati ranjivost. Primjeri ovakvih ranjivosti su polja za unos podataka, komentari, polja za pretraživanja i bilo kakva ostala polja gdje korisnici mogu unositi podatke. DVWA u sebi ima tri primjera XSS ranjivosti koje se mogu testirati; DOM, Reflected i Stored. Za potrebe testiranja koristiti će se varijanta Reflected. Za početak je potrebno ulogirati se u DVWA ukoliko već nismo i otvoriti rubriku XSS (Reflected).



Slika 6.12. Stranica unutar DVWA za testiranje XSS Reflected ranjivosti koja u sebi sadrži polje za unos

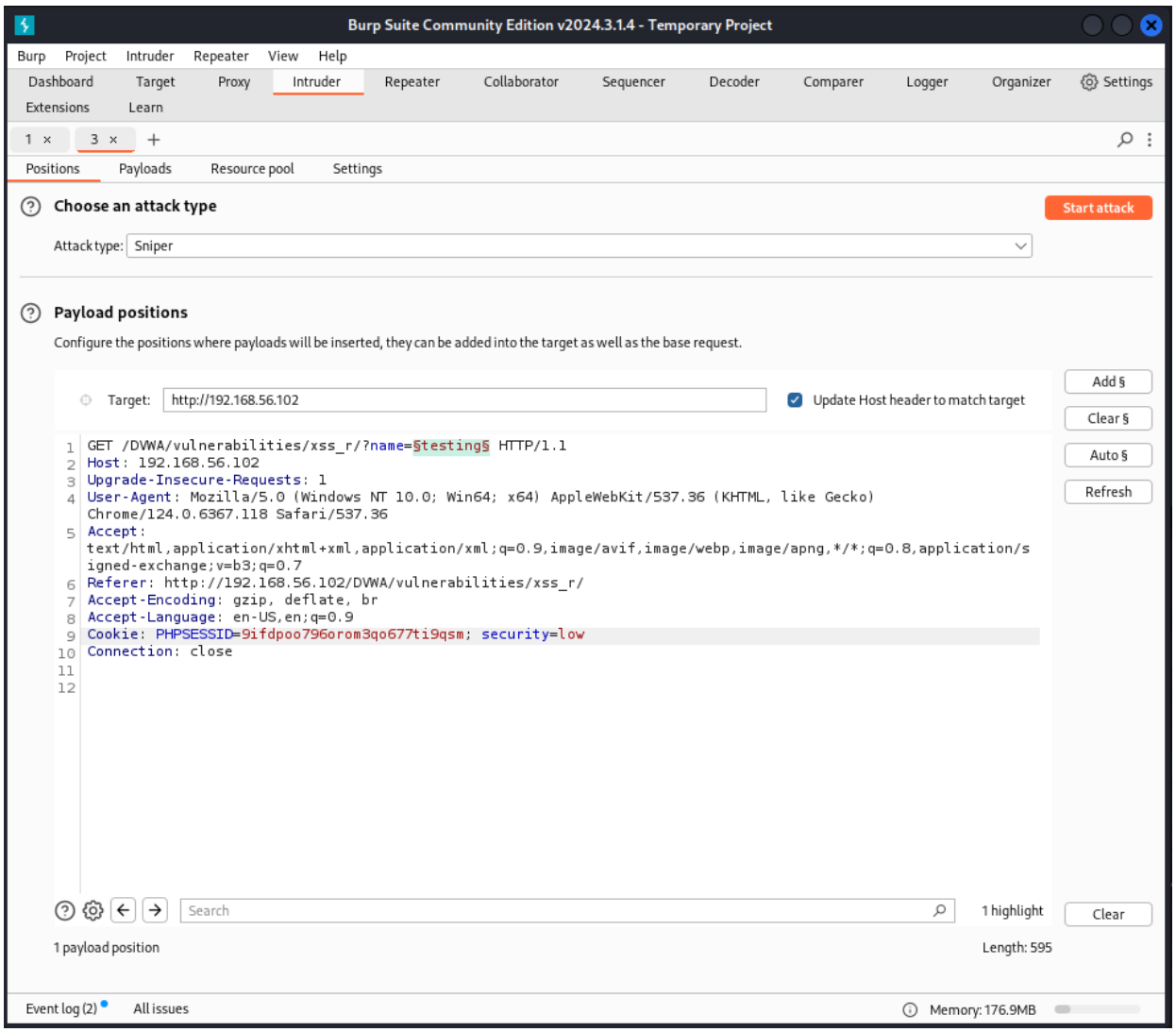
Nakon što je stranica otvorena, u polje se unosi neku vrijednost; u ovom slučaju „testing“, vraća se u Burp Suite i odabire opcija Proxy, te promatramo povijest HTTP zahtjeva koji su izvršeni. Prolaskom kroz listu pronalazi se posljednji zahtjev kojeg je Burp Suite presreo i koji odgovara posljednjoj posjećenoj stranici, odnosno stranici za testiranje XSS ranjivosti.



Slika 6.13. HTTP paket kojeg je Burp Suite presreo i koji u sebi sadrži XSS ranjivost

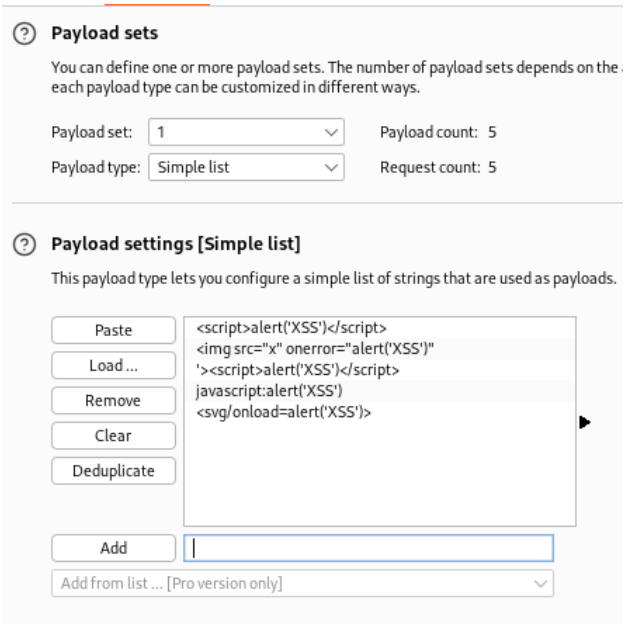
Iz slike se vidi paket kojeg je Proxy presreo; njegov sadržaj i izgled stranice s tekstom „Hello testing“ pri čemu je „testing“ tekst kojeg smo ranije unijeli.

Kao i u testiranjima prethodnih ranjivosti, potrebno je presretani HTTP paket koji sadrži ranjivost poslati u rubriku „Intruder“ kako bi se iskoristio za testiranje. U toj rubrici će se iskoristiti ovaj paket kako bi u njega ubacili poznate XSS payloade i poslali ga u web aplikaciju s ciljem utvrđivanja postoje li ranjivosti.



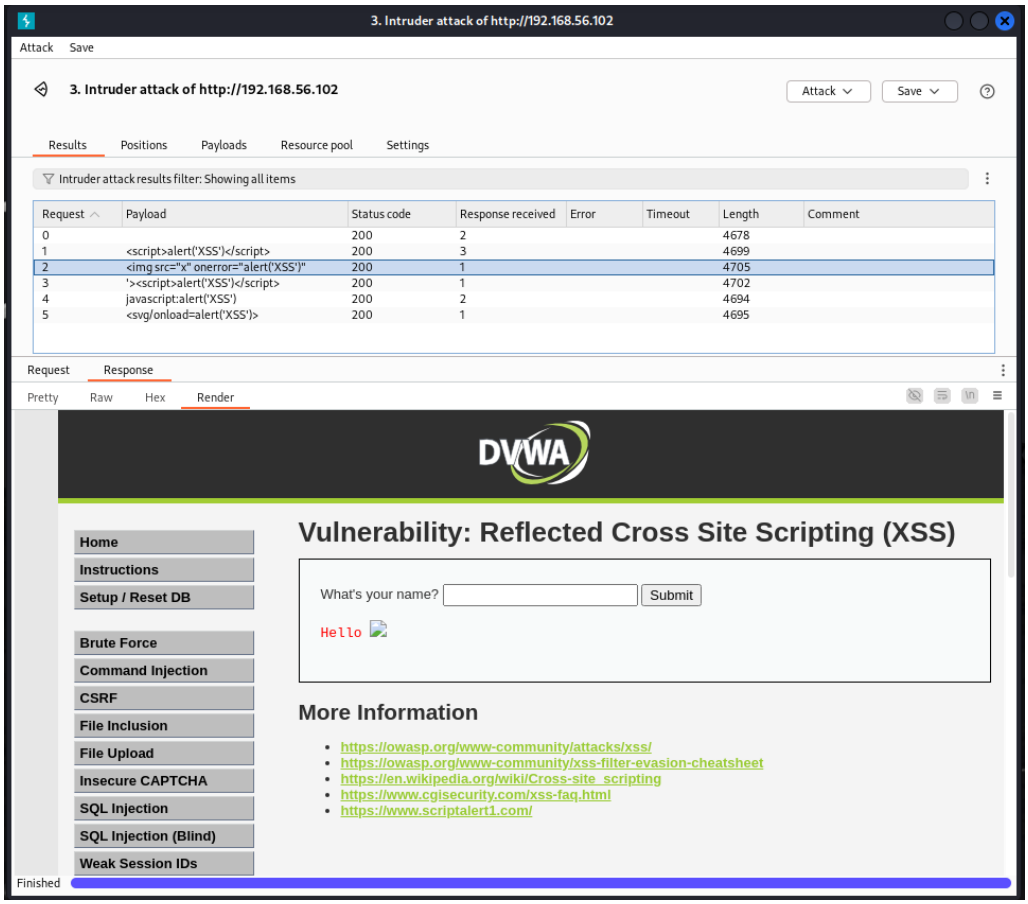
Slika 6.14. Pripremanje zahtjeva za XSS testiranje

Nakon što je zahtjev poslan u Intruder, potrebno je napraviti sljedeću promjenu: umjesto „name=testing“ napisati: „name=§testing§“. Sa znakovima § je označena riječ „testing“ jer se na tom mjestu u formi za unos teksta unosi tekstualni podatak, a prilikom testiranja će se automatski umjesto riječi „testing“ unositi različiti XSS payloadi za koje će se provjeravati izlaz web aplikacije. Sljedeći korak je definirati navedene XSS payloade koji će se koristiti za testiranje. Kao i prije, koristiti će se jednostavnu listu u koju će se dodati česti primjeri za iskorištavanje XSS ranjivosti (slika 6.17).



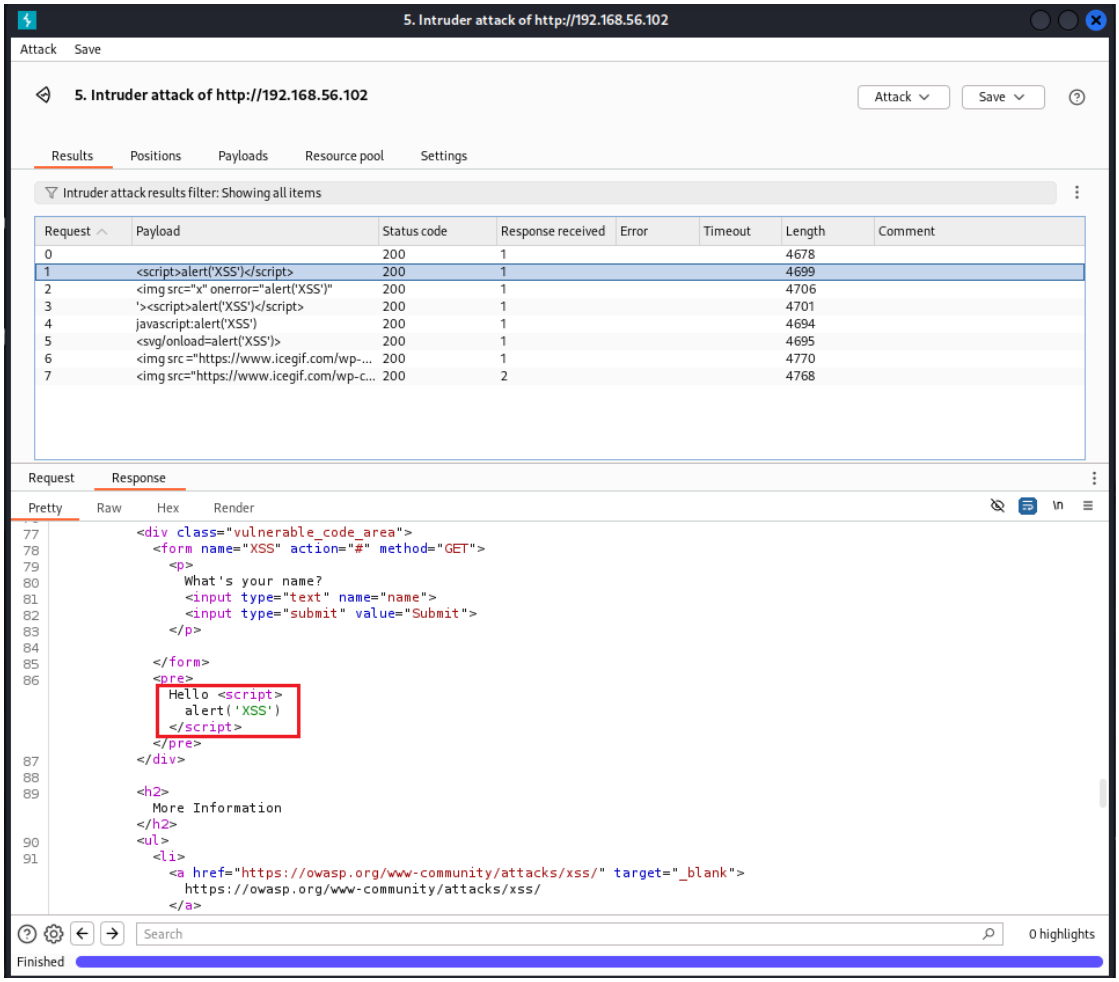
Slika 6.15. XSS payloadi koji testiraju razne načine umetanja zlonamjernog koda

Sada se može pokrenuti napad. Nakon što je napad izvršen, promatraju se rezultati napada:



Slika 6.16. Rezultati XSS napada

Napad je uspješno izvršen i utvrđeno je da postoje XSS ranjivosti. Za primjer sa slike 6.18 može se vidjeti da je moguće ubaciti sliku u polje koje bi samo trebalo primati tekst. Daljnjim prolaskom kroz rezultate napada ustanovljeno je da kroz ovo sučelje moguće čak ubaciti i skriptu koja će se onda pokrenuti. Kao payload za ovaj scenarij koristio se izraz *„<script>alert('XSS')</script>“* koji će na stranici samo prikazati obavijest „XSS“, a korisnik će na stranici vidjeti samo tekst „Hello“, bez da piše ikakvo ime. U stvarnim uvjetima, napadači bi unutar <script> tagova mogli staviti maliciozan kod koji bi mogao uzrokovati štetu.



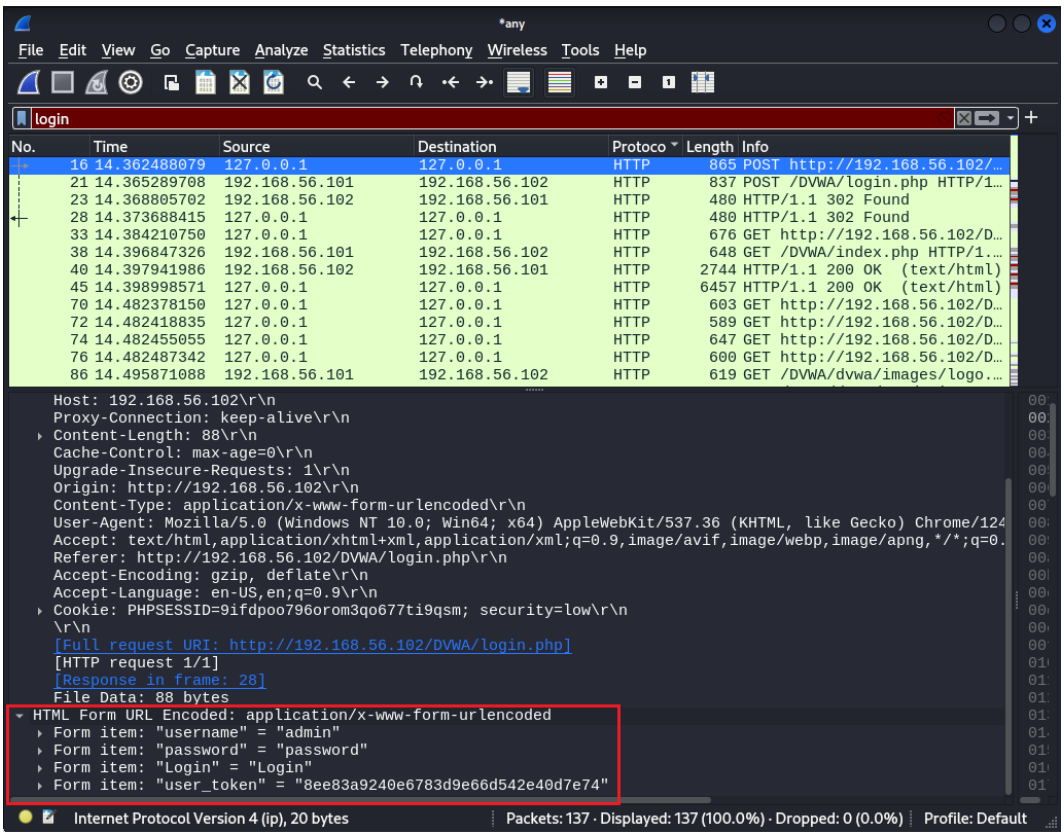
Slika 6.17. Ubačena skripta koristeći XSS ranjivosti

Ovim testiranjem ustanovilo se da aplikacija ima XSS ranjivosti koje je potrebno otkloniti. Preporuka kako ovo učiniti bila bi da se korisnički ulazi pročiste i formatiraju prije nego li se zahtjev izvrši na web stranici. Dobra ideja bila bi koristiti tzv. Content Security Policy (CSP) kako bi se ograničilo što se može učitati i izvršavati na stranici. Isto tako, ukoliko je moguće, potrebno je izbjegavati omogućavanje korisnicima da sami direktno umetaju HTML sadržaj.

Do sada su ispitane tri potencijalne ranjivosti: brute force napad, SQL injekcija i XSS. Ustanovljeno je da je aplikacija ranjiva na SQL injekciju i XSS napade, ali ukoliko bi se koristili odgovarajući alati koji su dio plaćene verzije Burp Suite-a, ustanovilo bi se da je aplikacija ranjiva i na brute force napad koji je ujedno bio i prvi koji se provjeravao.

Sljedeći alat je Wireshark koji se koristio i na kolegiju Računalne mreže. Služi za snimanje i analizu mrežnog prometa s ciljem da se otkriju osjetljive informacije. Za potrebe ovog testiranja nije previše korišten jer unutar samog Burp Suita možemo na puno pregledniji način vidjeti HTTP zahtjeve koji su nam bitni i koje možemo iskoristiti. Isto tako, Burp Suite daje više informacijama o samim zahtjevima, dok ih Wireshark promatra samo kao obične pakete.

No pravilnim korištenjem i dobrim postavljanjem filtera, te ukoliko paketi nisu dobro zaštićeni, moguće je pronaći pakete iz kojih se može saznati informacije koje ne bi trebale biti dostupne.



Slika 6.18. Otkriven paket koji u sebi sadrži osjetljive korisničke podatke

Primjerice, zbog nekorištenja enkripcije podataka, iz paketa koji se šalju između računala s kojeg se testira i servera, uspio se presresti paket unutar kojeg su detaljnijom analizom pronađeni korisnički podaci.

Ukoliko bi u stvarnom svijetu napadači uspjeli upasti u mrežu gdje mogu presretati podatke, na taj način bi mogli otkriti korisničko ime i lozinku administratora što predstavlja veliki sigurnosni rizik. Dakle, još jedna pronađena ranjivost je nekorištenje enkripcije.

Iako nema mnoge značajke poput Burp Suite-a, Wireshark je i dalje jako koristan za promatranje prometa između servera i ostalih uređaja jer se može steći uvid kako se odvija komunikacija. U ovom slučaju se radi loše zaštite podataka čak uspjelo otkriti osjetljive podatke koje bi napadači mogli iskoristiti za svoje ciljeve.

Posljednji alat koji će se koristiti u sklopu ovog testiranja je John the Ripper, alat za probijanje lozinki. Demonstrirati će se kako se s ovim alatom mogu probiti lozinke koje koriste hashing. S obzirom da postoji pristup bazi podataka, u njoj se može pristupiti tablici koja sadrži korisnička imena i njihove hashirane lozinke. Te informacije će biti iskorištene za ovu demonstraciju.

```
Database changed
mysql> show tables;
+-----------------+
| Tables_in_dvwa |
+-----------------+
| guestbook      |
| users          |
+-----------------+
2 rows in set (0.00 sec)

mysql> select user, password from users;
+---------+----------------------------------+
| user    | password                         |
+---------+----------------------------------+
| admin   | 5f4dcc3b5aa765d61d8327deb882cf99 |
| gordonb | e99a18c428cb38d5f260853678922e03 |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 |
+---------+----------------------------------+
5 rows in set (0.00 sec)
```

Slika 6.19. Prikaz tablice s korisničkim podacima u bazi podataka

Sljedeći korak je napraviti tekstualnu datoteku s navedenim podacima i formatirati je u oblik koji je čitljiv John the Ripperu. Format ima sljedeći oblik: <korisničko ime>:<lozinka>. Kreirana datoteka ima ime „pwordtesting.txt“, a njen sadržaj se nalazi na slici 6.22.

```
admin:5f4dcc3b5aa765d61d8327deb882cf99
gordonb:e99a18c428cb38d5f260853678922e03
1337:8d3533d75ae2c3966d7e0d4fcc69216b
pablo:6f047d9d5bb04cadee8d5c7e19e59b7
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Slika 6.20. Tekstualna datoteka s korisničkim podacima formatirana za JtR

Nakon što smo spremili datoteku s korisničkim podacima, može se krenuti sa samim postupkom probijanja lozinke. Za potrebe testiranja koristiti će se unaprijed definirana lista riječi koja se nalazi na Kali linuxu pod imenom „rockyou.txt“ i dio je direktorija „wordlists“. Navedena lista sadrži bazu najčešće korištenih lozinki na internetu

Korištenjem naredbe:

„john --format=Raw-MD5 --fork=2 --wordlist=/usr/share/wordlists/rockyou.txt ./pwordcracking.txt“

pokreće se postupak probijanja lozinke, te nakon nekog vremena, JtR uspješno dešifrira šifre koje su hashirane korištenjem MD5 formata.

```
┌──(kali㉿kali)-[~/Documents]
└─$ john --format=Raw-MD5 --fork=2  --wordlist=/usr/share/wordlists/rockyou.txt ./pwordcracking.txt

Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
Node numbers 1-2 of 2 (fork)
password         (admin)
abc123           (gordonb)
charley          (1337)
Press 'q' or Ctrl-C to abort, almost any other key for status
2 3g 0:00:00:00 DONE (2024-08-24 05:13) 75.00g/s 38400p/s 38400c/s 57600C/s irene..dangerous
1 0g 0:00:00:00 DONE (2024-08-24 05:13) 0g/s 9960Kp/s 9960Kc/s 29881KC/s  iluveddie1.a6_123
Waiting for 1 child to terminate
Session completed.
```

Slika 6.21. Rezultat probijanja lozinki

Na slici 6.23 se vide dešifrirane lozinke korisnika koje su na slici 6.22 vidljive u svom hashiranom obliku. Može se vidjeti da lozinke za korisnika „pablo“ i „smithy“ nisu probijene što može značiti da unutar datoteke „rockyou.txt“ ne postoje odgovarajuće lozinke. Druga vjerojatnost je da lozinke koriste neobične znakove, imaju veću duljinu ili složenost.

Cilj ovog probijanja lozinki je bio demonstrirati kako slabe lozinke korisnika, pogotovo onih koji imaju viša prava a to je u ovom slučaju administrator, može komprimirati sigurnost web aplikacije.

Ono što je u ovom slučaju još veći razlog za brigu je to što su lozinke probijene u jako kratkom vremenu, koristeći najpopularniji „wordlist“ što bi u stvarnim uvjetima napadačima uvelike olakšalo posao. Stoga je od ključne važnosti da korisnici koji imaju veća prava u web aplikaciji ili su njeni administratori koriste adekvatne i snažne lozinke kako bi se napadačima onemogućio pristup.

6.3. Zaključne misli

Kroz testiranje aplikacije DVWA ustanovljeno je kako ona posjeduje brojne ranjivosti koje su dokazane kroz postupak penetracijskog testiranja. Proveo se Gray box postupak testiranja u kojem su ispitane česte ranjivosti s kojima se moguće sresti na Internetu.

Prvi korak je bio mapiranje mrežnog okruženja web aplikacije s ciljem otkrivanja otvorenih portova i servisa na njima. Nmap je jedino uspio otkriti da je port 80/tcp otvoren i da se na njemu nalazi Apache server koji služi za „hostanje“ web aplikacije.

Nakon mapiranja mrežnog okruženja, krenulo se s testiranjem sigurnosti aplikacije koristeći program Burp suite u kojem je ispitana sigurnost aplikacije u odnosu na ranjivosti kao što su „fuzzing“, SQL injekcija i XSS. Prvi korak je bio korištenjem proxy značajke Burp Suitea pronaći HTTP paket koji se koristi za login i iz njega dohvatiti korisničke podatke. Zatim se za pokušaj „fuzzinga“ probao iskoristiti HTTP paket s kojim se korisnik već jednom uspješno ulogirao da se ponovno ulogira u aplikaciju. To nije uspjelo radi toga što aplikacija koristi CSRF zaštitu, ali bi se i ovaj mehanizam probio ukoliko bi se koristila Professional inačica alata Burp Suite.

Sljedeća testirana ranjivost je bila ranjivost na SQL injekciju za koju je dokazano da ista ranjivost postoji i da ju je potrebno otkloniti. Nakon toga je aplikacija testriana za XSS ranjivosti za što je isto ustanovljeno da ranjivost postoji i da ju je potrebno otkloniti. Za obe ranjivosti su prilikom testiranja navedeni postupci koje je potrebno napraviti kako bi se ranjivost otklonila i spriječilo njihovo iskorištavanje.

Wireshark je korišten kako bi se presretao, snimao i analizirao mrežni promet između računala s kojeg je testirana aplikacija i servera na kojem se nalazi aplikacija, te je ustanovljeno analizom jednog paketa da je moguće iz njega izvući lozinku i korisničko ime korisnika zbog manjka enkripcije podataka.

Za kraj je uz pomoć alata John the Ripper demonstrirano kako se slabe lozinke korisnika mogu jednostavno probiti koristeći poznate wordliste i kakav to sigurnosni rizik može predstavljati za cijelu aplikaciju.

Naposlijetku se može reći da je web aplikacija DVWA puna ranjivosti koje bi kod stvarnih aplikacija bile pogubne za njenu sigurnost, ali i sigurnost njenih korisnika. Stoga je od ključne važnosti da prilikom razvoja web aplikacija obratimo pažnju na sigurnosne mehanizme koje možemo uvesti ali i na otklanjanje ranjivosti koje bi napadači mogli iskoristiti.

7. ZAKLJUČAK

Prije početka pisanja ovog završnog rada posjedovao sam znanje o web sigurnosti koje sam stekao kroz svoje dosadašnje obrazovanje, ali kroz pisanje ovog rada dobio sam uvid u brojne napredne tehnike, alate i vještine koje su dio web sigurnosti. Penetracijsko testiranje je također jedna od brojnih disciplina koje spadaju u područje web sigurnosti.

Ovaj rad pružio je sveobuhvatan pregled ove discipline, počevši od samih začetaka web sigurnosti i kako se razvijala kroz vrijeme, te kako je to utjecalo na potrebe za naprednijim sigurnosnim mehanizmima, do detaljnog istraživanja o penetracijskom testiranju koje je pokrivalo teoretski i praktičan dio.

Povijest web sigurnosti je pokazala kako prijetnje i ranjivosti evoluiraju paralelno s razvojem tehnologije i mrežne infrastrukture. Razumijevanje ove pojave objašnjava kako konstantno treba unaprjeđivati postojeće i razvijati nove metode za očuvanje web sigurnosti. Penetracijsko testiranje je metoda koja je proizašla iz ovih potreba i danas je neophodna za ispitivanje sigurnosti web aplikacija.

Kroz istraživanje se ustanovilo da je penetracijsko testiranje složen postupak koji se sastoji od više kompleksnih faza i brojnih izazova s kojima se testeri moraju suočiti kako bi uspješno izvršili testiranje i otkrili ranjivosti u sustavu. Pokazalo se da je za uspješno provođenje testiranja potrebno dobro planiranje, detaljna analiza, visok stupanj stručnosti i znanja testera, kao i sposobnost nošenja s brojnim izazovima koji se mogu javiti tijekom testiranja.

U radu su se detaljno opisivale različite vrste penetracijskog testiranja; Black box, White box i Gray box testiranje. Za svaku vrstu testiranja detaljno su opisane njene karakteristike, cijeli proces testiranja; od početka do pisanja izvještaja, ali i uvjeti u kojima se provodi testiranje koji znatno utječu na sam proces testiranja. Prilikom opisivanja svake vrste testiranja usporedilo se po čemu je ta vrsta testiranja drugačija od drugih i kakvi su uvjeti i okolnosti za korištenje svake vrste.

Opisani su brojni alati kojima se služe testeri, njihove karakteristike i područja primjene. Svaki alat nudi funkcionalnosti i prednosti koje su jedinstvene za njega i ima važnu ulogu u cjelokupnom procesu penetracijskog testiranja.

Zadnja cjelina ovog rada je stvarni primjer penetracijskog testiranja koji demonstrira praktičnu primjenu prethodno opisanih teorijskih koncepata i alata u stvarnom okruženju, te čitatelju daje uvid u pripremu okruženja i postupke koji su potrebni kako bi se testiranje web aplikacije uspješno izvršilo.

Zaključno, penetracijsko testiranje je neophodan proces za osiguravanje sigurnosti web aplikacija za kojim je svakim danom sve veća potreba. Od iznimne je važnosti da profesionalci koji se bave ovim oblikom testiranja konstantno unaprjeđuju svoje znanje i prilagođavaju se novim tehnologijama radi svakodnevnog razvoja novih prijetnji koje mogu iskoristiti ranjivosti web aplikacija. Ovaj rad naglašava važnost sustavne, temeljite i strateške primjene penetracijskog testiranja kako bi se osigurala visoka razina sigurnosti u sve složenijem digitalnim okruženjima s kojima se danas susrećemo.

8. LITERATURA

[1] P. W. Singer i Allan Friedman: „Cybersecurity And Cyberwar: What Everyone Needs to Know“, Oxford University Press, 2014.

[2] Barry M. Leiner i dr.: „A Brief History of the Internet“, s Interneta, https://www.internetsociety.org/internet/history-internet/brief-history-internet/, 13. rujan, 2017.

[3] Edgescan: „2023 Vulnerability Statistics Report“, s Interneta, https://www.edgescan.com/wp-content/uploads/2024/03/2023-Vulnerability-Statistics-Report.pdf, 15. kolovoz, 2024.

[4] Federal Bureau of Investigation (FBI): „Internet Crime Report 2023“, https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3Report.pdf, s Interneta, 2023.

[5] Fortinet: članak „Recent Cyber Attacks“, https://www.fortinet.com/resources/cyberglossary/recent-cyber-attacks, 2024

[6] Dafydd Stuttard i Marcus Pinto: "The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws, Second Edition", Wiley Publishing, Inc., 2011

[7] Payment Card Industry (PCI) Security Standards Council: *"Penetration Testing Guidance: Information Supplement, Version 1.1"*, s Interneta, https://listings.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf, kolovoz 2017.

[8] Europska unija: "Uredba (EU) 2016/679 Europskog parlamenta i Vijeća od 27. travnja 2016. o zaštiti pojedinaca u vezi s obradom osobnih podataka i o slobodnom kretanju takvih podataka (Opća uredba o zaštiti podataka)", s Interneta, https://eur-lex.europa.eu/legal-content/HR/TXT/?uri=CELEX:32016R0679, pristupljeno 15. kolovoza 2024.

[9] OWASP Foundation: "OWASP Web Security Testing Guide - 2. Introduction," s Interneta, https://owasp.org/www-project-web-security-testing-guide/latest/2-Introduction/, pristupljeno 16. kolovoza, 2024.

[10] Carrie Crow: "Best Port Scanners: 9 Scanners to Quickly Find Open Ports," s Interneta, https://securitytrails.com/blog/best-port-scanners, pristupljeno 16. kolovoza, 2024.

[11] Wikipedia: „Burp Suite,“ s Interneta, https://en.wikipedia.org/wiki/Burp_Suite, pristupljeno 16. kolovoza, 2024.

[12] GeeksforGeeks: „Software Testing - White Box Penetration Testing,“ s Interneta, https://www.geeksforgeeks.org/software-testing-white-box-penetration-testing/, pristupljeno 16. kolovoza, 2024.

[13] Astra Security: „Gray Box Penetration Testing,“ s Interneta, https://www.getastra.com/blog/security-audit/gray-box-penetration-testing/, pristupljeno 16. kolovoza, 2024.

[14] Metasploit Documentation: "Metasploit Documentation," s Interneta, https://docs.metasploit.com/, pristupljeno 16. kolovoza, 2024.

[15] Nmap Documentation: "Nmap Documentation," s Interneta, https://nmap.org/, pristupljeno 18. kolovoza, 2024.

[16] Nmap Documentation: "Scan Methods" s Interneta, https://nmap.org/book/scan-methods.html, pristupljeno 18. kolovoza, 2024.

[17] Nmap Documentation: "Host Discovery," s Interneta, https://nmap.org/book/man-host-discovery.html, 18. kolovoz, 2024.

[18] Nmap Documentation: "OS Detection," s Interneta, https://nmap.org/book/man-os-detection.html, 18. kolovoz, 2024.

[19] Nmap Documentation: "Nmap Scripting Engine (NSE)," s Interneta, https://nmap.org/book/man-nse.html, 18. kolovoz, 2024.

[20] PortSwigger Documentation: "Burp Suite Documentation," s Interneta, https://portswigger.net/burp/documentation, 18. kolovoz, 2024.

[21] Kennedy, D., O'Gorman, J., Kearns, D., & Aharoni, M.: "Metasploit: The Penetration Tester's Guide," No Starch Press, 2011.

[22] G. F. Lyon: "Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning," Insecure.Com LLC, 2009.

[23] Combs, G., & Orebaugh, A.: "Wireshark Network Analysis (Second Edition): The Official Wireshark Certified Network Analyst Study Guide," Syngress, 2014.

[24] Eckert, P.: "The Book of GPG: Encryption for the Masses," No Starch Press, 2010.

[25] Openwall: "John the Ripper: Password Cracking Software," s Interneta, https://www.openwall.com/john/, 20. kolovoz, 2024.

[26] digininja: "Damn Vulnerable Web Application (DVWA)," s Interneta, https://github.com/digininja/DVWA, 21. kolovoz, 2024.

9. POPIS OZNAKA I KRATICA

- API – engl. Application Programming Interface
- ARP – engl. Address Resolution Protocol
- ARPANET – engl. Advanced Research Project Agency Network
- CERN – franc. Conseil Européen pour la Recherche Nucléaire: Europsko vijeće za nuklearna istraživanja
- CPU – engl. Central Processing Unit
- CRUD – engl. Create, Read, Update, Delete
- CSRF – engl. Cross-Site Request Forgery
- CSP – engl. Content Security Policy
- DHCP – engl. Dynamic Host Configuration Protocol
- DES – engl. Data Encryption Standard
- DNS – engl. Domain Name System
- DOM – engl. Document Object Model
- DVWA – engl. Damn Vulnerable Web Application
- FBI – engl. Federal Bureau of Investigation
- FTP – engl. File Transfer Protocol
- GDPR – engl. General Data Protection Regulation
- HTML – engl. Hypertext Markup Language
- HTTP – engl. Hypertext Transfer Protocol
- HTTPS – engl. HTTP Secure
- ICMP – engl. Internet Control Message Protocol
- ID korisnika – engl. User ID
- IP – engl. Internet Protocol
- IT – engl. Information Technology: Informacijska Tehnologija
- JtR – engl. John the Ripper
- LM – engl. LAN Manager
- MD5 – engl. Message-Digest Algorithm 5
- NAT – engl. Network Address Translation
- NTLM – engl. NT LAN Manager
- OS – engl. Operating System
- PHP – engl. Hypertext Preprocessor
- SNMP – engl. Simple Network Management Protocol

- SQL – engl. Structured Query Language: Strukturni upitni jezik
- TCP – engl. Transmission Control Protocol
- TCP/IP – engl. Transmission Control Protocol/Internet Protocol
- UDP – engl. User Datagram Protocol
- VPN – engl. Virtual Private Network
- WWW – engl. World Wide Web
- XSS – engl. Cross-Site Scripting

10. SAŽETAK I KLJUČNE RIJEČI NA HRVATSKOM JEZIKU

Ovaj rad bavi se penetracijskim testiranjem, ključnom metodom za procjenu sigurnosti informacijskih sustava i web aplikacija. U uvodnom dijelu razmatra se povijest web sigurnosti i pojave potrebe za penetracijskim testiranjem i naglašava potreba za njegovom primjenom u današnjem digitalnom okruženju. Penetracijsko testiranje omogućuje identifikaciju ranjivosti unutar web aplikacija i informacijskih sustava koje bi napadači mogli iskoristiti za ugrožavanje sustava.

Dalje, rad analizira proces penetracijskog testiranja, uključujući metode, izazove i pristupe poput Black box, White box i Gray box pristupa. Pregled i usporedba alata koji se koriste u penetracijskom testiranju, poput Burp Suitea i Nmapa, pruža dublji uvid u njihove prednosti i nedostatke ali i opisuje njihovu potrebu i ulogu u cjelokupnom procesu penetracijskog testiranja.

Praktičan dio rada uključuje primjer stvarnog penetracijskog testiranja, prikazujući konkretne korake i analizu rezultata. Zaključak naglašava važnost penetracijskog testiranja kao neizostavnog dijela strategije informacijske sigurnosti te potrebu za kontinuiranim usavršavanjem u ovom području.

Ključne riječi: Penetracijsko testiranje, Web sigurnost, Ranjivost aplikacija, Sigurnosni alati, Vrste testiranja, Black box, White box, Gray box, Nmap, Burp Suite, Metasploit Framework, Wireshark, John the Ripper, Web aplikacija

11. SUMMARY AND KEYWORDS IN ENGLISH LANGUAGE

This paper deals with penetration testing, a key method for assessing the security of information systems and web applications. The introduction discusses the history of web security and the rise of the need for penetration testing, emphasizing the necessity of its application in today's digital environment. Penetration testing enables the identification of vulnerabilities within web applications and information systems that attackers could exploit to compromise the systems.

Furthermore, the paper analyzes the penetration testing process, including methods, challenges, and approaches such as Black box, White box, and Gray box testing. A review and comparison of tools used in penetration testing provides deeper insight into their advantages and disadvantages, as well as describing their necessity and role in the overall penetration testing process.

The practical part of the paper includes an example of a real-world penetration testing, demonstrating specific steps and analyzing the results. The conclusion emphasizes the importance of penetration testing as an indispensable part of an information security strategy and the need for ongoing improvement in this field.

Keywords: Penetration Testing, Web Security, Application Vulnerabilities, Security Tools, Types of Testing, Black Box, White Box, Gray Box, Nmap, Burp Suite, Metasploit Framework, Wireshark, John the Ripper, Web Application